品質保証部門における W モデル適用の検討と実践

Examination and practice of W model application in a quality assurance unit

株式会社日立製作所 情報・通信システム社 IT プラットフォーム事業本部
開発統括本部 プラットフォーム QA 本部 ソフト品質保証部
Hitachi, Ltd., Information & Telecommunication Systems CompanyIT Platform Division Group,
IT Platform R & D Management Division, Platform Quality Assurance Operation, Software Quality Assurance
○富田 貴仁 [1)]　秦泉寺 貴文 [1)]　高山 啓 [1)]
○Tomida Takahito [1)]　Jinsenji Takafumi [1)]　Takayama Hiraku [1)]

Abstract

In our organization, large-scale software is mostly developed by the waterfall model, and quality improvement in upper process is always our subject. In this paper, we study an application of the W model, including the measures in output of quality assurance department and process schedule, for solving quality issues of the output of developmentin upper process at an early stage.

By carrying out this new process of W model, we have achieved quality improvement in the functional specification documents designed by development department and the inspection viewpoint matrix drew up by quality assurance.

1. はじめに

我々の組織では、ソフトウェア開発においてウォーターフォールで開発している場合が多い。上流工程の開発成果物に問題があるとバグを作りこみ、下流工程で問題が顕在化した場合、工程に大きな影響を与えるため、上流工程での品質向上は常に課題として挙げられている。我々の品質保証部では、品質向上のために、機能仕様書や設計仕様書のレビューを行うなど上流工程のうちから開発部とやり取りを行っている。また、開発プロセスと並行して検査施策を考え、製品検査のための準備を行っている。

今回、上流工程の品質を向上させる施策として、W モデル[1][2]の適用を検討した。W モデルとは、開発プロセスと並行してテストプロセスも進めるプロセスモデルで、上流工程のうちからテスト設計を行う。W モデルの考えの基、既存の検査プロセスの見直しを行い、上流工程の開発成果物の問題を早期に解決することを目標とし、施策の検討・実施を行った。

本論文では、2 章でこれまでの検査プロセスについて整理して問題点を挙げ、W モデル適用にあたりどうプロセスを改善するかを述べる。3 章では、問題解決の工夫点について述べ、4 章で適用し評価を行う。

1）株式会社日立製作所 情報・通信システム社 IT プラットフォーム事業本部開発統括本部 プラットフォーム QA 本部 ソフト品質保証部
Hitachi, Ltd., Information & Telecommunication Systems Company IT Platform Division Group, IT Platform R & D Management Division Platform Quality Assurance Operation Software Quality Assurance

神奈川県横浜市戸塚区吉田町 292 番地 Tel: 050-3139-8669
292, Yoshida-cho, Totsuka-ku,Yokohama Japan

2. 検査プロセスの分析

2.1 現状の検査プロセスの問題点

現状、V モデルに従い開発を行っている。既存の開発工程と品質保証工程の流れを図 2.1(W モデル適用前)に示す。開発部では、仕様書の作成、実装、単体・結合テストが主な業務である。品質保証部では、仕様書のレビュー、検査観点表の作成、検査項目の作成、製品検査が主な業務である。開発部のテストが全て終わり、品質保証部での製品検査が合格となると、最終的に製品出荷となる。

今回、検査観点表の作成のタイミングに着目した。検査観点表の作成とは、機能仕様書に基づき、検査観点表のフォーマットで定めた観点の分類に従って検査観点を抽出し、整理を行う作業である。ここで、検査観点とは、文献[3]のテスト観点と同義で、「テスト対象に対してテスト目的から設定するテストすべきことの概要」とする[4]。例えば、「コマンド」のテスト対象に対して、「タイミング」のテスト目的からは、「複数のコマンドを同時に実行しても競合しない」という検査観点が考えられる。

検査観点表作成の際に、機能仕様書の問題を検出することが少なくないが、現状の検査観点表を作成するタイミングは、開発工程を意識していないため、問題点修正の大きな手戻りが発生することがある。そこで、開発工程に合わせて検査観点表を作成し、開発部にフローアップすることで、上流工程での品質向上に繋がると考えた。

