研究コース３（GrayFox グループ）

ソフトウェア変更の影響範囲を考慮したスコア付けによるテストケース選定手法の提案

-DFD を利用したデグレード不具合の検知率向上-

Proposal of method for selecting test cases by the score of change impact at software - Improving detection rate of degradation using a data flow diagram –

リーダー　　：櫻田　健人（ヤマハ発動機株式会社）
研究員　　：石川　雄基（アイシン・エィ・ダブリュ株式会社）
　　豊田　千弘（カルソニックカンセイ株式会社）
　　坂東　文香（テックスエンジソリューションズ株式会社）
　　吉田　健雄（テックスエンジソリューションズ株式会社）
　　吉田　伸幸（アンリツエンジニアリング株式会社）
主査　　：喜多　義弘（東京工科大学）
副主査　　：上田　和樹（日本ナレッジ株式会社）
アドバイザー：秋山 浩一（富士ゼロックス株式会社）

研究概要

派生開発においては，新機能に対するテストの他に従来機能のデグレードがないことを回帰テストでテストする．回帰テストの項目すべてをテストすることは難しいため，変更に対する影響を考慮しながらテスト項目を選択する必要がある．選択されたテスト項目を使って回帰テストを行い，機能のデグレードを全て検出できることが望ましい．しかし実際には，回帰テストで全てのデグレードを検出できず，テスト全体の効率性を悪くしている．

そこで本研究では，システムテストの工程において，DFD を用いて要求機能にスコア付けを行い，その値を基に回帰テストのテスト項目を選択する手法を提案する．本手法を評価するために，実際にテスト工程が完了している派生開発プロジェクトに対して本手法を実験的に適用してみた．その結果，テスト項目の選択が改善されていることを確認した．

Abstract

In derivational development, it is necessary to conduct regression testing in order to confirm that the conventional functions do not degrade. It is difficult to test all items of the regression testing. We have to select the test items while considering the influence on the functions. It is the best that degradation of the functions is detected by the selected test items of regression testing. However, the selection of the regression test items is not enough to detect all the degradation. Therefore, the efficiency of testing phase is worse.

We propose a method to select items of regression test that items are scored by the functions using a data flow diagram at the system test phase. We experimentally applied the proposal method to the derivational development that already completed up to the testing process. We confirmed that the testers could select the test items accurately at the regression testing with our proposal method.

1.　はじめに

派生開発の現状として，短納期かつリソースが少ない現場におけるシステムテストフェーズでの回帰テストで，デグレード不具合を見つけることが課題となっている．研究員の担当する派生開発プロジェクトでは，デグレード不具合が占める割合は全体の不具合の約

1 割あり，その発見方法に回帰テストが用いられる[1]．従来の機能を一通り動作させる回帰テストではテストケース数が膨大な量であるため，自動化が用いられる．しかし，自動化ツールの導入やメンテナンスにおけるコスト，ツール導入に伴うメンバーのスキル習得やシステムのアーキテクトの問題などの要因により自動化が難しい場合には，全てのテストケースを限られた工数内で完了させることは困難である．このことからシステムテストフェーズにおける回帰テストでは短納期でのテスト実行を実現するために，変更に対して関連性の高いテストケースを選定することが必要である．

「影響波及パス分析法[2]」は，派生開発のシステムテストフェーズにおけるスクリプトテストに焦点を当て，制御の流れとデータ更新・参照の流れから影響箇所を特定可能という考えとソースコード解析の技術をもとに，影響波及パスという概念を定義し，それを利用した分析手法である．しかしこの手法は開発者自らがプログラミングした範囲に対してのみに効果が認められるものであり，他の開発者がプログラミングした内容とのつながりまでは考慮されていない．

本稿では「影響波及パス分析法」にて定義された概念を基に，システムテストに適用範囲を広げるために追加検討した「SFD 手法(System Flow Diagram Method)」を提案する．

