90

「影響波及パス分析法」の適用事例 ～統合テストでの影響範囲に対するテスト漏れ防止～[1]

1. 概要

派生開発では、ソースコードの変更箇所だけでなく、変更の影響箇所に潜在する欠陥を、回帰テストで見逃さず検出することが必要である。また、短納期での開発が求められるため、統合テストにおける回帰テストでは、変更に対する影響範囲を特定したうえで、実施するテスト項目を絞り込まざるを得ないことが多い。影響範囲に対するテスト漏れを防止することは、派生開発の比率が高い我々の組織の重要な課題であった。

我々のプロジェクトでは、統合テストにおいて、変更の影響範囲に対するテスト漏れを防止するためには、「確実な影響範囲の特定」「テストの網羅性の確認」「影響範囲を小さくする設計」の 3 つが必要であると考えた。そして、この 3 つの原則を現実的に実行可能とする手法「影響波及パス分析法」を考案した。

考案した影響波及パス分析法は以下の特徴を持つ。

- デグレードが発生する可能性のある影響範囲を絞り込み、絞り込んだ影響範囲に対して漏れなくテスト実施していることを確認する
- 「対象システムの有識者がいない」「網羅的なテストをするためのテスト自動化環境がない」という特徴を持つプロジェクトにも適用可能である
- ソースコード解析とカバレッジメトリクスを応用した手法

本編では、影響波及パス分析法と手法を導入するために構築した仕組みと、手法導入により得られた考察を紹介する。

2. 取り組みの目的

今回の取り組みは、変更の影響箇所で発生するデグレードを防止するために、統合テストにおける回帰テストの問題を解決することが目的である。解決する回帰テストの問題を詳しく説明する。

2.1. 流出を防止する欠陥

今回の取り組みでは、派生開発において、「変更の影響なし」と判断した領域で発見される欠陥の流出を防止することを目的としている。我々の組織における欠陥の検出領域の調査結

[1] 事例提供：株式会社デンソークリエイト 柏原 一雄 氏

果、欠陥の流出は変更箇所よりも変更の影響箇所で多く発生していることがわかった。我々の組織における欠陥の検出領域の割合を図 90-1 に示す。影響範囲が 17%、「変更の影響なし」と判断した領域が 69%であり、8 割以上が影響箇所で発見される欠陥である。

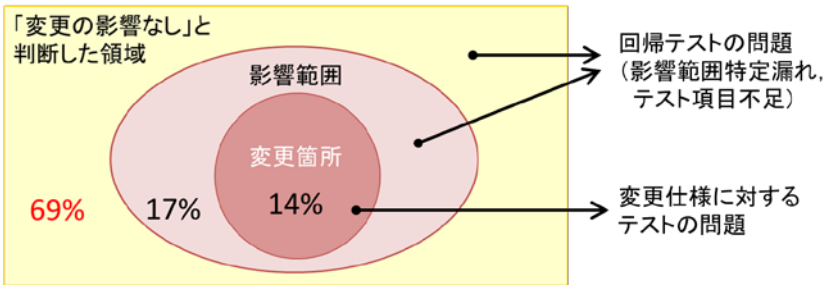

図 90-1 欠陥流出領域の分布

今回の取り組みでは、この影響箇所で発見される欠陥のうち、統合テストで検出すべき欠陥の流出を防止することを目的とした。

2.2. 統合テストにおける回帰テストの問題

回帰テストで対象範囲となる未変更部分は、変更量とは無関係に広いため、テスト漏れを防ぐためには、膨大な量のテストが必要となる[7]。しかし、短納期で多くのバリエーションの開発が求められる傾向にある現状の組込みソフトウェア開発の場合には、回帰テストで全範囲を網羅的にテストすることは困難である。そのため、回帰テストでは、変更に対する影響範囲を特定したうえで、実施するテスト項目を絞り込むことが多い。このような状況下で実施する回帰テストでは、以下の問題が発生していた。今回の取り組みでは、この 2 つの問題を解決した。

・影響範囲の特定誤り

回帰テストでは、仕様ベースで作成された既存のテスト項目から、必要な項目を選定している。変更仕様を示す文書はあるが、変更により影響する仕様を明確に示す文書はないため、仕様からテスト項目を選定することは困難である。変更仕様からではなく、設計・ソースコードの変更点から影響する仕様を特定する必要がある。変更前の設計を把握していない担当者の場合、特に影響する仕様の特定誤りを起こしやすい。

・テストの網羅性が確認できない

ユニットテストでは MCDC[2]等の網羅性を示す指標を計測する仕組みがあるが、統合テストレベルで影響範囲に対するテストの網羅性を計測する仕組みがない。ユニットテストを網羅的に実施しても、図 90-2 のようにデータを介して関数間で波及する影響がある場合、デグレードを見逃すことがある。膨大な量になる未変更部分に対するテストの網