2.2 W モデル適用によるプロセス改善

W モデル適用を検討し、機能仕様書に対して早期にフィードバックを行うため、検査観点表の作成時期を基本設計がほぼ終了したタイミングとした。そして、検査観点表を開発部にフィードバックすることで、上流工程で欠陥の作りこみを防止し、品質向上を目指すこととした(図 2.1(W モデル適用後))。検査観点表作成のタイミングや品質保証部で作成する成果物の検討を行い、以下のプロセスを実施した。

1)機能仕様書作成後のレビューを一通り終えたタイミングで、検査観点表の作成を開始し、設計仕様書の作成終了までに検査観点表を作成し終える。
2)検査観点表を作成する過程で、機能仕様書の問題点を問題点リストとして整理する。例えば、問題点として、曖昧な記述、記載漏れ、検査実施困難などの問題がある。
3)機能仕様書の問題点を解決するため、検査観点表と問題点リストに基づき、開発部とレビューを実施する。

3)では、問題点リストをベースにレビューを進める。なぜなら、検査観点表は、表そのものが大きくなる場合が多く、例えば、100 観点を超えることも珍しくないため、その表を見ながらレビューするのが煩雑な作業となるからである。

さらに、検査観点表を上流工程の早い段階で作成することで、機能仕様書の問題点を早期に洗い出せるほかに、開発部のテスト項目作成にフィードフォワードを行うことができる。開発部のテストは、内部モジュールの動作に観点を置いたホワイトボックステストが大部分を占めるが、品質保証部の検査は、機能仕様書から観点を出したブラックボックステストである。開発部も十分にテストを行っているが、観点が異なるため、品質保証部の製品検査で、バグが見つかることが多い。よって、品質保証部の検査

