Vol.2013-SE-179 No.18
2013/3/11

結合テストにおけるテスト設計自動化に関する取り組み －実プロジェクトで見えた効果と課題－

田代裕和[†1]　町田欣史[†1]　染谷雄介[†1]　飯塚裕一[†1]

システム開発において，結合テストのテスト設計における工数削減，品質向上が望まれている．本論文では，結合テスト設計における工数削減と品質向上を目的とした「結合テスト設計自動化」に関する取り組みを説明する．さらに，開発した自動化ツールを実際のシステム開発プロジェクトで適用した効果（工数削減，テストケース品質向上など）と課題について報告する．

Test Design Automation in Integration Test

HIROKAZU TASHIRO[†1]　YOSHINOBU MACHIDA[†1]
YUSUKE SOMEYA[†1]　YUICHI IIZUKA[†1]

Productivity improvement and quality improvement in integration test phase is desired in a lot of system development projects. In this document, the integration test design automation tool which is designed to improve productivity and quality is explained. Furthermore, the effect and future challenge are explained.

1. はじめに

システム開発における結合テストの工数や工期は開発全体の約 3 割を占めるという調査結果があり[1]，結合テストの工数・工期短縮はシステム開発全体の工数・工期短縮に大きく寄与する．この結合テストの一般的なプロセスは，以下に示す流れである[2].

1. テスト計画作業とコントロール
2. テストの分析と設計
3. テストの実装と実行
4. 終了基準の評価とレポート
5. 終了作業

この「2. テストの分析と設計」においては，テストのアプローチ・着眼点である「テスト観点」や「要件・設計内容」などに基づき，入力値・実行事前条件・期待結果・実行事後条件のセットである「テストケース」を作成する[2]が，現在のシステム開発現場ではテストケースの作成で「抜け・漏れなくテストケースを洗い出すことが困難」という課題があり，大きな負担と感じられている[3].

こうした課題への対応策の 1 つとして，単純作業や繰り返し作業の効率化・精度向上を実現する自動化ツールの利用がある[3]．しかし,「テスト設計（テストケース作成）」の自動化を実施している開発者の割合は，結合テストで自動化を行っている開発者の中でも約 3 割と決して高くないという調査結果がある[4]．テスト設計を自動化するためには，入力情報となる設計情報の解析自動化が必要である．

そこで,設計情報をモデル化しテストケースを作成する「モデルベースドテスト」の研究開発を行っている NTT サービスイノベーション総合研究所[5][6]と連携して，テスト設計の自動化に取り組んだ．本論文では，この取り組みについて説明するとともに，実プロジェクトで適用した結果について効果と課題を報告する．

2. 取り組み紹介

2.1 テスト設計自動化に関する取り組み概要

我々は，設計書を基にテストケース表を自動生成するツール（以下，本ツールとする）を開発し，システム開発プロジェクトでの活用を推進している．結合テスト設計自動化の第一段階として，Web アプリケーションを構成する個々の処理（一画面遷移）の振る舞い（ロジック）を網羅的にテストする結合テストのテスト設計自動化に取り組んだ.

Web アプリケーション開発における既存の取り組みでは画面遷移をモデル化し，テストケースを網羅的に自動生成するというツール[7]は存在するが，筆者らはより細かい粒度である一処理（一画面遷移）の設計を網羅的にテストするため，処理ロジックを UML アクティビティ図にモデル化し，一処理内のパターン（シナリオ）を網羅的にテストケース表へ自動生成することに取り組んだ.

本ツールでは，処理ロジックの網羅的なテストを実施する際の，通常のテスト設計プロセスにおいて設計書に記載された処理ロジックをもとにテストケース表を作成する作業を自動化する．したがって一処理（一画面遷移）の設計情報が記述される画面設計書と処理設計書が本ツールの入力となる（図 1）.

†1 NTT データ
NTT DATA

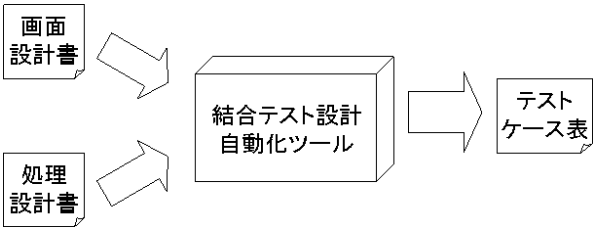

図 1 結合テスト設計自動化ツールの概要

また本ツール内部のアルゴリズム詳細は，文献[6]を参照されたい．

2.2 入力される設計書

本ツールの入力となる設計書は，以下 2 つである．

1. 画面設計書
2. 処理設計書

画面設計書は，各画面内の表示文字列や入力フィールドに関する情報，入力チェックに関する情報（入力フィールドの制約事項やチェックエラー時の処理情報）などが記述されるが，そのうち入力チェックに関する情報（図 2）を自動生成に用いる．本ツールの利用者には，本ツールがテストケース自動生成のために画面設計書を解析する必要があるため，以下の記述ルールに従うことを求めた．

