グレーボックステストによる効果的な品質確保の取り組み
－無駄なテストを排除し、影響範囲に焦点をあてた、テスト設計技法の紹介－

田中　桂三

オムロン株式会社　オートメーション統轄事業部　ソフトウェア開発部　第 2 開発課

〒525-0035 滋賀県草津市西草津 2 丁目 2-1
E-mail: keizo_tanaka@omron.co.jp

あらまし　テスト期間の短縮の為に「如何に無駄なテストを排除し、かつ不具合を短期間で発見するか？」に着目し、解決手段としてグレーボックステストを導入した。本論文では UML を活用したグレーボックステスト設計時の課題と対策、および効果と今後の展開について紹介する。

キーワード　品質確保、グレーボックステスト、テスト設計、UML、コミュニケーション図、アクティビティー図、データフロー図、W モデル開発

Quality Assurance activity by Grey-box testing
－Aim at prevention of outflow of defects to market－

Keizo Tanaka

OMRON Corporation　Industrial Automation Business Company

Automation Systems Division HQ, Development Center

Software Development Dept. Group 2

2-2-1, Nishi-Kusatsu, Kusatsu-City, Shiga-Prefecture, 525-0035 Japan
E-mail: keizo_tanaka@omron.co.jp

Abstract　In order to shorten the testing term, we considered how we remove useless test cases and find defects in short term, then we introduced Grey-box testing concept to solve this issue by referring to UML. I report issues, the solutions, and the efficiency after the solutions by introduction of Grey-box testing in this paper.　Also I report future development of Grey-box testing in this paper.

Keyword Quality Assurance, Grey-box test, UML, Communication diagram, Activity diagram, Dataflow diagram, W-Model

1. まえがき

お客様のご要望に迅速に対応する為に、短期間でのソフトウェア商品開発が求められている。それには短期間でのテスト実施が必要である。テストに期間がかかる理由について従来の商品開発を分析した結果、「内部変更に影響のない機能も含め、全機能を網羅したテストを実施しており、多大なテスト工数がかかる」および「テストの中盤～後半に複数条件で発生する不具合の発見数が多くなり、不具合収束の為にテスト期間がかかる」が問題であることが分かった。これらを解決する為に、「如何に無駄なテストを排除し、かつ複数条件で発生する不具合を短期間で発見するか？」に着目し、具体的手段として、内部構造を理解した上で外部動作をテストする「グレーボックステスト」を導入した。効果達成の目標値として、最も迅速なご要望の対応を求められる派生型開発プロジェクトにおいて、「不具合を収束させた上で、テスト期間を 50%短縮」を設定した。本論文では、グレーボックステスト導入時の課題と対策、および目標値に対する実績、および今後の展開について紹介する。

2. 課題の設定と解決方針

2.1. 課題の設定

「如何にテスト期間を短縮化すべきか？」

この解決策を導き出す為、テストに期間がかかる原因について従来の商品開発を分析した。その結果、以下の 2 点が問題であることがわかった。

問題 1) 全機能を網羅したテストの実施により多大なテスト工数が必要であり、テストに期間がかかった。

問題 2) テストの中盤～後半に、複数条件で発生する不具合の発見数が多くなり、不具合収束の為にテスト期間がかかっていた。

これらの問題を深堀した結果、以下の原因が判明した。

問題 1)の原因：内部変更による各機能への影響有無を判断できないので、全機能を網羅したテストケースを実施していた。

問題 2)の原因：複数条件で発生する不具合を発見するには、外部仕様書だけでは情報が足りず、実動作状況や上流設計者へのヒアリングを基に、テスト途中に探索型テスト観点を追加していた。これがテストの中盤～後半での発見不具合数の増大につながっていた。