観点を早期に開発部に提供することで、それを開発部のテスト項目に反映することが可能となり、製品の品質向上に繋がると考える。

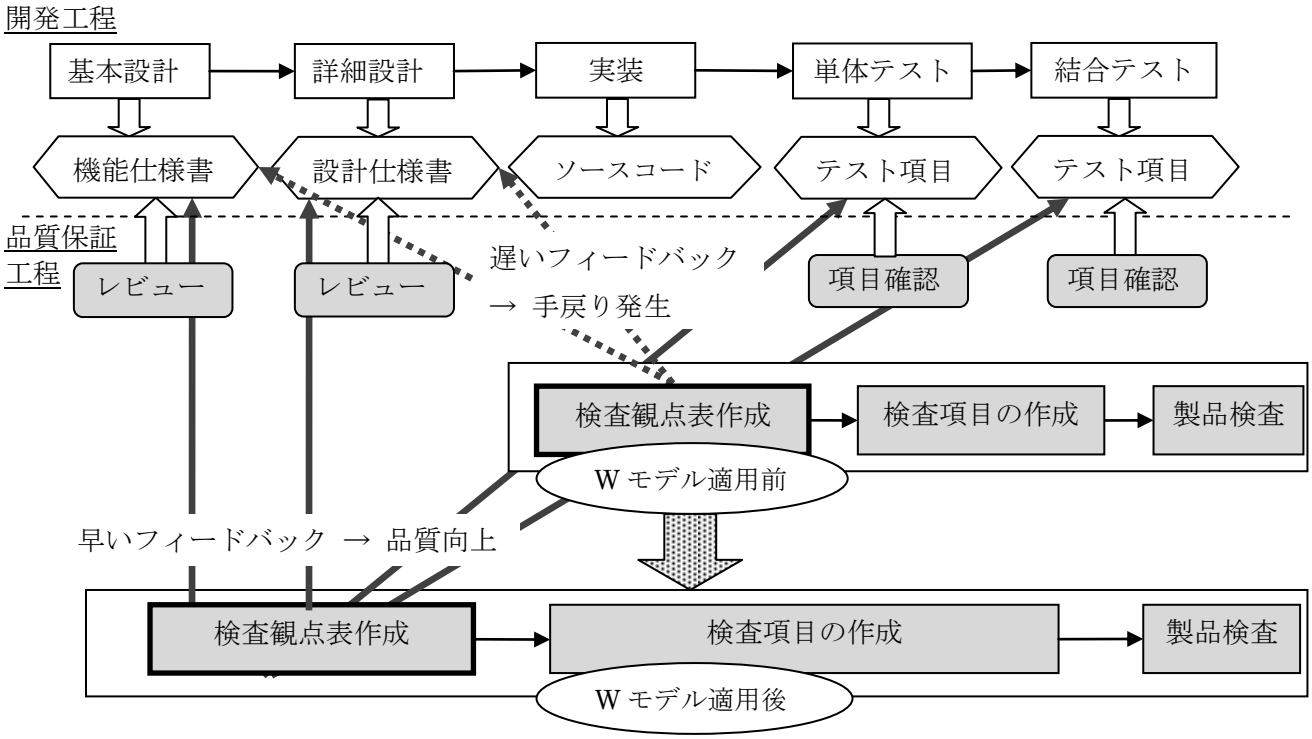

図 2.1 W モデル適用前と適用後の工程

3. 品質保証部の成果物と工夫点

3.1 検査観点表の作成

検査観点表とは、検査担当者が機能仕様書に記載された各機能に対して分析を行い、検査観点をまとめるためのものである。検査観点表の作成手順は次のとおりである。

(1)検査対象の機能について、定義、コマンド、動作などを整理し、仕様を理解する。

(2)各機能に対し、どのような観点で検査すべきか表 3.1 の分類に従いまとめる。

表 3.1 の分類は、これまでのソフトウェア開発のノウハウによって分類されたもので、検査のために意識すべきものとなっている。機能仕様書を基に、バグを作りこみそうな箇所や重要なポイントを検査観点として表 3.2 のように整理する。入力情報が機能仕様書のみでは、担当者の能力によって、検査観点の出来が大きく変わってしまうため、必要に応じて以下の情報などを活用する(表 3.3)。

(a)テスト観点知識ベース[3]を活用し、過去の事故事例を検査観点に反映させる

(b)FTA[5][6]により、重要事故についての観点を追加する

(c)製品有識者とのレビューにより製品の特徴的な観点を得る

(d)関連する他製品の検査観点表により効率的かつ効果的に観点を取り込む

表 3.1 検査観点表の観点の分類

#	分類	説明
1	入力条件	正常，異常，境界動作が規定された仕様どおりに動作するか。使いやすいインタフェースとなっているか。
2	タイミング	動作タイミングによるリソース競合が起きないか。
3	異常・特異条件	偶発的か故意かに関わらず，不正な操作やデータに対してシステムやデータを保護できるか。事故の局所化や回復性は考えられているか。
4	限界・境界・最大・最小	ループ回数，データ量，件数，コマンドの指定値などに問題はないか。
5	性能	時間効率，資源効率に問題はないか。
6	互換性	明示されたシステムと相互作用できるか。他の独立したソフトウェアと共存できるか。
7	過負荷	サーバが過負荷に陥ったときに，性能劣化や処理が実行できないなどのエラーがないか。
8	組合せ	他システムや，プログラム内の機能仕様相互の関係に問題はないか。
9	トラブルシュート	修正･改良箇所が明確か。トラブルシュート情報が十分採取できるか。操作やアウトプット情報が利用者に理解しやすいか。
10	回復処理	バックアップ・リストアや、リラン処理など、データを回復させる処理がきちんと備わっているか。
11	その他	仕様がユーザのニーズにマッチしているか。環境設定やチューニングが容易か。など。

表 3.2 検査観点の例

観点の分類	入出力条件	タイミング	異常・特異条件	～	その他
○○機能についての検査観点	･入力文字列数の境界値を確認する ･マルチバイト文字の入力はエラーとなることを確認する	･コマンド C と D の同時実行は可能か ･機能 B を実行中に機能 A を実行しても問題はないか	･SSH での通信中に、ネットワークが途切れた場合にバグは起きないか		･テストデータ生成の自動化を行う

表 3.3 検査観点表作成に用いるドキュメントの説明

対象ドキュメント	内容
テスト観点知識ベース	事故事例に基づいて最も抽象度の高いテスト観点を，より詳細化・具体化した様々なレベルのテスト観点からなるデータベース。
FTA	製品の重要事故の原因をトップダウン的に洗い出す分析手法。製品が持つリスクを把握し、製品開発力の強化や事故対応時の原因究明、テスト内容やマニュアル等のブラッシュアップを目指す。

