実践コース

欠陥混入メカニズムをもとにしたベースソフト調査手法の提案

Proposal of base software investigation method based on "Defect injection mechanism"

研 究 員：柏原　一雄（株式会社デンソークリエイト）
主　査　：飯泉　紀子（株式会社日立ハイテクノロジーズ）
アドバイザー：足立　久美（株式会社デンソー）
　　：清水　吉男（株式会社システムクリエイツ）

研究概要

派生開発において，ベースソフトの変更漏れに分類される欠陥の流出を防止することは重要な課題である．我々の組織では，変更漏れを防止するために，XDDP をはじめとする手法を適用し，効果を確認している．しかし，ベースソフト調査方法の問題により，設計制約の変化の影響で変更が必要となる箇所については，変更漏れを防止しきれていない．問題を解決するために，担当者の能力・知識に依存せず，ベースソフトを調査する手法が必要であると考えた．本稿では，欠陥混入メカニズムの知識を蓄積・利用するベースソフト調査手法を提案する．過去に変更漏れが発生した事例に対して，提案手法を適用してベースソフト調査を再実施したところ，変更箇所を特定できることを確認した．

Abstract

In derivational development, preventing "the lack of change in base software" is an important issue. In our organization, in order to solve this problem, we introduced methods such as XDDP and confirmed that it is effective. However, due to the problem of the base software investigation method, with respect to "parts affected by changes in design constraints", "the lack of change in base software" can not be prevented. In order to solve the problem, we need a method to investigate the base software independent of the knowledge, experience and ability of the investigator. In this paper, we propose base software investigation method based on based on knowledge of "Defect injection mechanism". We applied the proposed method and re-conducted the investigation of the base software, it was confirmed that we can identify the changes that we could not identify before.

1.　はじめに

ベースソフトを部分的にしか理解できていない状況での作業が強いられる派生開発において，変更漏れに分類される欠陥の流出を防止することは重要な課題である．我々の組織でも，派生開発において，変更漏れに分類される欠陥の割合が多い傾向があり，対策が必要であった．

派生開発において，ソースコードの変更箇所を漏れなく特定するために，XDDP をはじめとする様々な手法が提案されている．我々のプロジェクトでは，ソースコードの変更箇所の特定漏れを防止するため，XDDP[2]と影響波及パス分析法[3]を適用している．これらの手法で「変更仕様から直接特定可能な箇所」「ソースコードの変化点から直接影響を受ける箇所」については変更漏れを防止できたが，「ベースソフトの設計制約の変化が影響して変更が必要となる箇所」については，変更漏れが防止しきれていない．

このような変更漏れの原因は，ベースソフト調査方法にあると考えた．自プロジェクトのベースソフト調査方法を調べたところ，次の 2 つの問題があることがわかった．1 つ目の問題は，ベースソフトを理解するための事前調査における「機能とデータ関係の把握漏れ」である．2 つ目の問題は，変更点を特定するための調査における「設計制約の変化に

よる影響箇所の抽出漏れ」である．どちらの問題も，ベースソフトの調査が，人の能力・知識・経験に依存していることから起きる．

本研究では，「設計制約の変化から影響を受ける箇所」を対象に，変更箇所の特定漏れ防止を目的として，ベースソフト調査の問題を解決可能な手法を開発する．「機能とデータ関係の把握漏れ」については，静的コード解析を活用し，担当者の能力を補い解決する方針とした．「設計制約の変化から影響を受ける箇所の抽出漏れ」については，変更箇所が問題なく特定できているときに利用している知識を明らかにし，その知識を蓄積し，担当者の知識を補い解決する方針とした．変更箇所が問題なく特定できているとき，担当者は欠陥が混入するメカニズムを理解しており，変化点から発生し得る欠陥を予測したうえで，ベースソフトを調査し，変更箇所を特定している．このことから，欠陥混入メカニズムの知識を蓄積し利用することで，担当者の知識に依存しない調査が可能となると考えた．

提案手法は，「データと機能の関係情報」と「欠陥混入メカニズムの知識」を利用するベースソフト調査手法である．

本手法によって，担当者の能力・知識に依存しない調査が可能となったかどうかを評価した．評価方法は，アンケートによる評価と過去の変更箇所特定漏れが発生した事例に対する再調査実施とした．欠陥混入メカニズム知識の整理と静的コード解析ツールの開発をしたうえで，評価を実施した．評価の結果，担当者の能力・知識を補い，手法適用前は特定できなかった変更箇所が特定できるように変わったことを確認した．