[2] MCDC：Modified Condition Decision Coverage 国際技術標準 D0-178B(RTCA)準拠

羅性をレビューなどの方法により人手で確認することは困難であり、テスト漏れを見逃す可能性は高い。

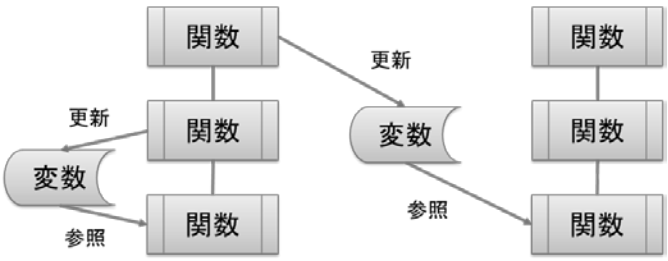

図 90-2 データを介して関数間で波及する影響がある場合のイメージ

3. 取り組みの方針

課題を解決するために参考とした技術と課題の解決方針を示す。

3.1. 課題

影響範囲に対するテスト漏れを防止するためには、「影響範囲の特定誤り」と「テストの網羅性が確認できない」という問題を解決することが必要である。

テスト漏れを防止するために、松尾谷[1]は検証指向設計とテストデバッグという理論を提唱している。検証指向設計は、テスト能力の範囲内で設計をするべき、つまり検証空間が小さくなる設計をすべきという理論である。テストデバッグは、テストのデバッグ環境を与えると飛躍的にテスト漏れを少なくすることが可能であり、テストの網羅計測ができる環境を用意すべきという理論である。

この理論をもとに、影響範囲に対するテスト漏れを防止するため、以下の 3 つの原則を考えた。3 原則のイメージを図 90-3 に示す。

A) 確実な影響範囲の特定

人の知識に頼らず、手続き的に影響範囲を特定する。

B) テストの網羅性の確認

特定した影響範囲に対して網羅的にテストしたことを確認する。

C) 影響範囲を小さくする設計

テストする影響範囲を小さくするように設計する。

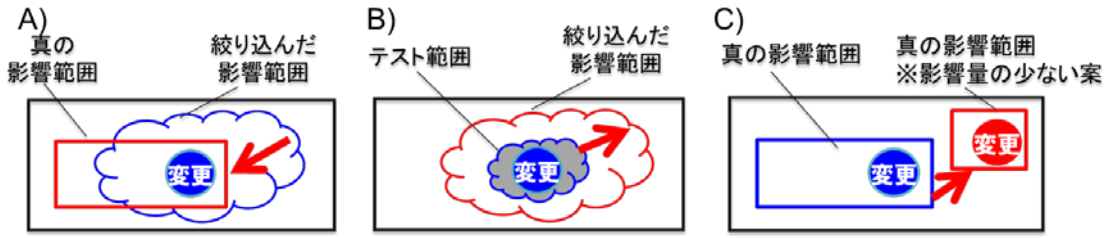

図 90-3 影響範囲に対するテスト漏れを防止するための 3 原則のイメージ

今回の取り組みでは、影響範囲に対するテスト漏れを防止するための 3 原則を現実的に実行可能とする合理的な手法を開発した。考案手法は「対象システムの有識者がいない」「網羅的なテストをするためのテスト自動化環境がない」という特徴を持つプロジェクトにも適用可能とした。

3.2. 参考とした技術

「確実な影響範囲の特定」を可能とするためソースコード解析を利用した影響分析手法を参考とした。また、「テストの網羅性の確認」を可能とするため、コールカバレッジの考え方を利用した。

3.2.1. 影響分析手法

影響分析手法に関する先行研究には、トレーサビリティマトリクスを使用した手法とソースコード解析を利用した手法がある。

トレーサビリティマトリクスを使用した手法として、二元表[3]や機能間マトリクス[4]、T 型マトリクス[5]を用いた手法が提案されている。関連する機能やテスト項目を特定可能とするためのトレーサビリティマトリクスを作成するには、知識と時間が必要であり、完全性の保証や保守が困難であると考えた。トレーサビリティマトリクスを作成するにしても、全範囲ではなく影響範囲を絞り込んで作成したいと考えた。

考案手法では、人の知識に頼らず手続き的に影響範囲を特定するため、ソースコード解析を利用した手法をベースとした。ソースコード解析を利用した手法には、表 90-1 のように Call Graph、Dependency Analysis、Program Slicing といった手法が存在する。解析結果はグラフで表現されることが多い。考案手法では、特定した影響範囲をもとに「テストの網羅性の確認」と「影響範囲を小さくする設計」を可能とするために、ソースコード解析結果の表現方法を工夫している。「テストの網羅性の確認」を可能とするために、テスト実施結果との比較を可能な表現とした。また、「影響範囲を小さくする設計」を可能とするために、設計案毎の影響範囲の大きさを比較可能な表現した。

表 90-1 ソースコード解析を利用した手法[6]

手法名	説明
Call Graph	関数、メソッド呼び出し（制御構造）の依存関係を明らかにする。
Dependency Analysis	制御、データ、継承（オブジェクト指向プログラミング言語の場合）による依存関係を明らかにする。
Program Slicing	調べたいソースコードの一部を選択し、そこに関連する部分のみを抽出する。

3.2.2. カバレッジメトリクス

統合テストレベルでテストの網羅性を評価するためのカバレッジメトリクスとして、コー

ルカバレッジ[8]がある。コールカバレッジは、ソフトウェア全体のすべての関数コール数と、テストにおいて実行された関数コールの割合を示すメトリクスである。コールカバレッジは、影響範囲に対して計測するメトリクスではない。考案手法では、コールカバレッジの考え方をもとに、影響範囲に対してテストの網羅性を評価するための指標を定義した（表 90-2）。