SFD 手法とは，下記の特徴を持つ．

・デグレード不具合が発生する可能性のある範囲をシステムレベルで捉えて，影響波及パスを導出する．
・影響範囲に対するスコア付けに応じて優先順位を割り当て，実施すべきテストケースを選定可能である．

以下，本稿の構成を述べる．2 章では，解決すべき問題と関連する先行研究を示す．3 章では，SFD 手法を適用した解決策を提案する．4 章では，解決策の評価と結果の考察を示す．5 章では，まとめと今後の進め方を示す．

2. 解決すべき問題

2.1 現状分析

派生開発においては，新規機能が正しく動作するかを検証することに加えて，機能として変更がない部分に対して，従来通り正常に動作するかを検証する必要がある．従来機能においても，新規機能追加のために内部的に変更が必要であり，そのことが問題なく動作していた従来機能に影響しデグレード不具合を生じさせる可能性がある．デグレード不具合を検出するために，派生開発の検証においては従来の機能を一通り動作させる回帰テストを行う．回帰テストのテストケース数は元のシステムが大きければ大きいほど膨大な量となるため，テストの自動化が行えない場合，短納期である派生開発では日程・リソース的にすべてのテストケースを実施するのが困難である．

研究員が担当する実際のプロジェクトでも回帰テスト実施の際に，従来機能に対する全てのテストケースを短時間で行うことは難しく，テスト設計者が今までの経験や開発に与える影響度から，スクリプトテストのテストケースを削減したことでデグレード不具合が検出されず，探索的テストで検出されている事例がある．

ここで研究員が参画していた新規開発プロジェクト及び派生開発プロジェクトの状況について言及する．図 1 は新規開発プロジェクトと派生開発プロジェクトにおけるスクリプトテストの実行テストケース数を示し，図 2 と図 3 では新規開発プロジェクトおよび派生開発プロジェクトにおいてデグレード不具合の検出されたテストの割合を示す．図 1 より，派生開発プロジェクトで実行したスクリプトテストのテストケース数は新規開発プロジェクトと比較して 5 割前後まで減少しているのが確認できる．また，図 2，図 3 では新規開発プロジェクトにおけるスクリプトテストで 9 割以上のデグレード不具合が検出されていることが示されているが，派生開発プロジェクトにおいてはスクリプトテストでのデグレード検出は 3 割程度であり，残りの 7 割は探索的テストにおいて発見されているということが示されている．

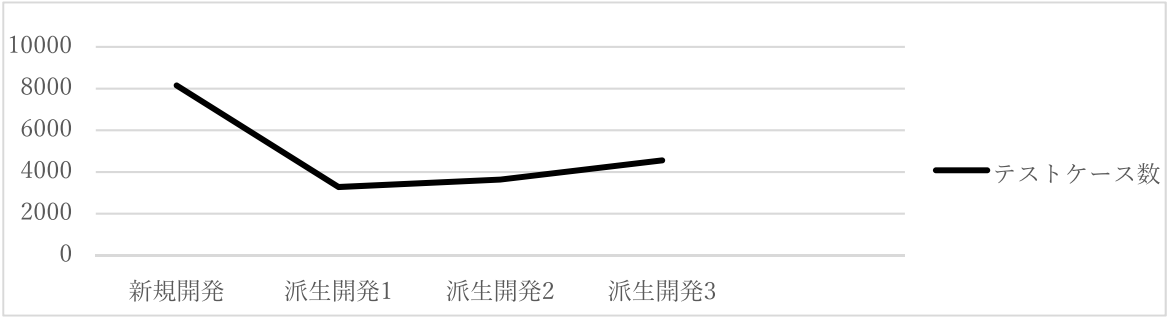

図 1　システムテストにおけるテストケース数の変遷(現場)

