第 5 分科会(テスト設計グループ)

派生開発における影響箇所の把握改善によるテスト範囲の特定方法の提案

<Proposal for improvement of extraction method of influence part in derivation development>

主査　秋山　浩一（富士ゼロックス株式会社）
副主査　奥村　有紀子、堀田　文明（有限会社デバッグ工学研究所）
リーダ　阿部　啓（TIS 株式会社）
長田　倫明（株式会社セゾン情報システムズ）
鈴木　淳一（矢崎総業株式会社）
鈴木　昭吾（株式会社日立ハイテクソリューションズ）

概要

日本情報システム・ユーザー協会（JUAS）の企業 IT 動向調査の 2009 年度版[1]では、システム開発の 58%は保守運用であると予測されている。

ところが、研究員の現場では、システム開発やソフトウェアテストの標準化の取り組みは新規開発が中心であり、派生開発（過去のコードを流用して製品を開発すること）については、体系的な方法論が整備されていなかった。

本研究では、派生開発におけるソフトウェアテストの改善をテーマとして取り組んだ。開発要件による影響度の可視化を狙いとし、そこからテスト範囲を特定する方法を考案し、効果を測定した。

Abstract

It is forecast that 58% of the system development is maintenance operation in enterprise IT trend survey of Japan User Association of Information (JUAS) in the 2009 ' s version.

However, the approach of the standardization of the system development and the software test was a center the novel exploitation, and was not maintained a systematic methodology about the derivation development (Develop software by misappropriating an existing source code) on researcher's site.

In this research, it worked as a theme the improvement of the software test in the derivation development. It is assumed to be an aim to make the influence level when software is changed by the derivation development visible, and proposes the method of selecting the range of test.

1. 背景

今日のシステム開発は、新規にシステムを開発する「新規開発」より、過去のコードを流用して製品を開発する「派生開発」が多く、ソフトウェア開発全体では 9 割が派生開発によるものとされている[2]。

「派生開発」は「新規開発」に比べ、短納期・低コストの開発が要求され、他人が書いたコードを読み取る必要もあるため「全体を理解」できる状況ではなく、思い込みと勘違いが混入する「部分理解」の罠に陥りやすい特徴を持っている。

清水吉男氏が提唱した「XDDP(eXtreme Derivative Development Process：派生開発プロセス)」により、派生開発に特化した開発プロセスが整備され、派生開発時の設計・実装の問題は改善されてきている。しかし、出荷後のシステム障害は依然として発生しており、開発プロセスの変化に伴う、テストプロセスの見直しが必要であると考えた。

2. 問題

派生開発におけるテストプロセスでは、変更箇所とそれによる影響箇所が持つ関係を把握し、テスト範囲を特定する必要がある。本研究で考える影響とは、「変更・追加部分を含む機能」（以下、変更箇所）とそれ以外との関係性であり、その種類を以下に列挙する。

・親子関係(変更箇所が呼び出し元となりそれ以外の箇所を呼び出す関係)
・順序関係(変更箇所の後続でそれ以外の箇所の処理がされる関係)
・競合関係(変更箇所とそれ以外の箇所とで資源が競合する関係)

実際に、我々の経験においては、影響箇所が的確に把握できていないことが原因で、システムテストや受け入れテストにおいて、システム全体を対象とするといった無駄な工数の発生や、テスト漏れといった問題がほとんどのプロジェクトで発生している。

3. 目的

本研究では、派生開発時における変更による影響箇所の把握方法、および、テスト範囲を特定する方法を提案することを目的とした。また、テストレベルとしては、システムテストや受け入れテストを対象にした。その理由は、それより前の単体テストや結合テストなどでは、変更箇所そのもののテストが主体となり、影響箇所の特定が課題となるのは、システムテストや受け入れテストであると考えたからである。

目的の達成により、テストの優先度の判定や適切な工数見積りにも活用することができる。

4. 活動の経緯

テスト範囲を特定するための改善案を提案するにあたり、以下の手順で活動を進めた。

(1)先行研究の調査

XDDPにおける影響範囲の特定方法として、いくつかの報告([2][3][6])がされている。しかし、これらは、変更箇所へのテストには言及しているが、変更によって発生するシステムの他の機能への影響については論じていない。また、影響範囲に気づく成果物として機能間の依存関係を表すマトリクスの必要性を主張しているが具体的な表現方法には触れていない。