表 90-2 カバレッジメトリクス[8]

カバレッジ メトリクス	利用テストレベル	説明
ステートメントカバレッジ	ユニットテスト	全コード内の実行命令に対するテストケースの網羅率を示す。
ブランチカバレッジ	ユニットテスト	全コード内のブランチ（分岐）に対するテストケースの網羅率を示す。
MC/DC	ユニットテスト	全コード内の条件や判定に対するテストケースの網羅率を示す。
ファンクションカバレッジ	統合テスト	ソフトウェア全体の関数数と、テストにおいて実行された関数の割合を示す。
コールカバレッジ	統合テスト	ソフトウェア全体のすべての関数コール数と、テストにおいて実行された関数コールの割合を示す。

3.3. 考案手法「影響波及パス分析法」

3.3.1. 概要

影響波及パス分析法は、想定外のデグレードが発生する可能性のある領域を絞り込み、統合テストレベルでの回帰テスト項目を漏れなく選定するための手法である。以下の 3 つの方針で手法を開発している。

A) 影響範囲はソースコード解析で抽出
「命令依存関係とデータ依存関係から影響箇所を特定可能」[2][9]という考え方をもとに、ソースコードから影響範囲を抽出する。

B) 影響範囲に対してテスト網羅性を評価
コールカバレッジを影響範囲に対して計測可能とする。そのために、影響範囲はコールカバレッジが算出可能な形式で表現する。

C) 影響範囲を設計にフィードバック
設計案毎に影響範囲の大きさを比較可能とする。そのために、影響範囲は比尺度[10]で表現する。

影響範囲を示すために、変更箇所を起点とした関数コールの繋がりを表現した「影響波及パス」という概念を定義した。影響波及パスは、スタティックスライシング技術[9]の「命令の間のデータ依存関係と制御依存関係を用いて、命令の実行に影響を与える可能性のあるすべての命令の集合を抽出することが可能」という考え方を参考に定義した。

影響波及パス分析法のイメージを図 90-4 に示す。2 つのフィードバック・ループを備えていることが特徴である。 1 つ目は、不足テスト項目の追加を可能とするフィードバック・ループである。テスト網羅性の情報を回帰テストにフィードバックする。2 つ目は、影響範囲が小さい設計案の選択を可能とするフィードバック・ループである。影響量の情報を設計にフィードバックする。これにより、影響範囲を小さくし、影響範囲の特定漏れの防止とテスト項目および工数の削減に繋げる。

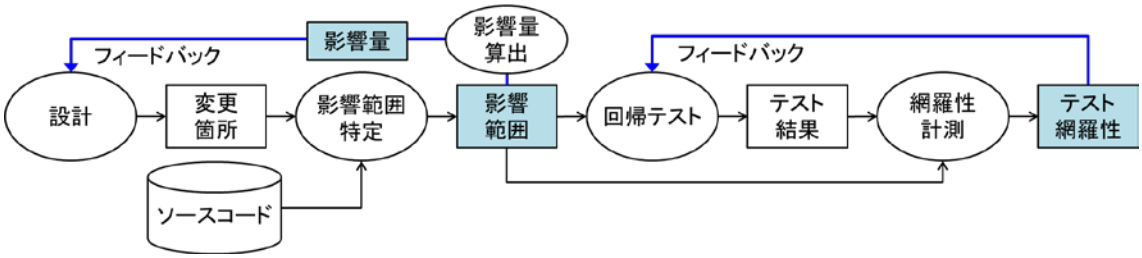

図 90-4 影響波及パス分析法のイメージ

3.3.2. 影響波及パスの定義

影響波及パスは、制御とデータの流れによる影響箇所を関数コールパスで表現する。影響波及パスのイメージを図 90-5 に示す。また、影響波及パスの抽出に必要となる要素を表 90-3 に定義する。

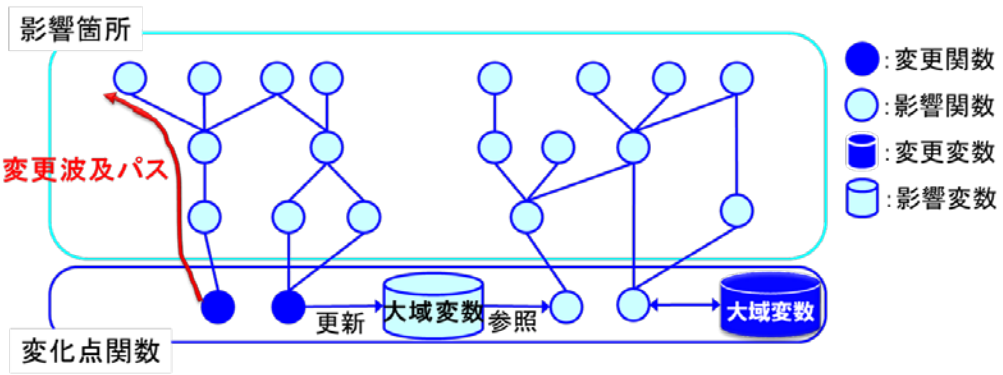

図 90-5 影響波及パスのイメージ

表 90-3 影響波及パスに必要となる要素の定義

名前	説明
変更関数	実際に変更（追加、削除）が行われた関数。
変更変数	実際に変更（追加、削除）が行われた変数。
影響関数	影響波及パス上の関数。変化点関数の呼び出し元関数。
影響変数	変更関数で更新（生成、更新、削除）があった大域変数。
変化点関数	影響波及の起点となる関数。変更関数は変化点関数の 1 つ。大域変数によって影響が波及する以下の関数も変化点関数と位置付ける。
変化点データ	影響波及の起点となる大域変数。