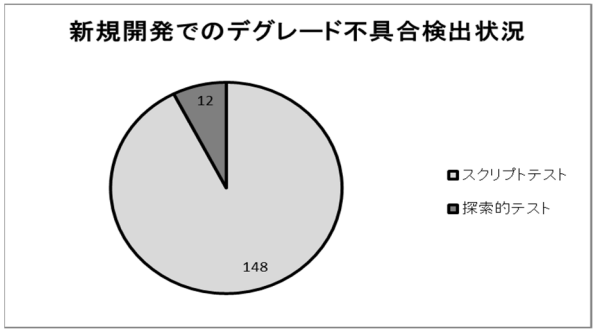

図 2　新規開発でのデグレード
不具合検出状況

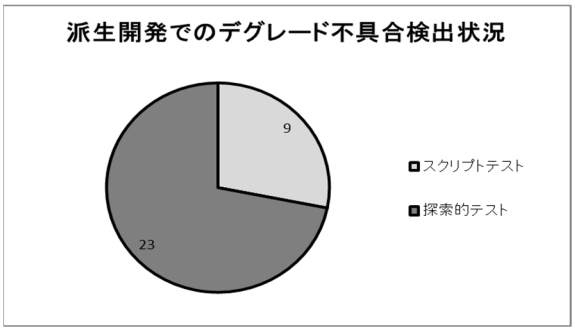

図 3 派生開発でのデグレード
不具合検出状況

2.2　課題提起

前節の現状分析より，派生開発におけるデグレード不具合の多くは探索的テストにより検出されていることが確認できた．ここで，従来機能に対するテスト範囲を明確にすべき回帰テストにおいて現状が適切であるかを考える．

探索的テストは Cem Kaner 氏[3]，James Bach 氏ら[4]によって提唱された，テストのスタイルあるいはアプローチをいう．テストを実施する過程で，テスト担当者がテスト実施情報を活用しながらテスト設計をコントロールし，短納期で質の高い不具合の検出が行える手法[5]である．

探索的テストに対し，スクリプトテストはテスト設計・実装・実行というプロセスを順に行ってテストケースを洗い出し，これを実行するアプローチを取る．探索的テストと比較した場合，テスト設計ドキュメントの作成や変更対応のたびに起こる保守に対するコストが掛かり，検出する不具合に対するコストが探索的テストに比べて大きい[6]．

しかし，探索的テストが短納期で質の高い不具合の検出を発揮するのはテスト対象を熟知した実施者が行う場合のみであり，実施者の知識やスキルに依存するなど属人的な結果となりやすい．このため探索的テストは，スクリプトテストに比べて少ないドキュメントしか残さないという特性上，テストのカバレッジという観点から品質を保証しづらいという欠点がある．

回帰テストでは，テストを実施した範囲の中で従来機能のデグレードがないことを示すため，カバレッジの保証が求められる．そのため，研究員の所属している現場では回帰テストを行う際には通常探索的テストと共にカバレッジを担保するためにスクリプトテストも実施する．しかし，前節でも述べたようにカバレッジを保証するための従来機能に関わるテストケースを全て実施することは難しく，その自動化においても課題を抱えている．研究員の現場でも回帰テストに割り当てられるリソースでは従来機能に対する全てのテストケースを行うことは難しく，変更点に対してテスト設計者が要否を判断しテストケースの削減が行われている．この削減に対する判断基準が不明確な場合，削減方法がテスト設計者の主観になると,テストのカバレッジが下がるために,不具合流出のリスクが上がる．

そのため変更点に対する影響を定量的に示し，従来検出できていたデグレード不具合も見つけられる判断基準が必要である．この判断基準を用いて，カバレッジをコントロールしつつ不具合流出のリスクを抑えるテストケース選定を行う必要があると考える．