一方、ソフトウェア変更影響に関する先行研究[7]では、変更影響を二元表で可視化する手順を

提案している。これと目的は同様であるが、今回の研究では、XDDPの成果物と組み合わせる方法を考案した。先行研究では、テスト範囲特定のため、予めComponentの仕様の詳細把握が必要となっている。一方、今回考案した方法では、最初はテスト範囲特定までの情報は要求せず、影響範囲の特定に必要なレベルまでとし、その後にテスト範囲を特定することとした。派生開発の短納期という前提条件下では、まずは影響箇所を知った上で、ピンポイントにテスト範囲を特定していく方が、現場での適用が容易であると考える。

(2) 開発手法の調査

開発手法の調査として、XDDP による開発手法での成果物の調査をした。XDDP は主要な下記 3 つの成果物を作成し派生開発に特有の課題を解決するプロセスである。[3]

<表 1.派生開発成果物とその情報>

成果物	情報種別	概要
変更要求仕様書	What	変更要求を明確化する資料
TM:トレーサビリティ・マトリクス	Where	変更箇所を特定する資料
変更設計書	How	変更方法を定義する資料

(3) 問題点の把握

派生開発のテストプロセスの問題について検討した。

派生開発におけるテストは以下の 2 つに分類される。

- 変更箇所に対して行うテスト
- 変更箇所により影響を受ける箇所に行うテスト

前者については、XDDP の成果物の利用により、テスト範囲を特定することができる。

後者については、今回変更された箇所が影響を及ぼす箇所を把握することが必要であるが、XDDP の成果物だけでは影響箇所を特定することができない。なぜならば、情報として変更箇所の影響による制御フロー、変更に伴う処理の順序性といった影響箇所が示されていないためである。

(4) 問題点解決のための解決策の検討

前述の問題を解決するには、機能同士の関係性を表現するための「機能間マトリクス」が必要であると仮説を立てた。これにより、システム全体の機能のうち、どこが変更箇所であるかということに加えて、その他の箇所との関係性が把握できるからである。XDDP の「TM」と今回考案した「機能間マトリクス」を活用すれば、以下の 3 つに変更箇所による影響箇所を分類することができる。

- 変更箇所：変更した箇所そのもの(TM で識別可能)
- 影響あり：変更していない、かつ、変更箇所により影響のある箇所(機能間マトリクスで識別可能)
- 影響なし：変更していない、かつ、変更箇所により影響のない箇所(機能間マトリクスで識別可能)

また、このように分類することで、テスト範囲の特定をすることができると考えた。

(5) 解決策の妥当性の検証

解決策の妥当性を示すために、「C 言語プログラミング能力認定試験」の 1 級試験のテーマプログラム[4]を題材とした。

テーマプログラムのサンプル問題を開発要件として、「TM」と「機能間マトリクス」を作成し、変更箇所による影響箇所の特定ができるかを検証した。なお、本検証での機能間マトリクスの変更箇所の粒度はテーマプログラムの中で機能として識別できる関数単位とした。各資料については付録を参照のこと。

5. 各表の概要と記述手順

今回用いた「TM」と「機能間マトリクス」の概要と記述手順を示す。尚、TM の記述手順は XDDP に原則として準拠しているため、今回の提案部分は機能間マトリクスの記法についてである。

5.1 TM(付録 1)

ソフトウェアに求められる開発要件と機能との関係を表す表である。本表の利用で、開発要件に対する関連機能を把握することができる。以下に TM の作り方と交点の記述例を示す。

(1) TM の作り方

- 縦軸にソフトウェアに求められる開発要件を記述する。粒度については、XDDP の変更要求仕様にて抽出された単位とする。
- 横軸に、ソフトウェアが持つ機能を記述する。XDDP では、横軸に沿って記述される機能の情報を、常に固定しておくことを推奨している。粒度については、ソフトウェアの規模や構成によるので一概に言えないが、本研究では、後述する機能間マトリクスとの整合性を考慮し、関数単位とした。