- Microsoft® Excel®の xls 形式を利用
- 社内標準として規定された表形式で記述

入力チェック対象	チェックルール	エラー時のメッセージキー	処理	
			PR001	PR002
テキストボックスA	必須	required	○	
テキストボックスB	最大文字数	max-length		○

図 2 画面設計書の記載内容（入力チェック情報）

処理設計書には，各処理内（一画面遷移内）のロジック詳細（ロジック順，分岐条件，DB アクセス情報など）や処理後の状態（遷移先画面，出力メッセージなど）などが記述される．本ツールの利用者には，本ツールがテストケース自動生成のために処理設計情報を解析する必要がるため，以下の記述ルールに従うことを求めた．

- Microsoft® Visio® 2007 の vdx 形式を利用
- UML アクティビティ図で記述
- Microsoft® Visio® 2007 付属の UML ステンシルを利用

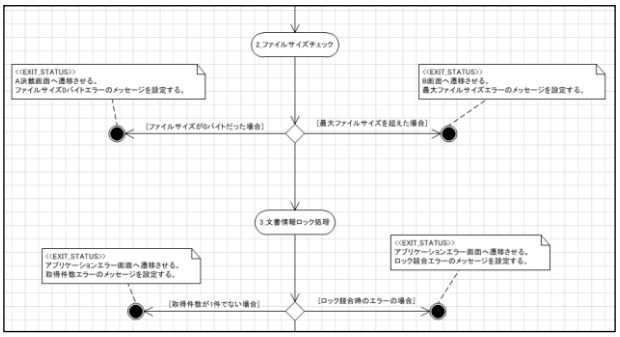

図 3 処理設計書

2.3 出力されるテストケース表とテスト設計プロセス

本ツールを通して自動生成されるテストケース表では，入力設計書から解析・抽出された全経路（ループが存在する場合はループを通らない経路と 1 回通る経路）がテストシナリオとして網羅的に出力される[6]．具体的に出力されるテストシナリオは，以下 2 点である．

1. 入力チェックに関するシナリオ
   （画面設計書に設計される内容）
2. 処理ロジックに関するシナリオ
   （処理設計書に設計される内容）

これらのシナリオ以外のテストシナリオ（ループを 2 回通る経路など）は，プロジェクトのテスト方針に基づき，手動での追加が必要である．また，各テストにおける具体的な入力値や期待結果は自動生成されない．

よって，本ツールを利用する結合テスト設計プロセスは，以下の通りとなる．

1. テストシナリオの自動生成
   2.2 節で示した各設計書を本ツールに入力し，テストシナリオを網羅的に出力したテストケース表を本ツールによって出力（自動生成）する
2. テストシナリオの追加，取捨選択
   各プロジェクトにおけるテスト方針に基づき，必要なテストシナリオをテストケース表に手動で追加，もしくは取捨選択を行う
3. テストケース情報の追記
   1，2 を通して作成したテストケース表に対して，テストシナリオを参考にテストケース情報（入力値や期待結果など）を手動で追記する

3. 実プロジェクトにおける効果検証

3.1 プロジェクト概要

本ツールを表 1 に示すプロジェクトに適用し，本ツールの導入効果を検証した．

表 1 適用プロジェクトの概要

システム概要	物流・販売管理システム
開発区分	新規開発

3.2 検証方法

対象のプロジェクトにおいて，典型的な 2 つの処理（以下，処理 A, 処理 B とする.）に対してツールの利用有無による工数および品質を比較する対照実験を行った.

具体的な検証項目は以下の通りとした.

1. 工数削減効果
2. 品質（テストケースの品質）向上効果
   本ツール利用による, テストケースの作成漏れの回避に関する効果
3. ツール利用の前提条件に関する評価
   本ツール利用のために追加された設計書記述ルールの設計アクティビティへの影響など
4. テスト設計プロセスの評価
   本ツール利用時のテスト設計プロセスに関する課題など

3.3 検証結果 1 ： 工数削減効果

工数に関する検証結果を表 2 に示す.

表 2 工数（時間）

処理	利用有無	テストシナリオ作成 [*1]	テストケース作成 [*2]	テスト設計合計
A	無し	1 時間 30 分	1 時間 40 分	3 時間 10 分
	有り	35 分	1 時間 10 分	1 時間 45 分
B	無し	6 時間	4 時間	10 時間
	有り	10 分	2 時間	2 時間 10 分

[*1]テストシナリオの自動生成，追加，取捨選択に要した工数（レビュー含む）

[*2]テストケース情報の追記に要した工数（レビュー含む）

本ツールを利用した場合，テスト設計全体の工数が処理 A で約 45%, 処理 B で約 78%削減されたことが確認できた. テスト設計のなかで本ツールが自動化する「テストシナリオ作成」については，処理 A で約 61%，処理 B で約 97%削減されており，本ツールによる自動化がテスト設計工数全体の工数を削減させていることが確認できた．また，テストケース作成については，処理 A，B ともに本ツールを利用した場合の方がテストケース数が多い（3.4 節参照）にもかかわらず，工数が削減されている．これは，本ツールが生成するテストシナリオ情報がテストケース作成に参考になったためだと考えられる.