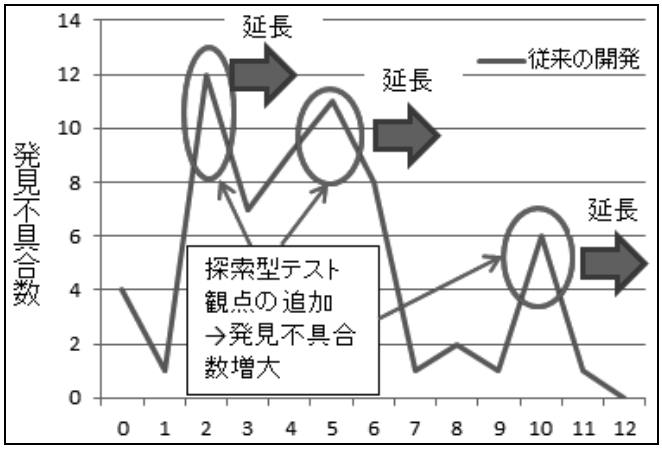

図 1. 従来開発でのテスト期間と発見不具合数

上記問題の解決に向け、2 つの課題を設定した。

課題 1)内部変更による各機能への影響有無を特定する。影響のない機能は無駄テストとして削除し、影響のある機能に焦点をあてたテストを実施する。

課題 2)探索型テストで発見していた、複数条件で発生する不具合を、短期間で発見する。

2.2. 解決方針

上記 2 課題を解決する為に、派生型開発での内部変更による各機能への影響有無を、テスト開始迄に特定する必要がある。そこで内部構造を理解した上で外部動作をテストする「グレーボックステスト」をテスト設計時に導入した。その際 3 つの解決方針を設定した。

・解決方針 1)漏れなく内部変更情報を集めて、各機能への影響有無を特定する。

・解決方針 2) 1)で「影響あり」と判断した機能の中で、漏れなく、影響を及ぼす処理（操作）を特定する。

・解決方針 3) 1)で「影響あり」と判断した機能の中で、漏れなく、影響を及ぼすデータと更新タイミングを特定する。

解決方針 1)は課題 1)の解決策として設定した。一方解決方針 2)と 3)は、課題 2)で挙げた、複数条件で発生する不具合の混入原因を分析し、これらを解決する対策として設定した。

混入原因 A) 異常、キャンセルケースの考慮漏れ

・外部機器と通信できない等の異常ケース

・操作を途中でキャンセルしたケース

⇒対策として、解決方針 2)を設定

混入原因 B) データ更新の考慮漏れ

・別操作によるデータの更新有無の考慮漏れ

・複数コンポーネント間の排他処理、タイミングのずれによるデータの更新漏れ

⇒対策として、解決方針 3)を設定

3. グレーボックステストの実施内容

3.1. UML をインプットとしたテスト設計

1) 内部変更により影響を及ぼす機能の特定方法

解決方針 1)「漏れなく内部変更情報を集めて、各機能への影響有無を特定する。」 このインプット情報として、コミュニケーション図を活用した。コミュニケーション図は、複数のオブジェクトが連携して目的を達成する振る舞いをモデリングした図である。ところが本図の参照中に以下の問題が発生した。

問題 1) コミュニケーション図上で、変更箇所が記載されているものの、変更理由が理解できない。

問題 2)コミュニケーション図上で、オブジェクト名とインターフェース名（メソッド名）のみが記載されており、内部構造を知らないテスタが理解できない。

問題 3)コミュニケーション図に記載漏れがあると、テストケースが漏れてしまう。

問題 4)複数の上流設計者から収集したコミュニケーション図から各機能への影響有無を特定する際に、テスタの誤り（仕様の誤解、反映漏れ）の有無が判断できない。

上記 4 点の問題に対し、以下の対策をとった。

対策 1)要件を実現する為の、内部変更方針（実現手段）を数行にまとめて、コミュニケーション図に追記した。これによりテスタが変更概要を把握し、変更内容の理解を促進した。