<表 2：TM>

	機能 A	機能 B	機能 C	機能 D	機能 E	機能 F	機能 G	機能 H
開発要件①				○				
開発要件②				○				
開発要件③			○		○			

(2) 交点の記述

開発要件から変更が発生する機能に対して「○」を記述する。上記の表 2 を文章で表現すると以下のようになる。

「開発要件①に対し、機能 D が変更される。開発要件②に対し、機能 D が変更される。開発要件③に対し、機能 C および機能 E が変更される。」

5.2 機能間マトリクス(付録 2)

機能間マトリクスは、機能間の「親子関係」「順序関係」および「競合関係」を表す表である。本表の利用で、TM で示された、変更箇所に対する影響箇所を把握することができる。以下に機能間マトリクスの作り方と交点の記述例を示す。

(1) 機能間マトリクスの作り方

- 縦軸に TM で示した機能(既存の機能名および開発要件に対し新規追加する機能名)および共有資源(本研究ではグローバル変数)を記述する。機能の配置順は処理の時系列順にするのが可読性の点で妥当と思われる。

・ 横軸に縦軸の記述と同じ機能(既存の機能名および開発要件に対し新規追加する機能名)を記述する。機能の配置順は縦軸に合わせる。

<表 3：機能間マトリクス>

	機能 A	機能 B	機能 C	機能 D	機能 E	機能 F	機能 G	機能 H
機能 A		○					○	
機能 B			○,1	○,②	○,②	○,3		
機能 C								
機能 D								○
機能 E								
機能 F								
機能 G								
機能 H								
共有資源Ⅰ	●	●		●				●
共有資源Ⅱ	●		●				●	
共有資源Ⅲ	●						●	●

(2) 交点の記述

機能と機能の関係

設計情報にある、機能構成図、機能定義書等を入力情報として、縦軸の機能に対し横軸の機能が親子関係にあり順序関係にない場合は「○」を記述し、親子関係にあり順序関係にある場合は「○」を「数値」に置き換えて(例：1、2・・・)記述する。

<表 4：交点の表記>

項番	親子関係	順序関係	交点の表記
1	あり	なし	○
2	なし	あり※	数値(例：1、2・・・)
3	あり	あり※	項番 1,2 の情報をカンマ区切りで併記する
4	なし	なし	記入しない

※機能間に複数の順序関係のない機能がある場合、「○」と「数値」を合わせて(例：②)記述する。

表 3 の機能と機能の関係を文章で表現すると以下のようになる。

「機能 A は機能 B、機能 G と親子関係である。機能 B、機能 G に順序関係はない。機能 B は機能 C、機能 D、機能 E、機能 F の親である。機能 C、機能 D、機能 E、機能 F は機能 C 後に機能 D、機能 E、機能 F が動作するという順序関係がある。ただし、機能 E、機能 F には順序関係がない。機能 D は機能 H と親子関係である。」

共有資源と機能の関係（競合関係）

縦軸の共有資源を使用している横軸の機能に「●」を記述する。

表 3 の共有資源と機能の関係を文章で表現すると以下のようになる。

「共有資源Ⅰは機能 A、機能 B、機能 D、機能 H で使用されている。共有資源Ⅱは機能 A、機能 C、機能 G で使用されている。共有資源Ⅲは機能 A、機能 G、機能 H で使用されている。」