影響波及パスは、変化点関数を末尾として変化点関数から外部定義関数までの関数名の繋がりを以下のように表現する。

[影響関数名（外部定義関数）]→[影響関数名]→・・・→[変化点関数名]

この定義の結果、回帰テスト対象製品上のすべての関数を、影響関数と非影響関数に分類できる。非影響関数は、回帰テストの対象とならず、影響量にも含まれない。

3.3.3. 影響波及パス分析法の定義

影響範囲を特定するだけでなく、特定した影響範囲をもとに「テストの網羅性の確認」と「影響範囲を小さくする設計」を可能とするために、影響波及パス分析法は、次の 3 つを技術要素とする。

・影響波及パス分析活用プロセス

影響波及パス、影響波及パスカバレッジ、影響波及パス数を入出力とした活動の定義

・影響波及パスカバレッジ

テスト網羅性を示すメトリクス

・影響波及パス数

影響量を示すメトリクス

(1) 影響波及パス分析活用プロセス

影響波及パス分析を活用したプロセスのフローを図 90-6 に示す。プロセスフロー内の各活動の詳細定義は表 90-4 に示す。

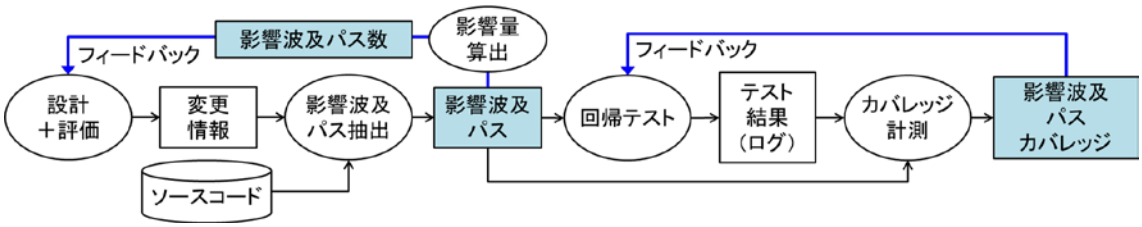

図 90-6 影響波及パス分析活用プロセスのフロー

設計の変更情報とソースコードから影響波及パスを抽出する。そして、影響波及パスをもとに、影響波及パスカバレッジ、影響波及パスを算出する。

影響波及パスカバレッジのフィードバックは、不足テスト項目を追加するアクションに結びつける。影響波及パス数のフィードバックは、設計を見直すアクションに結びつける。

表 90-4 影響波及パス分析活用プロセスの各活動の定義

活動	説明
影響波及パス抽出	ソースコードと変更情報（変更関数、変更変数）を入力として影響波及パスを抽出する。ソースコード解析により、手続き的に影響波及パスを抽出する。
設計＋評価	影響波及パスをもとに計測した影響波及パス数を入力として設計評価をする。複数の設計案に対して、それぞれ影響波及パス数を算出することで、影響量の少ない（テスト項目の少ない）設計案を選択する。
カバレッジ計測	影響波及パスとテスト結果（関数コールログ）を入力としてカバレッジ計測を実施し、影響波及パスカバレッジを計測する。
回帰テスト	影響波及パスを入力に、不足しているテスト項目を追加する。また、計測した影響波及パスカバレッジを入力とし、不足しているテスト項目を特定し、回帰テストを再実施する。

(2) 影響波及パスカバレッジ

影響波及パスカバレッジは、コード内のすべての影響波及パスに対するテストが、少なくとも 1 回は実行されるようにするための指標である。影響波及パスカバレッジは変化点関数毎に算出する。以下の計算式で算出する。 図 90-7 に影響波及パスカバレッジ算出のイメージを示す。

【計算式】

影響波及パスカバレッジ = 影響波及パスとテスト実行パスの合致数 / 影響波及パス数

影響波及パス	存在判定	テスト実行パス
/･･･/･･･/･･･/･･･/	○	/･･･/･･･/･･･/･･･/
/･･･/･･･/･･･/･･･/	×	-
/･･･/･･･/･･･/･･･/	○	/･･･/･･･/･･･/･･･/
/･･･/･･･/･･･/･･･/	×	-

変化点関数名	影響波及パス数	テスト実行パスの合致数	カバレッジ
･･･	1	0	0.0%
･･･	3	2	66.7%
･･･	34	19	55.9%
･･･	34	34	100.0%

図 90-7 影響波及パスカバレッジ算出のイメージ

影響波及パスとテスト実行パスの合致数と影響波及パス数は、変化点関数毎に算出する。影響波及パスに対して、合致する関数コールログが 1 つ以上存在している場合、影響波及パスとテスト実行パスが合致と判断する。影響波及パスカバレッジから、テスト実施されていないパスを特定し、不足テスト項目を追加することが可能となる。