対策 2)コミュニケーション図に、外部仕様の用語（機能名、データ名、処理名）を用いた注釈を追記した。

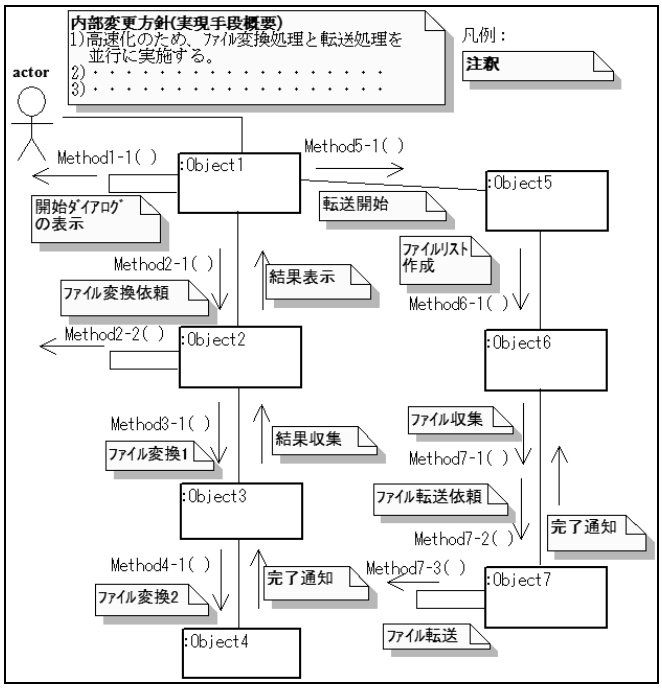

図 2：コミュニケーション図の例

対策 3)テスト対象のビルドのソースコード変更履歴を参照し、コミュニケーション図と比較することで記載漏れ有無を確認した。

対策 4) テスタがコミュニケーション図から内部変更による各機能への影響有無を理解し、「要件-機能マトリクス」を作成した。また上流設計者を集めてウォークスルーレビューを実施し、テスタの誤りの有無を確認した。さらに、確実に品質確保をした上で効率良くテストを実施するために、内部変更による影響度合いを以下の 5 レベルに細分化し、レベル毎の検証方法を定義した。

＜レベルの分類と検証方法＞

●、○（※）：要件対応により、対象機能の内部変更あり。

⇒担当の上流設計チームがホワイトボックステスト、テスタがグレーボックステストで検証。

▲、△（※）：要件対応により、対象機能の内部変更はないが、外部仕様に影響がある。

⇒テスタがグレーボックステストで実施

空白：要件対応により、内部変更がなく、外部仕様にも影響がない。

⇒無駄なテストとして実施対象外

（※）●、▲：該当要件に対し重要度の高い不具合が発生する機能

○と△：重要度の高い不具合が発生しない機能

●と○、▲と△の分類手段として、ソフトウェアテストシンポジウム 2012Tokyo で経験論文として発表した「信頼性分析による品質確保の取り組み」の手法を活用した。●と▲については、上記検証方法に加え、本手法の実践により市場への不具合流出を抑えた。

機能 / 要件	大機能1			大機能2			大機能3		
	小機能1-1	小機能1-2	.....	小機能2-1	小機能2-2	.....	小機能3-1	小機能3-2	.....
要件1	○	○		●	○		△	▲	
要件2	○	○		▲	△				
要件3	○	○		●	○		○	●	
要件4	○	○		▲	△		○	▲	
要件5	○	○					△	▲	
・									
要件16	▲	▲		●	●				
要件17	○	○		▲	○				
要件18				▲	△		○	●	
要件19	○	○		●	△		△	▲	
要件20				●	△		△	▲	

図 3：要件-機能マトリクスの例

なお、対策 1)2)3)は上流設計者間の構造レビュー等でも有効な情報である為、上流設計者がマスタデータとして直接追記した。対策 4)はテストのインプット情報としてのみ必要である為、テスタがテスト設計ドキュメントとして作成した。

2) 内部変更により影響を及ぼす処理の特定方法

解決方針 2)「解決方針 1)で「影響あり」と判断した機能の中で、漏れなく、影響を及ぼす処理（操作）を特定する。」この入力情報として、アクティビティー図を活用した。アクティビティー図は、UML に定められたダイアグラムの一つで、一連の処理を構成する動作（action）に着目し、その実行順序や条件、制御などの依存関係を示した図である。システムの振る舞いやワークフローなどを表現できる利点がある。ところが、2.2 章の混入原因 A)の通り、異常ケース、キャンセルケースが記載されていなかった。よって以下の対策をとった。

対策 1) 途中でキャンセルした処理を、アクティビティー図に追記した。

対策 2)異常処理は各フローで発生するので､本図に記載すると分岐数が膨大になり、図が複雑になって見づらくなる。また、異常処理について、大部分が各フローに関わらず、共通仕様で設計されている。よってフロー図では表現せず､注釈として追記した。