6　TM と機能間マトリクスを使った影響箇所の抽出

TM と機能間マトリクスを使った、変更箇所に対する影響箇所の抽出手順について示す。

(1) TM にて、開発要件に対する変更機能を確認する。

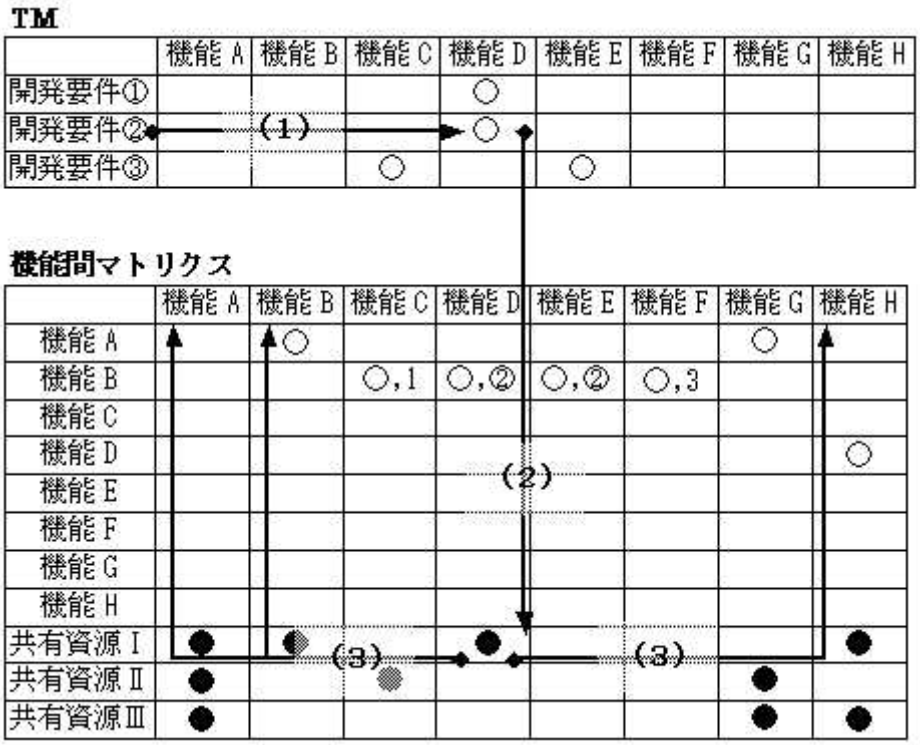
確認内容：開発要件②に対し、機能 D を変更している。

(2) 機能間マトリクスにて、変更機能で使用している共有資源の使用有無を確認する。

確認内容：変更機能 D では、共有資源Ⅰを使用している。

(3) 機能間マトリクスにて、(2)で確認した結果、共有資源の使用が他の機能で有る場合、変更箇所に対する影響箇所として抽出する。共有資源の使用が無い場合、変更箇所に対する影響箇所は、変更箇所のみである。以下の確認内容を元にテスト範囲の特定を行う。

確認内容：(2)で確認した共有資源Ⅰは機能 D の他に、機能 A、機能 B、機能 H で使用している。これによりテスト範囲は、変更箇所の機能 D のみではなく、影響箇所として抽出された機能 A、機能 B、機能 H を含めた箇所とする。



<図 1: TM と機能間マトリクスを使った影響箇所の把握>

7．効果

テーマプログラムのサンプル問題にて、TM のみを使用した場合と、TM と機能間マトリクスの両方を使用した場合(6. (1),(2),(3)を実施)において、変更箇所と影響箇所を把握結果がどのように変化するかを比較した。その結果、TM のみを使用した場合は、影響あり・なしの判断が不明であったのに対し、TM と機能間マトリクスを併用した場合には、変更箇所による影響のあり・なしを把握することが可能となり、テスト範囲として特定することができた。(なお機能数は、付録 1：開発要件 3 に対する変更箇所から抽出した影響箇所数である。)

<表 6：検証結果（機能数）>

	変更箇所	影響あり	影響なし	影響不明
TM	1	-	-	27
TM と機能間マトリクス	1	6※	21	0

※1 変更箇所数を含む

8. 考察

効果の項で述べたとおり、本研究の方法により影響あり・なしが明確に機能単位で特定できたことから、「変更箇所による影響箇所を把握しテスト範囲を特定すること」という目的に対して、期待通りの効果が得られたと考える。

機能間マトリクスを用いることで変更箇所とその他の機能の関連性が見える状態となっているため、機械的に影響範囲を見出すことができた。機械的な方法であるが故に、テストケース作成者の経験度やスキルレベルによるばらつきを押さえることが可能であり、これは少人数、短納期、必ずしも既存システムに精通している要員が担当しているわけではない、という派生開発の現状を考えると、有効な点であると考えている。また、影響範囲特定のツール化も可能である。これにより現場への適用がさらに容易になると考えている。