2.3 先行研究

テストケース削減のための判断基準を定義する手法として，リスクベースドテスト[7]のような手法が挙げられる．リスクベースドテストでは，ある機能に不具合が発生した場合に生ずる「顧客への迷惑度」を，不具合の頻度やユーザーに対する影響度からテストケースの重要度を評価し，その重要度を判断基準としてテストケースの削減を行う．しかし，テストケースの重要度を評価する際には，該当システムの特性を深く理解している有識者に聞き込みを行う必要がある．研究員へのヒアリングではこの点に関して，担当者の入れ替わりや十分な時間が確保できないなどの理由によりリスクベースドテストが導入されていない．そこで，本研究では内部構造に着目した評価方法について検討を行った．

柏原らの先行研究[2]ではモジュール単位の変更に対する影響度を分析し，テストのカバレッジを選定する手法が考案されている．影響波及パス分析法は，統合テストフェーズの回帰テストにおいてソースコード解析とコールカバレッジの考え方を利用した手法であり，その結果として未経験者でも，経験者と同等の質で影響範囲に対するテストケースの抽出が可能となっており，有識者レビューによるテスト漏れ検出件数は 0 件であったことから効果が得られたと述べている．

上記より，ソフトウェアの変更に対する影響度を考慮した影響波及パス分析法を用いることで，変更の影響度を定量化した判断基準を抽出できると考えた．

3. 解決策の提案

影響波及パス分析法における影響波及パスの導出には，インプットとしてソースコードに加えて変更された関数のリスト,関数呼び出し関係リストなどの統計情報が用いられる．これらソースコードや関数といった粒度の情報を，今回の対象であるモジュールや機能が統合されたシステム全体をテストするテストフェーズに用いた場合,粒度が細かすぎることにより，全プロセスに適用すると膨大な工数になるので使用することはできない．そのため,影響波及パス分析法を今回の対象フェーズに適用する場合,システム全体の機能を俯瞰できるようなインプットが必要である.また,インプットを選定するにあたり,現状のシステムテストフェーズではどのようなデグレード不具合が問題となっているのか,過去の後工程流出状況から検討を行った．

3.1 課題の解決方針

影響波及パス分析法の適用先を統合テストフェーズからシステムテストフェーズに変更する際に，システムテストフェーズではどういったデグレード不具合を見つけるべきかを検討した．表 1 は研究員の現場で実施したデグレード不具合の後工程流出状況を元に，その要因分析を行った結果である．ここでは検出された不具合に，その要因の種類，回帰テストケースが削減されたことにより漏れたか，どのテストフェーズで見つかったかを分類した．不具合の要因の種類については，今回「データベース由来」，「他社製品由来」「内製モジュール由来」「不明」の 4 種に分類してある．「データベース由来」については機能間におけるデータや入出力の仕様不一致の不具合であり，「他社製品由来」については他社製品での既存不具合や仕様不一致，「内製モジュール由来」は関数変更における影響の不具合であり,「不明」は不具合分析を行ったが原因がわからなかったものである．また，これに加えて表 2 に表 1 の分析結果における要因別の詳細な要因を示す．

上記のデグレード不具合検出の要因分析結果より，全デグレード不具合の 32 件中，「データベース由来」が 5 件，「他社製品由来」が 7 件，「内製モジュール由来」が 7 件，「不明」は 4 件であった．「データベース由来」が要因であるものについては，データの加工に関する誤りや外部インターフェースにおける誤り，そしてデータ定義での誤りから不具合が発生した．「他社製品由来」における不具合は仕様書が開示されていないため，詳細

は不明であった．

次に「内製モジュール由来」の不具合項目については主にモジュール内での処理に原因のあるものであった．これら不具合項目はデータの流れに関する不具合とモジュール内の処理に関する不具合に大きく分類できると考える．モジュール内の処理に関する不具合については関数単位で変更を加えたことによる影響であるため，柏原らの影響波及パス分析法を用いることにより統合テストの段階で不具合検出が行えると思われる．