なお､上記 2 対策は上流設計者間の構造レビュー等でも有効な情報である為、上流設計者がマスタデータとして直接追記した。

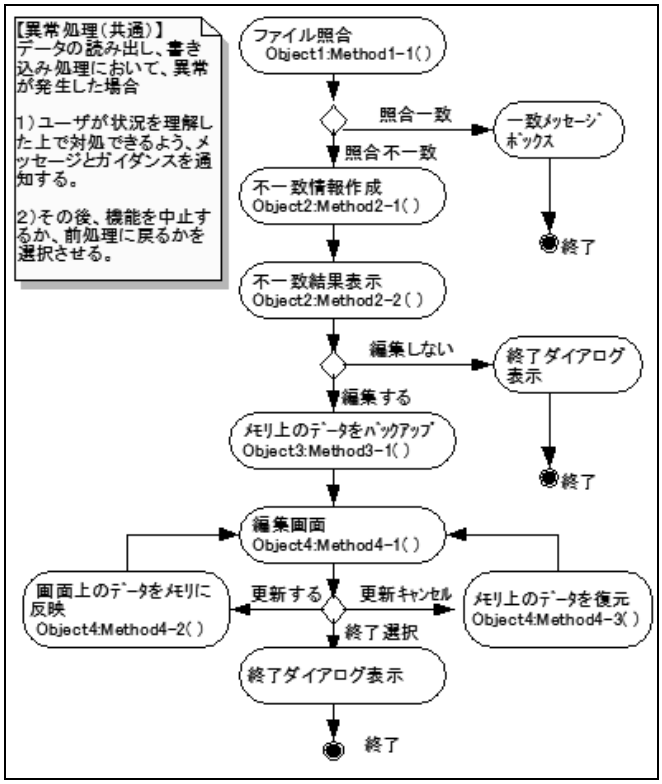

図 4：アクティビティー図の例

3)　内部変更により影響を及ぼすデータの特定方法

解決方針 3)「解決方針 1)で「影響あり」と判断した機能の中で、漏れなく、影響を及ぼすデータと更新タイミングを特定する。」この入力情報として、1)と同様にコミュニケーション図を活用した。また 2.2 章の混入原因 B)に挙げた、データの更新状態の考慮漏れも、注釈追記によりコミュニケーション図上で表現した。ところがテスト設計中に以下の問題が判明した。

問題)コミュニケーション図は、オブジェクト間と機能間の振る舞いが中心であり、データに着目した形で記載していない。一方テスト対象のソフトウェアは、各機能の実行中に、頻繁にデータ変換が行われる。コミュニケーション図では、データのライフサイクルが表現しにくく、「どのタイミングでどのデータに影響を及ぼすのか？」が理解できない。

この為、以下の対策をとった。

対策)テスト設計ドキュメントとして、データの流れを中心に表現できる、データフロー図を作成した。また本図の作成後に、上流設計者を集めてデータフロー図のウォークスルーレビューを実施し、テスタの誤解、影響箇所特定の漏れがないかを確認した。

なお、本取り組みの中で、データフロー図はテストのインプット情報としてのみ必要である為、テスタが設計ドキュメントとして作成した。

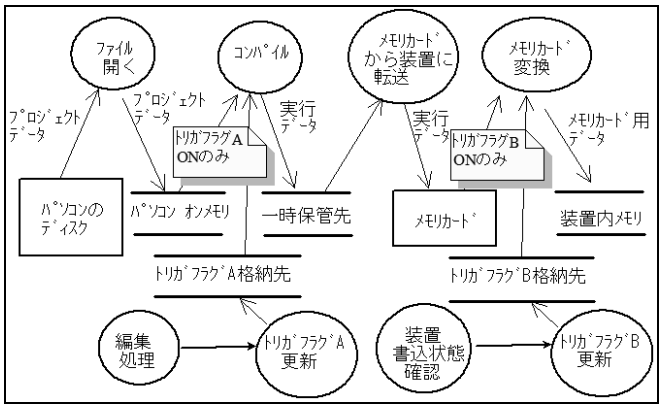

図 5：データフロー図の例

3.2. 無駄テストの削除

3.1.1)で、内部変更による影響がないと判別した機能（要件-機能マトリクスで空白の箇所）は「無駄なテスト」と判断し、テスト実施対象外とした。

一方で、対象外の範囲も構成管理上の問題で不具合が流出するリスクがある為、万一に備えてテスト自動化による回帰テストを実施することで本リスクを回避した。なお、ソフトウェア品質シンポジウム　2012 で経験論文として発表した「テスト種類に着目した最適な自動テスト支援ツールの選定方法と実践」の手法により、テスト対象の全ケースを自動化した。

3.3. 探索型テストでの発見不具合の早期発見

探索型テストで発見する複数条件起因の不具合を短期間で発見する為、3.1.2)と 3.1.3)の情報を元にテスト設計を行った。

・3.1.2)のアクティビティー図から、ワークフローの条件分岐、ループ処理、キャンセル処理、異常処理の条件を網羅したテストを実施した。

・テスト対象のソフトウェアは、「実ユーザが認識する画面上のデータ」⇒「メモリ上やテンポラリファイルに保存される中間データ」⇒「プロジェクトファイルやターゲット装置に転送される最終データ」の流れでデータ変換処理が実行される。3.1.3)のデータフロー図上で、本変換処理のタイミングに使う、トリガフラグの更新に着目した状態遷移を図式化した。本情報を元に、データ変換のトリガとなる操作、コンポーネント間の排他処理とタイミングを網羅したテストを実施した。