一方、「テストの優先度の判定」、「適切な工数見積もりにも活用」という二点は、今回の検証では直接的に効果を測定することまではできなかった。

また、その他に、検証の過程で気づいた点として以下が挙げられる。

・機能間マトリクスの作成負荷は書式化、規則化により現実的に作成が可能なレベルであった
・関係性の類型や機能の抽出単位については適用時に入念な検討が必要である

機能間マトリクスの作成負荷は書式化、規則化により現実的に作成が可能なレベルであった、という点であるが、機能間マトリクスをテスト設計の参考成果物とすることによるデメリットとして、その成果物を作成する負荷自体が問題となることを懸念していた。しかしながら、実際に試してみたところ、マトリクス作成に際して書式化、記述方法の規則化を進めておけば比較的機械的な作業として取り組むことができた。なお、一旦作成すれば保守開発・派生開発の案件の都度、変更箇所のみをメンテナンスすればよいので、十分に実用に耐えると考える。

関係性の類型や機能の抽出単位については適用時に入念な検討が必要であるという点は、試した際に苦労した点になる。システムの複雑性や規模により、どのレベルの機能をマトリクスの一要素として抽出するかというのが重要になる。これについては、適用の際の検討課題となるが、あまり粗すぎると関連性が識別できなくなるので、モジュール単位やユースケース単位といったレベルが望ましいと考えている。

9. 課題

「テストの優先度の判定」、「適切な工数見積もりにも活用」について検証できなかった点は課題である。機能間マトリクスで影響箇所が可視化できるため、テスト計画やテスト設計（テストケース作成）の作業のインプットにすることにより、改善が可能ではないかと考える。

また、以下の点については当研究だけでは解決できないテーマである。

・候補の抽出には有効であるが実際にテストするか否かは個別に判定が必要
例）あるファイルにデータ項目の区分値を追加したとする。
そのファイルを後続で入力としている機能があれば、機能間マトリクスからデータの受け渡し（データフロー）による関連を読み取ることまではできる。しかしながら後続の機能が区分値を追加したデータ項目を使用しているかどうかまでは分からないため、実際にテストが必要かどうかは個別に機能の詳細を確認する必要がある。

・大規模システムにおける機能間マトリクスの分割方法
相当な大規模システムの場合には、機能間マトリクス自体を分割する必要がある。大規模システムは、サブシステムに分割されている場合が多いため、サブシステム単位に作成するなどが一般的と考えられる。しかしながら、サブシステム間をまたがる関連の取り扱い方については今回の研究では実際の試行・検証はできていない。想定としてはサブシステム間のインタフェースを別途整理して併用するという形になると考えている。

10. おわりに

本研究では、派生開発における影響度把握の方法について、XDDP の成果物を補足する形での改善を試みた。内容としては機能間マトリクスにて変更対象の機能と関連を持つ（影響を受ける）機能を可視化し、テスト範囲の特定に役立てるというものである。これにより、影響度の把握に有効であることが確認できた。今後は、実際の業務の中での利用を通じて課題の解決を含めて更なる改善を図っていきたい。

11. 参考文献

[1] JUAS 第 15 回企業 IIT 動向調査 2009
(http://www.juas.or.jp/project/survey/it09/summary09_0507.pdf)
[2] 清水吉男:派生開発における母体に由来するバグとその対応, JaSST'09 講演資料
(http://jasst.jp/archives/jasst09e/pdf/A2.pdf)
[3] 清水吉男:「派生開発」を成功させるプロセス改善の技術と極意, 技術評論社, 2007
[4] C 言語プログラミング能力認定試験 (http://www.sikaku.gr.jp/js/index_cp.html)
[5] 清水吉男:要求を仕様化する技術・表現する, 技術評論社, 2005
[6] 清水吉男:テストの質を上げるための要求仕様書, JaSST'07 講演資料
(http://jasst.jp/archives/jasst07e/pdf/B5-1.pdf)
[7] 日科技連 第 20 年度ソフトウェア品質管理研究会：ソフトウェア変更影響の可視化手順書
(http://www.juse.or.jp/software/pdf/20_spc/8/7_c_report_1.pdf)