また，他の本ツール適用プロジェクトに対して工数に関するヒアリングを実施したところ，レビューの生産性向上という効果もあったと報告を受けている．具体的には，設計書から機械的に（網羅的に）テストシナリオが生成されるので，レビューにおいても網羅観点のチェックが省略でき，工数削減につながったとのことである.

3.4 検証結果 2 ： 品質向上効果

品質（テストケース数）に関する検証結果を表 3 に示す.

表 3 テストケース数比較

処理	テストケース種別	テストケース数	
		ツール利用無し	ツール利用有り
A	入力チェック	6	7
	処理ロジック	3	6
	合計	9	13
B	入力チェック	6	16
	処理ロジック	9	27
	合計	15	43

本ツールを利用した場合，テストケース数が処理 A で約 44%，処理 B で約 187%増加している．これらのテストケースを全て分析した結果，全てのテストケースが本プロジェクトで必要なテストケースであることが確認できた．つまり，本ツールの利用により，設計書からの転記漏れ・転記ミスの回避やテスト設計者のスキル不足を補うことが可能になり，テストケース表の品質の向上につながったと考えられる.

また，他の本ツール適用プロジェクトに対して品質に関するヒアリングを実施したところ，以下に記す効果があったと報告を受けている.

1. 設計書品質の向上
   自動化ツールの記述ルールに従うことで設計書の記述が厳格化されるため，設計時の属人性の排除，記述漏れの回避につながった
2. 設計レビューの精度向上
   設計時に UML アクティビティ図の作成が必須となり，自由記述（自然言語）による設計のみの場合と比較して，設計レビューが容易になり，かつ，精度が上がった

3.5 検証結果 3 ： ツール利用の前提条件に関する評価

画面設計書については, 2.2 節に示す通り社内標準を利用するため，本ツール導入に伴う記述ルールに関する問題課題は発生しなかった.

処理設計書については，本ツールに起因する記述ルールに慣れるまで，特別な時間を要した．ただ，この課題に関しては，2 回目（2 処理目）以降は解決された．よって，自動化対象が多い場合，規定の記述ルールに慣れた設計者が利用する場合は，プロジェクト全体での工数削減効果がより大きくなると考えられる.

3.6 検証結果 4 ： テスト設計プロセスの評価

テスト設計プロセスについては，従来のテスト設計プロセス（社内の規定プロセス）に準拠していたため，本ツール導入に伴う問題課題は発生しなかった.

4. 結論 ～まとめと今後の課題～

結合テストのテスト設計における生産性および品質に関する問題を解決するために，テスト設計の自動化に取り組んだ．実際のプロジェクトに適用した結果，生産性・品質ともに向上が見られ，テスト設計の自動化の有効性が確認できた.

今後は以下の課題に対し，プロジェクト適用・検証・フィードバック収集を継続することで改良に取り組む予定である.

- 一処理（一画面遷移）内の結合テストにおける，今回対象外であったテストのテスト設計自動化（画面レイアウトなど）
- 一処理（一画面遷移）内の結合テストにおける，テストケース表内の自動出力部分の拡張（入力値や期待結果などの自動生成）
- より後続の結合テスト（画面遷移を複数含む単位での結合テスト）におけるテスト設計の自動化
- UML アクティビティ図からの経路抽出技法の拡充
- テスト実行の自動化ツールとの連携

1 日本情報システム・ユーザー協会:”ソフトウェア開発管理基準に関する調査報告書. （ソフトウェアメトリックス調査）”，2012 年 2 月
2 JSTQB:”ISTQB テスト技術者資格制度 Foundation Level シラバス日本語版 Version 2011.J02”, 2012 年 6 月
3 池上俊也:”テストツール 知られざる便利機能”，ITpro，2012 年 7 月
4 松山貴之:”開発支援ツール徹底調査 2011 テスト編 -ツールの満足度-“，ITpro，2011 年 6 月
5 Xiaojing Zhang, Haruto Tanno, Takashi Hoshino: "Introducing Test Case Derivation Techniques into Traditional Software Development: Obstacles and Potentialities", Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops.(ICSTW 2011)
6 丹野 治門, 張 暁晶, 星野 隆: "結合テストにおけるテスト項目自動生成手法の提案と評価", 電子情報通信学会技術研究報, vol. 110, no. 227
7 Xupper, http://www.xupper.com/