(3) 影響波及パス数

影響波及パス数は、影響量を示すためのメトリクスである。変化点関数を起点とした影響波及パスの合計数である。変化点関数毎に算出した影響波及パス数を合算することができる。これにより、開発案件毎や変更要求毎に、影響波及パス数を集計することも可能となる。

影響波及パス数が大きくなれば、回帰テスト項目数も大きくなる。影響波及パス数を確認することで､変更量ではなく影響量を比較し､設計案を選択することが可能となる。具体的には､図 90-8 に示すように､変更関数数だけでなく影響波及パス数を比較したうえで、設計案を選択することが可能となる。

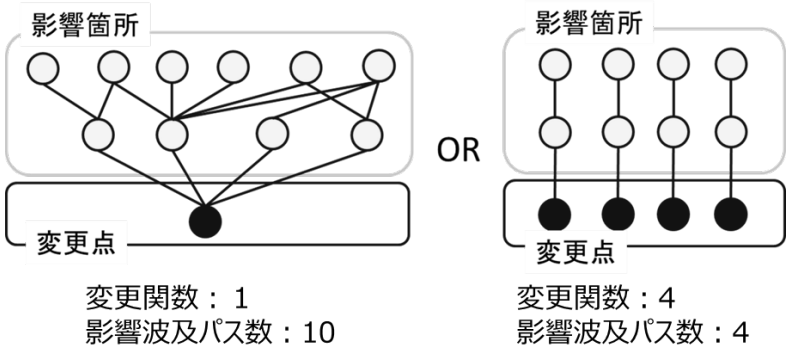

図 90-8 影響量を考慮した設計案選択のイメージ

変更量を比較しているだけでは考慮することが難しかった影響範囲に対するテスト項目数やテスト工数を考慮し、設計案を選択することが可能となる。

4. 取り組みの事例

影響波及パス分析法を適用したプロジェクトの特徴、手法適用のために構築した仕組み、取り組みの結果を説明する。

4.1. プロジェクトの特徴

考案手法を適用したプロジェクトで開発しているソフトウェアは、構造化手法で設計を行い、C 言語でコーディングをしている。また、以下の状況で、統合テストの回帰テストにおけるテスト漏れを防止するために、考案手法を適用した。

・開発担当者は、既存の仕様・設計・コードについての知識を持っていない。
・網羅的なテストを可能とするテスト自動実行環境は作成していない。影響範囲に対して、必要な分だけ自動テスト環境を構築する。

影響波及パス分析法は、あくまで統合テストレベルでの回帰テスト項目を漏れなく選定する手法である。変更の影響箇所に潜在する欠陥の流出を防ぐため、以下の対策も合わせて実施している。

・変更箇所および特定した影響範囲におけるテスト項目を仕様から過不足なく抽出するため、統合テスト設計では、状態遷移テストや CFD、デシジョンテーブルなどのテスト技法を利用する。
・ユニットテストで、関数単体での回帰テストを実施する。

4.2. 手法適用のために構築した仕組み

3.3.3.（1）影響波及パス分析活用プロセスに示した「影響波及パス抽出」と「影響波及パスカバレッジ計測」を実行可能とする仕組みを構築し、手法を実プロジェクトに適用可能とした。また、3.3.3.（1）影響波及パス分析活用プロセスに示した「回帰テスト」で、影響波

及パスから回帰テスト項目の選定をしやすくするために、外部定義関数-仕様-テスト項目のトレースを可能とする仕組みを用意した。

4.2.1. 影響波及パス抽出の仕組み

既存の静的コード解析ツールでは、関数ポインタによる関数呼び出しやポインタを利用してデータにアクセスしている箇所の特定が困難である。これに対して、ツールで解析が困難なことは手動で補う方針で、現実的に実行可能な仕組みを構築した。自動化できる作業と自動化できない作業を分離することがポイントである。また、解析対象のソースコードに、ポインタを使用しデータアクセスしている箇所や関数コールをしている箇所が少ないほうが、影響波及パス抽出作業は効率的に実施できる。

影響波及パス抽出では、変更関数・変更変数・影響変数・変化点関数を同定し、変化点関数を起点とする影響波及パスを明らかにする。影響波及パスの具体的な抽出手順は、DFD を用いて図 90-9 に示す。ソースコードに加えて、表 90-5 に示した設計情報を入力とする。

影響波及パスリストの作成は、ツールを開発し、自動実行を可能とした。通常の関数コール関係はソースコードから自動解析するが、関数ポインタによる関数コール関係は、人手で解析し、関数呼び出し関係リストにまとめ、影響波及パス生成の入力とする。