付録1 トレーサビリティマトリクス

	開発要件＼機能	会員管理メイン制御	入会登録処理	空きコード表読込み処理	実行確認入力処理	空きコード表更新処理	会員名入力処理	計測日付入力	計測データ表追加処理	コード・データ対照表更新処理	計測記録入力処理	空きコード表読込み処理	コード・データ対照表読込み処理	会員コード入力	計測日付入力	計測データ入力	入力計測データ表示	実行確認入力処理	個人計測データ読込み	運動指数計算	個人計測データ更新	個人計測データ表示	計測データソート・表示	計測結果順位ソート	計測結果順位表示	登録削除処理	コード・データ対照表削除	個人計測データ読込み	個人計測データ表示	実行確認入力処理	個人計測データ削除	空きコード表追加	空きコード表読込み処理	空きコード表作成処理	コード・データ対照表作成処理
1	「入会登録処理」において、会員氏名（姓）、会員氏名（名）及び入会日付を入力させる		○				○	○	○																									
2	会員氏名（姓）、会員氏名（名）を以下の条件で入力させる ①1文字以上の半角英字である。 ②すべて半角英大文字に変換する。 ③8文字以上入力したときは、先頭の8文字とする。 ④8文字未満の時は、左詰めで残りの桁は空白で埋める。		○				○		○																									
3	「計測日付入力処理」を利用し、入会日付を入力させる		○					○	○																									
4	「個人計測データ表示」において、会員コードの次に会員氏名（姓）、会員氏名（名）、入会日付を表示させる																					○						○						

共有機能
非共有機能

付録2 機能間マトリクス

機能No	機能名	機能No																																	
		001	002	003	004	005	006	007	008	009	010	003	011	012	007	013	014	004	015	016	017	018	019	020	021	022	023	015	018	004	024	025	003	026	027
001	会員管理メイン制御		○								○																								
002	入会登録処理			○,1	○,2	○,3	○,4	○,5	○,6	○,7																									
003	空きコード表読込み処理																																		
004	実行確認入力処理																																		
005	空きコード表更新処理																																		
006	姓名入力判定																																		
007	計測日付入力処理																																		
008	計測データ表追加処理																																		
009	コード･データ対照表更新処理																																		
010	計測記録入力処理											○,1	○,2	○,3	○,3	○,3	○,4	○,5	○,6	○,7	○,8	○,9	○,10												
003	空きコード表更新処理																																		
011	コード･データ対照表読込み処理																																		
012	会員コード入力処理																																		
007	計測日付入力処理																																		
013	計測データ入力処理																																		
014	入力計測データ表示処理																																		
004	実行確認入力処理																																		
015	個人計測データ表読込み処理																																		
016	運動指数計算処理																																		
017	個人計測データ表更新処理																																		
018	個人計測データ表示処理																																		
019	計測結果順位ソート･表示処理																							○,1	○,2										
020	計測結果順位ソート処理																																		
021	測定結果順位表示処理																																		
022	登録削除処理																										○					○			
023	コード･データ対照表削除処理																											○,1	○,2	○,3	○,4				
015	個人計測データ表読込み処理																																		
018	個人計測データ表示処理																																		
004	実行確認入力処理																																		
024	個人計測データ表削除処理																																		
025	空きコード表追加処理																																○		
003	空きコード表読込み処理																																		
026	空きコード表作成処理																																		
027	コード･データ対照表作成処理																																		

No	共有資源名																																		
I	アキコードヒョウ	●	●	●		●					●	●																				●	●	●	
II	コードデータタイショウヒョウ	●								●			●	●						●		●						●	●						●
III	ニュウリョクケイソクデータテーブル															●	●				●														
IV	コジンベツケイソクデータヒョウ	●							●											●		●	●						●	●		●				
V	ソートヨウケイソクデータヒョウ	●																						●	●	●									

共有機能
非共有機能