しかし，データの流れに関する不具合については，統合テストでは検出の難しい不具合であり，通常システムテストフェーズで発見されるものである．

よって，システムテストフェーズにおける回帰テストでは，データの流れに着目することで，データ依存の不具合を検出するべきであると考えた．

ここで，影響波及パス分析手法をシステムテストフェーズに用いたときに，データ依存の不具合を検出するための最適なインプットを考える．データに依存する不具合を検出するためには，各機能においてどのようなデータのやり取りが行われているのかを明確にしなくてはならない．また，影響波及パスを用いて分析後，各機能におけるテストケースの優先順位付けを行う必要があるため，機能とテストケースのトレーサビリティが取れるドキュメントが必要である．よってデータ依存の不具合を検出するための最適なインプットは上記の要求を満たす DFD[8]（データフローダイアグラム)を利用する．DFD とは要素間のデータの流れを表した図であり，データがどこで発生し，どこからどこへ運ばれ，どこへ出力・保管されるのかを図示するものである．今回はプロセスを機能単位とした DFD を作成することでシステムテストフェーズにおける影響波及パス分析法に用いることとした．

表 1　デグレード不具合検出の要因分析結果

デグレード 不具合要因	回帰テストケースが削減されたことにより漏れたのか	見つかったテスト	集計
他社製品や内製モジュールに起因しているもの	No	スクリプトテスト	9
データベース	Yes	探索的テスト	5
他社製品	Yes	探索的テスト	7
内製モジュール	Yes	探索的テスト	7
不明	Yes	探索的テスト	4
総計			32

表 2　表 1 の不具合詳細要因分析結果

デグレード不具合要因	詳細要因	対応策
データベース	データ加工ミス	本研究提案手法
	外部インターフェース誤り	本研究提案手法
	データ定義誤り	本研究提案手法
他社製品	仕様が不明なため特定できず	−
内製モジュール	モジュールの繋がりの間における，A 対応における B の対応漏れ	影響波及分析法
内製モジュール	インターフェースの変換対応漏れ	影響波及分析法
内製モジュール	ログの掃き出しすぎ	−
内製モジュール	上位モジュール-下位モジュールの関係で，下位モジュールを変更したための弊害	影響波及分析法

3.2　SFD 手法の定義

本研究では，影響波及パス分析法に DFD を用いて，システムテストフェーズに適用する SFD 手法(System Flow Diagram Method)の提案を行う．

回帰テストのスクリプトは機能ごとに作成されているため，本手法で使用する DFD はプロセスではなく機能で記載する．

したがって本手法で使用する DFD とは下記 3 点を満たすものと定義する．

1：機能同士のつながりがわかること
　○対象範囲システム:自システムのみとする
　○非対象システム：他システムは非対象とする．他システムは仕様を知る方法がないため，ブラックボックスとして扱い対象外とする

2：機能間の入出力がわかること
　○データの種類は問わないこと
　○データの入出力方向を記述すること

3：機能がどのシステムに所属しているかがわかること

SFD 手法を，図 4 の DFD を例に挙げて説明する．

図 4 の例では以下の機能で構成されている．

・変更のあった機能 A
・機能 A にデータを Input する Database X
・機能 A の Output を利用する機能 B，機能 C
・機能 B の Output を利用する機能 D，機能 E
・機能 D の Output を利用する機能 E，機能 F，機能 G
・機能 F の Output を利用する機能 B
・機能 A の Output を利用する他システム

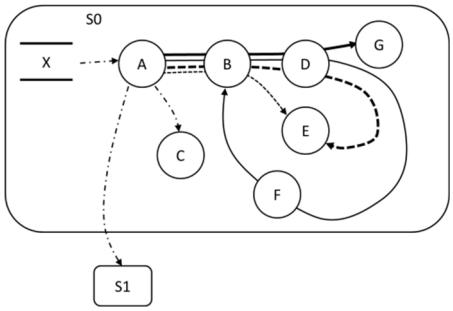

図 4：SFD 手法による DFD のデータ経路の一例

表 3：DFD の一例をスコア付けした結果

変更機能Aから影響を受ける機能	B	C	D	E	F	G
スコア	4	1	3	2	1	1

矢印は機能間のデータの流れを表している．SFD 手法では機能間のデータの流れを追って，変更機能からのデータを扱う末端の機能までの経路を導出する．ここで末端の機能とは，DFD 中でデータの行き先が一度通った経路に合流した，または行き止まりになった機能である．データ経路を導出した結果を，図 4 内の①太線②細線③太破線④細破線⑤一点鎖線で表す．この機能間に記載された線の数がそのままデータ経路の数であり，これを「スコア」とする．