4. グレーボックステスト導入効果

グレーボックステスト導入と 3 章に挙げた取り組みを実施した結果、以下の 3 点の効果が得られ、テスト期間を従来開発の 12 週間から 5 週間に短縮した。効果達成の目標値として挙げた、「不具合を収束させた上で、テスト期間を 50%短縮」に対し、60%の短縮を達成した。

1)無駄テストの削除により、従来開発と比べ、テス

トケース数とテスト期間を 20%削減した。

2)テスト開始迄に探索型テストの観点をテストケースに落とすことで、テスト開始直後から複数条件で発生する不具合を多数発見し、また途中での探索型テストによるテスト期間の延長を防止した。これによりテスト期間を 30%削減した。

3) UML の各図をレビューし改善することで、副次効果として上流工程での品質が向上し、テスト工程での不具合数が 30%削減された。これにより修正確認時間が短縮し、テスト期間を 10%削減した。目標を上回った理由は、本効果によるものである。

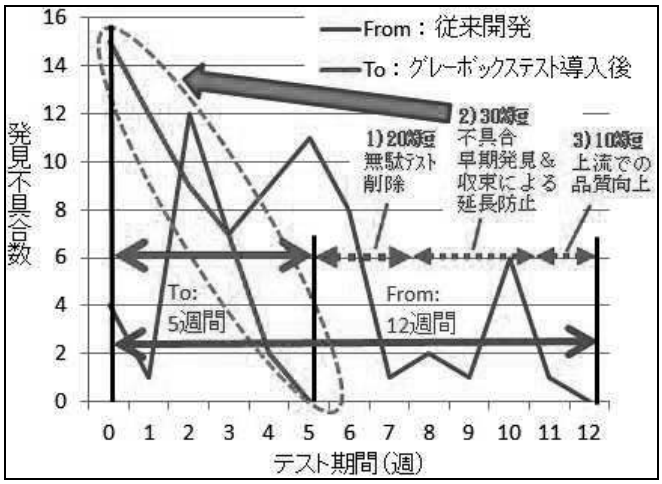

図 6:テスト期間−発見不具合数の変化

5. 今後の展開

5.1. グレーボックステストに最適なインプット情報の検証

グレーボックステストに最適なインプット情報として、コミュニケーション図とアクティビティー図を活用したが、グレーボックステストのインプット情報として十分ではなく、テスタがテスト設計ドキュメントとして、要件-機能マトリクスとデータフロー図を作成した。これによりテスト設計工数が 2 人月増加した。

今後継続してグレーボックステストを実践する中で、他の UML や構造設計ドキュメントを調査し、「テスト設計工数を増やさずに、設計・テスト両方で活用できる、最適な設計ドキュメントは何か？」を検討する。

5.2. 上流工程での品質確保の取り組み

UML の各図をレビューし改善することで、上流工程での品質が向上し、テスト工程での不具合数が 30%削減されたものの、2.2.章に記載した混入原因 A)と混入原因 B)に起因する不具合が、依然テスト工程で数多く発見されている。

一般に上流工程で品質確保すればより品質が向上すると言われている。本取り組みでは、グレーボックスのテスト設計が構造設計工程で完了できず、実装・デバッグ工程と並行して実施した。

さらなる上流工程での品質確保の為、今後はグレーボックスの観点をより早期の段階で実施し、テスト設計が構造設計工程内で完了できるように進める。具体的手段として、テスト設計を上流工程から開始し上流設計プロセスとテストプロセスを並行して進める「W モデル」を導入し、「構造設計、レビュー、構造設計ドキュメントへのレビュー反映、テスト設計」の一連の作業を構造設計工程期間内で実施可能となるよう、開発プロセスの改善を進める。

文　献

[1] 西康晴氏, ソフトウェアテスト技術, pp.55-101, 経営情報学会, 第三回情報システム工学研究部会, 東京､2007.

[2] André C. Coulter, Graybox Software Testing in Real-Time in the Real World Ver.08, Lockheed Martin Missiles and Fire Control – Orlando, 2001

[3] Jean Hartmann, Claudio Imoberdorf, Michael Meisinger, UML-Based Integration Testing, pp.2-11,U.S, 2003

[4] 田中桂三, 信頼性分析による品質確保の取り組み、pp1-3,ソフトウェアテストシンポジウム 2012 東京,東京,2012

[5] 田中桂三, テスト種類に着目した最適な自動テスト支援ツールの選定方法と実践、pp1-8, ソフトウェア品質シンポジウム 2012, 東京,2012