3.2 問題点リストの作成

問題点リストとは、検査観点表を作成するときに出てきた疑問点や問題点をまとめたものである。形式は、表 3.4 のとおりである。問題点リストに記載するのは、開発部への質問や要求など何でも良い。検査項目や検査環境を考える上で必要なことは全て記載する。文字だけで書ききれない場合は、表や図を別途作成し、内容を補足する。

表 3.4 問題点リストの形式

#	確認内容	開発部回答	開発部確認	QA 確認
1	仕様書で制限している入力値以外のものを指定するとどのような動作をするのか。	テキストボックスが赤枠表示され、「実行確認」ボタンを非活性化する。	☐	☐
2	ログイン処理について、同一ユーザが別ブラウザから同時ログインすることは可能か。	可能だが、同時ログインできる上限は、関連製品に依存する。	☐	☐

開発部への質問内容を記載　　質問内容に対して開発部から回答　　確認したらチェック

3.3 開発部とのレビュー

検査観点表と問題点リストを作成したら開発部に提供し、問題点リストの確認内容について開発部とレビューを行う。機能の曖昧な部分について質問を行ったり、どのように検査を行うか相談したり、製品検査を行うにあたり問題となりそうな部分は早めに解決する。開発部からの回答を、必要に応じて検査観点表への反映などを行う。

検査観点表については、品質保証部の観点を開発部のテスト観点に反映させることが目的であるため、開発部へ提供しテスト観点に反映してもらうよう周知した。

4.　評価

4.1 施策の実施と定量的評価

施策の実施は、アプリケーションサーバの自動構築を行うソフトウェアを対象とし、勤続年数 5 年程度の検査担当者 2 名で、以下の手順で行った。

1)作成を終えた機能仕様書の機能から順次、検査観点表の作成を行った。
2)検査観点表作成の過程で浮き彫りとなった、仕様の不明確な点や、検査実施方法の課題について、開発部に確認したい項目を問題点リストにまとめた。
3)開発リーダー2 名とレビューを実施した。

施策の結果は以下の通りである(表 4.1)。

1)作成した検査観点表は､品質保証部内で観点のブラッシュアップを行い､計 136 件の観点を導出した。
2)問題点リストの問題点の数は 11 件となった。
3)問題点 11 件について質疑応答を行った。複数条件が重なった場合の処理や、他製品機能の使用有無

などを確認し、機能仕様書の記載が不足していたため 4 件の修正を行った。7 件については、機能仕様書の修正は不要であった。

さらに、品質保証部の検査観点が、開発部での作業にどのくらい反映されたのかを開発部にヒアリングしたところ、表 4.2 のようになった。136 件ある検査観点のうち 25 件が開発部のテスト項目作成に反映され、単体テストで 2 件、結合テストで 9 件の計 11 件のバグの発見に繋がった。参考に、開発部でのテスト項目数とバグの総計を表 4.3 に示す。

今回の施策により、仕様書作成工程で 4 件の機能仕様書の修正を行うことができ、テスト工程で 11 件のバグの発見に繋がったことから、施策に効果があったと考えている。

表 4.1 施策の実施に関する情報

検査観点表を作成した機能の数	5 機能
検査観点の数	136 件
問題点リストの確認内容の数	11 件
検査観点表から開発部のテスト項目に反映された検査観点の数	25 件
問題点リストのレビューにより機能仕様書の修正を行った数	4 件

表 4.2 検査観点表から反映した開発部テスト項目の件数と、発見されたバグの合計

	テスト項目数	発見されたバグの数
単体テスト	95 件	2 件
結合テスト	340 件	9 件

表 4.3 開発部で作成したテスト項目の総計と発見されたバグの総計

	テスト項目数	発見されたバグの数
単体テスト	3910 件	158 件
結合テスト	2013 件	79 件

4.2 定性的評価

開発部のリーダー3 名に、今回の施策についてアンケートを行った。その結果、以下のような回答が得られ、好評であったと言える。

・確認は数時間程度で開発部の負担はさほどない。
・バグの早期発見ができた。
・機能仕様書/チェックリストの見直しに有効だった。

品質保証部の担当者の感想は以下となった。多少手間は増えるが品質向上に効果があると言える。

・検査観点表作成後にも仕様変更が多々あるため、その度に修正を行うのが大変だった。
・W モデル適用前と比べ検査観点表の質を上げることができた。
・早期に開発部と綿密なやり取りを行うため、仕様の思い違いがなく検査を行えた。