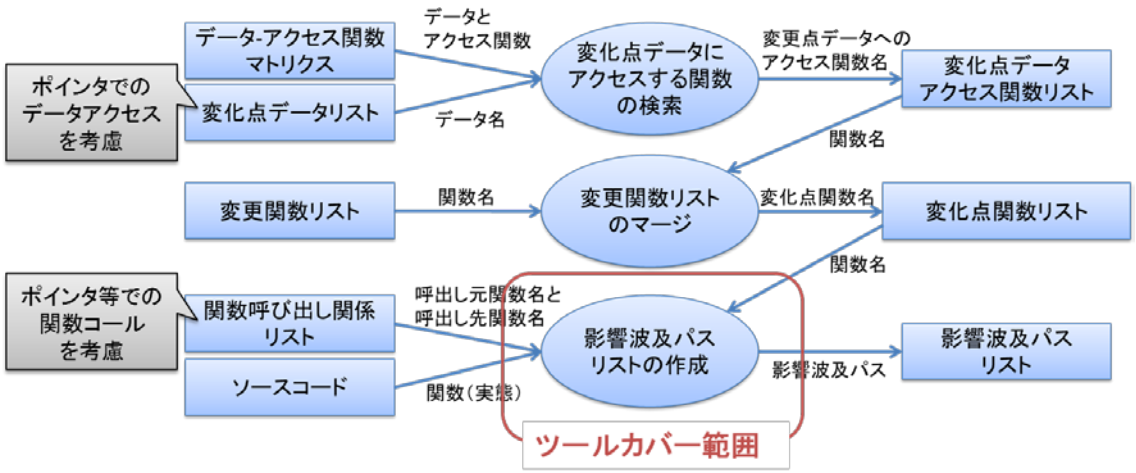

図 90-9 影響波及パス抽出手順

表 90-5 影響波及パス抽出に必要な設計情報の一覧

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td>変更関数リスト</td><td>変更された関数のリスト。</td></tr>
<tr><td>関数呼び出し関係リスト</td><td>呼び出し元関数と呼び出し先関数のリスト。
ソースコードから自動解析できない関数ポインタ等による特殊な関数呼び出し関係のみ示す。
【成果物イメージ】
<table><tr><th>呼出し元関数</th><th>呼出し先関数</th></tr><tr><td>FuncA</td><td>FuncB</td></tr><tr><td>FuncA</td><td>FuncC</td></tr><tr><td>FuncB</td><td>FuncC</td></tr><tr><td>FuncD</td><td>FuncE</td></tr></table></td></tr>
<tr><td>データ-アクセス関数
マトリクス</td><td>データと関数の関係を示したマトリクス。
ポインタによるデータアクセスも考慮する。
【成果物イメージ】
例：FuncA で DataA の参照処理と更新処理がある。
<table><tr><th rowspan="2">データアクセス関数</th><th colspan="3">データ</th></tr><tr><th>DataA</th><th>DataB</th><th>DataC</th></tr><tr><td>FuncA</td><td>WR</td><td></td><td>R</td></tr><tr><td>FuncB</td><td></td><td>W</td><td></td></tr><tr><td>FuncC</td><td>R</td><td></td><td>W</td></tr><tr><td>FuncD</td><td></td><td>R</td><td></td></tr></table>※W･･･データの更新処理あり
※R･･･データの参照処理あり</td></tr>
</table>

4.2.2. 影響波及パスカバレッジ計測の仕組み

影響波及パスカバレッジ計測では、影響波及パス抽出により特定された影響波及パスに対するテストが実施されているかを確認する。統合テスト実施により得られたテスト項目毎の関数コールログに、影響波及パスが含まれるかを確認する。影響波及パスカバレッジ計測の具体的な手順は、DFD を用いて図 90-10 に示す。また、成果物の説明を表 90-6 に示す。

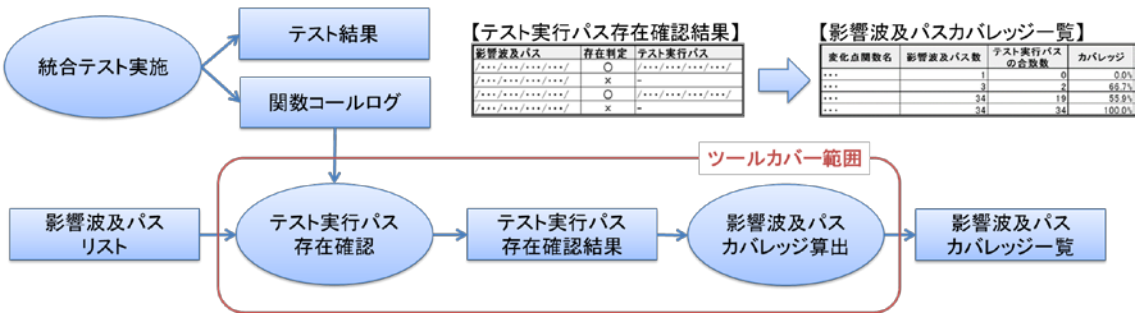

図 90-10 影響波及パスカバレッジ計測手順

表 90-6 影響波及パスカバレッジ計測の成果物

<table>
<tr><th>名前</th><th>説明</th></tr>
<tr><td>テスト実行パス存在確認結果</td><td>影響波及パスに対して、テスト実行パスが存在しているか否かを示す。
【成果物イメージ】
<table>
<tr><th>影響波及パス</th><th>存在判定</th><th>テスト実行パス</th></tr>
<tr><td>/・・・/・・・/・・・/・・・/</td><td>○</td><td>/・・・/・・・/・・・/・・・/</td></tr>
<tr><td>/・・・/・・・/・・・/・・・/</td><td>×</td><td>-</td></tr>
<tr><td>/・・・/・・・/・・・/・・・/</td><td>○</td><td>/・・・/・・・/・・・/・・・/</td></tr>
<tr><td>/・・・/・・・/・・・/・・・/</td><td>×</td><td>-</td></tr>
</table>
</td></tr>
<tr><td>影響波及パスカバレッジ一覧</td><td>変化点関数毎に、算出した影響波及パスカバレッジを示す。
【成果物イメージ】
<table>
<tr><th>変化点関数名</th><th>影響波及パス数</th><th>テスト実行パスの合致数</th><th>カバレッジ</th></tr>
<tr><td>・・・</td><td>1</td><td>0</td><td>0.0%</td></tr>
<tr><td>・・・</td><td>3</td><td>2</td><td>66.7%</td></tr>
<tr><td>・・・</td><td>34</td><td>19</td><td>55.9%</td></tr>
<tr><td>・・・</td><td>34</td><td>34</td><td>100.0%</td></tr>
</table>
</td></tr>
</table>