SFD 手法を使うと図 4 の A-B 機能間では①②③④のデータ経路があるので A-B 機能間のスコアは 4 となる．表 3 に SFD 手法により導出された変更機能 A から見た，それぞれの機能のスコアは機能に対する Input から導出される．ここで導出されたスコアは，機能 B は別の機能へ更に影響を及ぼすためスコアが高く，機能 C は別の機能に影響なく完結するためスコアが低いという結果となる．このスコアに応じて優先順位を割り当て，実施すべきテストケースの選定が可能となる．

4. 解決策の評価

評価を実施するにあたり，上流工程で作成される DFD が必要となる．今回の評価では研究内で定義した前提条件を基に，実際に検出した不具合と，それに関連する機能・システムに当てはめて，図 5 のように DFD を作成した．

S0 は自システム，S1～S6 は他システム，A～K は機能，Data a ～ Data d はデータベースを示す．また，機能に対しては既存テストケースとトレースの取れる ID を記載した．A に関しては今回変更のあった機能であり，これらの機能が変更された際に影響の波及先に対してのスコア付けを実施し，既存テストケースに対してスコアを当てはめることで評価を実施する．スコア付けの結果は表 4 のようになった．

既存テストケースに A～K の機能を割付し，そこに表 4 の結果を当てはめ，スコア毎にテストケースを絞ることで表 5 を得た．これは，スコア毎にテストケース数を定量的に提示している．

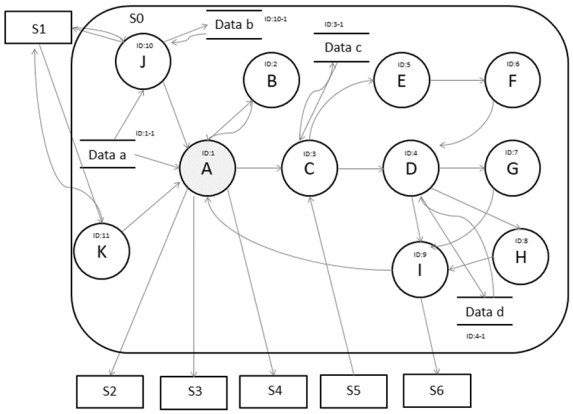

図 5：研究員の現場で開発されたシステム間のデータのやりとり

表 4：図 5 を元に導出したスコア付け結果

変更機能Aから影響を受ける機能	B	C	D	E	F	G	H	I	J	K
スコア	1	15	7	7	7	2	2	6	1	1

変更機能Aから影響を受けるDatabase	Data a	Data b	Data c	Data d
スコア	1	0	1	1

変更機能Aから影響を受けるSystem	S1	S2	S3	S4	S5	S6
スコア	0	1	1	1	0	1

表 5：機能 A に関係するテストケースのスコア付け結果

変更機能	新規開発時_テストケース数	手法適用後のスコアとテストケース数					
		スコア:0点以上	スコア:1点以上	スコア:2点以上	スコア:6点以上	スコア:7点以上	スコア:15点以上
A	245	245	237	191	186	167	158