4.3 機能仕様書の品質向上以外の効果

W モデル適用の目的として、上流工程において開発部の成果物の品質を向上させるということであったが、品質保証部の成果物に対しても効果を上げることができた。品質保証部で仕様を誤解している箇所があり、開発部からフィードバックを得られた。仕様について早い段階で開発部と意識あわせができたため、後工程での余計な確認作業やミスを減らすことができた。

4.4 W モデル実施前と実施後の工数の比較

W モデルの適用有無によって、バグ修正にかかる工数を試算して比較する。工数は、各担当者の作業時間の累計である。表 4.2 のバグ計 11 件について、W モデルを適用しなかった場合は、品質保証部の製品検査でバグが検出されるものとして考える。本取り組みの W モデル適用を実施した場合、すなわちバグが開発部のテストで摘出された場合と、適用を実施しなかった場合、すなわち品質保証部の検査で摘出された場合で、1 件あたりの修正にかかる工数の試算を表 4.4 に示す。品質保証部の製品検査で摘出された場合、開発部と品質保証部間で修正内容の打ち合せを行い、再検査を行うため工数が増える。W モデル適用有無による工数の差は表 4.5 のように試算される。よって、バグ修正にかかる工数の比較という観点に限定すると、製品検査以前にバグを多く摘出できた場合、修正に必要な工数を削減できる。

一方で、W モデル適用にあたり検査までの作業量が増え、表 4.6 に示すような工数が増加した。施策がバグの摘出に結びつかない場合は、工数が余分に増えることとなるが、バグ流出リスクを考えると、早期にテスト項目に反映し、リスク軽減を図ることが望ましい。

表 4.4 バグ修正の 1 件あたりの試算工数

#	不良発見のタイミング	品質保証部の工数	開発部の工数
1	開発部での単体テスト	0 時間	1 時間
2	開発部での結合テスト	0 時間	3 時間
3	品質保証部での製品検査	4 時間	10 時間

表 4.5 W モデル実施有無によるバグ修正にかかる工数試算の比較

	W モデル有	W モデル無
品質保証部工数	0 時間	4 時間×11 件 = 44 時間
開発部工数	単体テスト:1 時間×2 件 = 2 時間 結合テスト:3 時間×9 件 = 27 時間	10 時間×11 件 = 110 時間
合計	29 時間	154 時間

表 4.6 W モデル適用による実工数の増加

#	作業内容	対象	工数
1	仕様変更に伴う検査観点表の修正	品質保証部	3 時間
2	検査観点表の内容が開発部のテスト項目に反映されたかを確認	品質保証部	5 時間
3	確認表の回答とその調査、検査観点表の確認	開発部	5 時間
4	検査観点表をテスト項目へ反映	開発部	10 時間
5	追加したテスト項目の実施	開発部	20 時間

5. まとめ

今回、ソフトウェア開発における品質保証部の業務に W モデルを適用した。これまで、検査観点表の活用については、各検査担当者にとどまっていることがほとんどであったが、上流工程の早い段階で作成して開発部に提供することで、製品の品質向上に繋げることができた。

今後の課題として、以下について検討を行う必要があると考えている。

・設計仕様書に対して、評価を行い開発部にフィードバックする方法

・開発側のテスト項目について、漏れがないことを効率良くチェックする方法

今後も、今回の施策を引き続き適用し、どのように工夫すれば開発部側でバグを取りきることができるのか考え、検査観点の作成やその他有効なアプローチを検討し、実施していく。

参考文献

[1] 秋山 浩一 他, W モデルとは何か, JaSST 2012 Tokyo

[2] 秋山 浩一 他, W モデル導入の手引き, JaSST 2013 Tokyo

[3] 光永 洋, 故障事例によるテスト観点知識ベース構築とテスト設計への適用, SQiP2012

[4] 吉澤 智美, テストアーキテクチャ解説～テストアーキテクチャ設計を実践するには～, JaSST2012

[5] 高山 啓, ソフトウェア品質開発における FTA による信頼性リスク分析, SQiP2010

[6] 塩見 弘, FMEA、FTA の活用（日科技連信頼性工学シリーズ（7)), 日科技連（1983/01)