影響波及パスカバレッジ計測は、全手順をツールで実行可能とした。開発したツールでは、関数コールログと影響波及パスリストを入力に、影響波及パスカバレッジ一覧を出力する。ただし、本手順を自動実行可能とするためには、入力となる関数コールログが必要となる。統合テストで関数コールログを出力可能なツールを使用する必要がある。今回の事例では、テストツールとしてカバレッジマスターWinAMS（ガイオテクノロジー）を使用した。

4.2.3. 回帰テスト項目選定の仕組み

影響波及パスから回帰テスト項目を選定するために、トレース情報を設計書・テスト仕様書に記載している。具体的には、図 90-11 に示すように、仕様 ID を利用し、影響波及パスからテスト項目を選定可能としている。

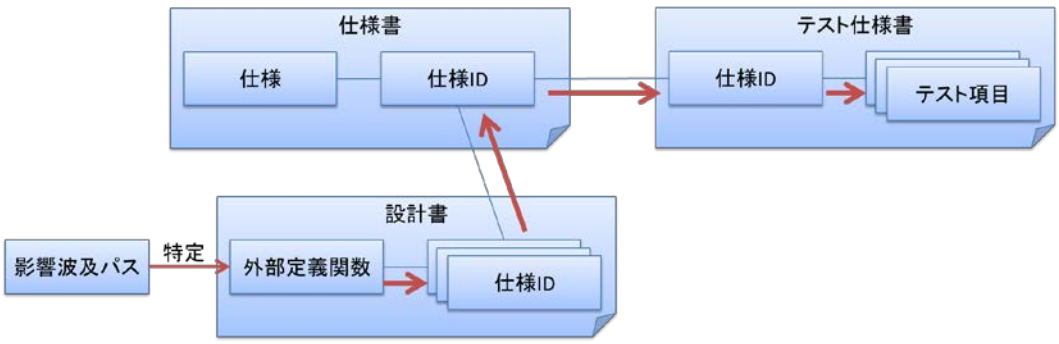

図 90-11 回帰テスト項目選定のためのトレース情報

仕様 ID は、外部定義関数とテスト項目に関連付けている。この情報により、影響波及パスから影響する外部定義関数名を特定し、外部定義関数名から関連する仕様 ID を特定し、仕様 ID から関連するテスト項目を特定するという流れで、回帰テスト項目の選定を可能としている。

今回の事例では、回帰テスト項目選定については、ツールは使用せず手作業で実施した。

4.3. 取り組みの結果

構築した仕組みを利用し、影響波及パス分析法を適用した結果を示す。

影響範囲に対するテスト漏れを防止するための 3 原則「確実な影響範囲の特定」「テストの網羅性の確認」「影響範囲を小さくする設計」が現実的に実行可能であるかを評価している。評価観点は次の 3 つである。

A) 経験なしでも経験者と同等の質で、影響範囲に対するテスト項目抽出が可能か？
影響波及パス分析により、テスト項目を抽出後に、有識者によるレビューでテスト項目の抽出漏れがないことを確認する。

B) テストの網羅性の計測・評価が可能か？
影響波及パスとテスト結果（関数コールログ）を入力として、影響波及パスカバレッジを計測し、テスト漏れを検出可能であることを確認する。

C) 変更の影響量の計測・評価が可能か？
ソースコードと変更情報（変更関数、変更変数）を入力として、影響波及パス数が計測可能であることを確認する。

影響波及パス分析手法を適用した結果を、適用案件毎に、表 90-7 に示す。手法を適用した案件は 5 案件である。

表 90-7 影響波及パス分析手法を適用結果

案件 ID	変更 LOC	影響波及パス数	影響波及パスカバレッジ	テスト漏れ検出数	
				影響波及パス分析	有識者レビュー
A	160	182	42.3%	0	0
B	130	14	42.9%	0	0
C	3649	68001	3.4%	12	0
D	585	288	59.7%	6	0
E	15	3022	10.1%	21	0

影響波及パス分析手法を適用した結果をもとに、観点毎に評価結果を示す。

A) 経験なしでも経験者と同等の質で、影響範囲に対するテスト項目抽出が可能か？
全案件で、有識者レビューによるテスト漏れ検出数は 0 件であった。有識者レビュー前に影響波及パス分析による回帰テスト項目選定をした効果が得られたと判断する。

B) テストの網羅性の計測・評価が可能か？
影響波及パスとテスト結果（関数コールログ）を入力として、影響波及パスカバレッジが算出できている。また、影響波及パスカバレッジをもとに、テスト担当者がテスト漏れを検出できている。具体的には、以下の問題に起因するテスト漏れを検出でき