表 5 より，研究員の現場の回帰テストのテストケースに本手法「SFD 手法」を用いることで，テストケースを定量的な値でスコア付けをすることができた．また，A の変更による実際のプロジェクトにて発生した後工程へ流出したデグレード不具合の検出状況の SFD 手法による改善は表 6 である．従来手法においても，同様な考え方で A～K の機能を割付しており，C と I の機能に対して，それぞれの機能に関連するテストケース数およびそのテストケースで検出される不具合についてまとめている．C と I の選定基準は，選定したテストスクリプトの中に，A，C および I が同時に出現していたため，C と I に対しての改善が確認できれば SFD 手法の有効性について述べることが可能と考えたためである．従来手法では C および I に関してはデグレード不具合が発見されなかったが，SFD 手法において C と I は，A からの影響が大きいことが分かり，C と I に関連するテストケースに厚みが出て，不具合が検出されることを確認した．

表 6 の結果より，SFD 手法の有効性について確認ができた．ただし，今回はどのスコア付けが最適であるかまで結論付けするには至らず，現場のリソースに応じて選択し適用する必要がある．

表 6：機能 A 変更における他機能への影響と改善結果

手法	分類	機能	
		C	I
従来手法	不具合件数	0	0
	テストケース数	7	10
SFD手法	不具合件数	4	3
	テストケース数	15	19

また，今回の取り組みを通して下記 2 点の気づきが得られた．

・新規開発時に行うテスト分析として DFD を用いて影響範囲を特定することが重要である．それにより回帰テスト時に速やかにテストケースの絞り込みが可能となる．

・DFD を作成するうえで，どの粒度で作成するのか事前に開発者と議論が必要であること.今回は評価をするうえで研究に必要な粒度で DFD を作成したが,実際の現場では，開発者が DFD を作成することになるため，事前に合意を取る必要がある．

5. まとめ

本論文では，派生開発におけるシステムテストレベルの回帰テストでのデグレード不具合の検出状況に基づいて，DFD を用いたテスト優先度の検討時に指標となるスコア付けの提案を行った．SFD 手法を適用することで，不具合流出のリスクを抑えるテストケース選定を行えることを説明した．今回は提案した手法を利用して 1 つのシステムに対して部分的に適用した例を示した．その結果から今後の取り組みとして 3 つの課題が挙げられる．ひとつはシステム全体に適用した場合のスコア付けである．今回の取り組みでは対象機能を部分的にスコア付けできることを示したが，今後はシステム全体に対してスコア付けした場合の検証を行う必要がある．次に，スコア割付け後のテストケース選定に最適な閾値決めである．今回は，定量的な指標としてスコアを提示できるようにしたが，どの閾値が最適であるかは現場の判断に委ねることとしているため，最適な閾値決めができる仕組みの確立が必要である．最後に SFD 手法を適用するプロセスの確立である．今回は SFD 手法の検証数が少ないため，様々なプロジェクトへのテーラリングには至っていない．今後はプロジェクトを考慮した SFD 手法の運用プロセスを検討したい．

6. 参考文献

[1] 西康晴，古川善吾，ソフトウェアテスト総論，情報処理，pp1-6,2008.

[2] 柏原一雄，村上雅哉，小嶋秀和，都築功，統合テストにおいて影響範囲に対するテスト漏れを防止する「影響波及パス分析法」の提案，ソフトウェアテストシンポジウム 2017，pp1-8，2017.

[3] Cem Kaner，Testing Computer Software，Wiley，1999.

[4] James Bach，http://www.satisfice.com/articles/et-article.pdf,(2018 年 12 月 23 日参照).

[5] ソフトウェアテスト技術振興協会，JSTQB 認定テスト技術者資格-シラバス（学習事項）・用語集-，http://jstqb.jp/syllabus.html(2018 年 12 月 23 日参照)

[6] 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，p93 2013.

[7] 石田智亮，石山康介，米田征弘，西康晴，Risk Based Testing の実践方法と適用事例からの考察～短期間で要求品質を確保する～，ソフトウェアテストシンポジウム 2008，pp1-29，2008

[8] 青山義彦，ソフトウェア開発に用いられる図形表現法，情報処理学会,pp1-24,1984.