た。

・テストデータ・環境（ドライバ・スタブ・入力値・期待値）誤り

・データ参照による影響波及の見逃し

・プリプロセッサ命令による処理切り替えがある場合の考慮不足

また､テスト実施しないパスについては､リスクとして共有することが可能となった。

このことから効果が得られたと判断する。

C) 変更の影響量の計測・評価が可能か？

全案件で、ソースコードと変更情報（変更関数、変更変数）を入力として、影響波及パス数を算出でき、設計レビューで変更 LOC と共に影響波及パス数を確認できている。また、設計者から「コーディング前に、影響波及パス数を計測し、設計案選択時に評価指標の 1 つとできる」とコメントを得た。このことから効果が得られたと判断する。

この結果から、変更の影響範囲に対するテスト漏れによる欠陥流出の防止に繋がる効果が確認できた。今回の手法導入による増加工数は、設計情報の整理の工数を除くと、変更規模が最も大きい案件でも、1 案件あたり約 16H（工数比率：約 2%）であり、工数面からも現実的に実行可能な手法であると言える。

5. 考察および今後の取り組み

5.1. 考察

今回の取り組みで､影響波及パス分析法を現実的に実行可能とするための仕組みを構築し、実開発に適用することができた。それにより、統合テストの回帰テストにおける「影響範囲の特定誤り」と「テストの網羅性が確認できない」という問題を解決することができた。解析しやすい・検証しやすいソースコードにする意識も高まる副次効果も得られた。影響波及パス分析法は、変更の影響箇所に潜在する欠陥を漏れなく検出することへの大きな貢献が期待できる。

また、今回の取り組みを通して、以下の 3 点の気づきが得られた。

・派生開発においてデグレードがないことを保証するためには、そもそも「影響範囲が広がらない設計」とすることが重要

・静的コード解析をツールで実行可能とするためには、ソースコードの解析容易性を向上させるためのコーディング基準（例：ポインタの使用制限等）が必要

・設計やテストの見直しのアクションに結びつきやすいように、フィードバックすべき情報・メトリクスを検討することが重要

5.2. 今後の取り組み

今後は、手法の改善、手法の展開、技術の応用を行っていく。今後の取り組みの詳細を以下に示す。

・手法の改善（使用性の向上）

ソースコードの解析容易性を向上させ、変更波及パス分析を効率的に実施可能とするために、設計基準・コーディング基準（ポインタ使用制限等）を定義する。また、回帰テスト項目選定作業をツールにより自動実行可能とする。

・手法の改善（信頼性の向上）

影響波及パス分析法を適用したプロジェクトで流出を防止できなかった欠陥がある。考案手法により、検出可能な欠陥、検出不可能な欠陥を明確に定義する。また、検出不可能な欠陥の情報をもとに、手法の改善を行う。更に必要に応じて、防止できなかった欠陥を検出できる新たな手法の開発を行う。

・手法の展開

他プロジェクトでも導入しやすい手法とし、考案手法の適用プロジェクトを拡大させるために、オープンソースの静的解析ツールの活用も検討する。

・技術の応用

メトリクス「影響波及パス数」は、見積りや品質データ分析で利用する規模を示す新たなメトリクスの１つとなり得る。メトリクス「影響波及パス数」の新たな活用場面を検討する。

本取り組みに対して、有限会社デバッグ工学研究所 松尾谷徹氏、堀田文明氏、株式会社デンソー 足立久美氏、株式会社デンソー技研センター 古畑慶次氏、株式会社デンソークリエイト 辻村健治郎氏、脇義昭氏、竹下千晶氏 から有益なご助言を戴いた。ここに、感謝の意を表する。

参考文献

[1] 松尾谷徹、「テストから観た実体のモデリングと実装構造の評価」、JaSST'15Tokai、2015

[2] 松尾谷徹、「何をテストするの？有則／無則／禁則」、ESEC 第 14 回、2011

[3] 第 20 年度ソフトウェア品質管理研究会 第 7 分科会、「変更影響の可視化によるテスト効率の向上」、(財)日本科学技術連盟、2004

[4] 第 25 年度ソフトウェア品質管理研究会 第 5 分科会、「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」、 (財)日本科学技術連盟、2009

[5] 派生開発推進協議会Ｔ４研究会、「Ｔ型マトリクスを用いたＸＤＤＰとテストプロセスの接続」、派生開発カンファレンス 2012

[6] 森崎修司、「ソースコード変更波及解析の 8 カテゴリ - どのくらいご存知ですか？」、http://blogs.itmedia.co.jp/morisaki/2013/03/8---7883.html、2013

[7] 松尾谷徹、増田聡、湯本剛、植月啓次、津田和彦、「Concolic Testing を活用した実装ベースの回帰テスト」、ソフトウェアシンポジウム 2015、2015

[8] ビジネスキューブ・アンド・パートナーズ、「ISO26262 実践ガイドブック」、日経 BP 社、 2013

[9] 下村隆夫、「プログラムスライシング技術と応用」、共立出版、1995

[10] 野中誠、小池利和、小室睦「データ指向のソフトウェア品質マネジメント」、日科技連出版社、2012

独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター (IPA/SEC)