以降，2 章で研究の課題と関連する先行研究を示す．3 章では，解決策を提案する．4 章では，提案手法の評価結果と考察を示す．5 章では，まとめと今後の進め方を示す．

2. 課題設定

2.1 現状分析

(1) ベースソフトの変更箇所特定漏れの分類

組織の流出欠陥情報を分析し，派生開発で特定が漏れるソースコードの変更箇所には，次の 3 つのタイプがあることがわかった．

A) 変更仕様から直接特定可能な箇所
変更要求を実現するために，変更が必要な関数・データ等である．変更要求から漏れなく変更仕様が抽出できていれば，変更仕様から特定可能である．

B) ソースコードの変化点から直接影響を受ける箇所
A)で特定した変更箇所が，制御の流れとデータの流れを介して影響を与える箇所である．変更箇所が複数機能間の共通関数・共有データ等の場合に，デグレードを防ぐために，変更が必要となる箇所である．

C) 設計制約の変化から影響を受ける箇所
変更要求に対応することで，ベースソフトの設計の制約が変化することがある．例えば，「機能の並行実行不可」「機能のキャンセル不可」「バッファへの格納可能データ数」「データの参照可能条件（有効区間）」などの制約が変化することがある．この制約の変化に対応するために，変更が必要な箇所である．

組込みソフトウェア開発では，処理時間やリソース使用量等の要求に対応するために，制約を設けたうえで，設計をする場合が多い．そのため，C)の「設計制約の変化から影響を受ける箇所」があり，変更箇所特定が必要となる．

A)の「変更仕様から直接特定可能な箇所」については，XDDP を適用することで，変更漏れを防止することが可能である．B)の「ソースコードの変化点から直接影響を受ける箇所」については，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を適用することで，変更漏れを防止することが可能である．C)の「設計制約の変化から影響を受ける箇所」については，変更前と期待動作が変わるため，回帰テストによりデグレードがないことを確認しても，変更漏れは検出できない．

(2) ベースソフトの変更漏れに起因する欠陥の事例

我々のプロジェクトでは，XDDP と影響波及パス分析法を活用し，変更箇所特定漏れを防止する取り組みを実施している．この取り組みの結果，A)と B)に分類される変更箇所については，変更漏れを防止できた．しかし，C)の「設計制約の変化から影響を受ける箇所」については，変更箇所特定漏れにより，欠陥が流出した．この欠陥の事例を表 1 に示す．

表 1 「設計制約の変化から影響を受ける箇所」の変更漏れに起因する欠陥事例

事例 No	変更仕様概要	欠陥内容	ベースソフトの設計	ベースソフトの設計制約	
				仕様変更前	仕様変更後
1	機能実行中に，他機能の実行が要求されても，実行中機能の処理をキャンセルせず継続する	共有データの排他処理漏れ（共有データの個別データ化漏れ）	機能間で共有しているデータあり	キャンセル不可となった機能は他機能と並列実行不可能	キャンセル不可となったことで機能の並列実行が可能となる
2	バッファのサイズを縮小する	バッファオーバーフローのガード処理漏れ（一部の箇所のみ）	同じデータが格納される（格納可能データ数が同一の）複数のバッファあり	バッファオーバーフローの可能性がないようにバッファのサイズを定義	バッファサイズが小さくなったことでバッファオーバーフローの可能性あり

(3) ベースソフト調査方法の問題点

表 1 に挙げた欠陥の原因は，ベースソフト調査方法にあると考えた．自プロジェクトのベースソフト調査方法を調べ，問題点を明らかにした．

派生開発におけるベースソフト調査は，「事前調査」「変更箇所調査」「影響調査」の 3 つの調査ステージに区分される[4]．今回の問題は，ベースソフトを理解するための「事前調査」と，変更要求を実現するために必要な変更点を特定するための「変更箇所調査」にあることがわかった.「事前調査」では，機能とデータ関係の把握漏れが問題であった.「変更箇所調査」では，設計制約の変化から影響を受ける箇所の抽出漏れが問題であった．

- 機能とデータ関係の把握漏れ
  大規模なソースコードを処理と機能の関係を人手のみで理解するには限界がある．データを参照する関数は複数あり，その関数を呼び出す関数も複数ある．データ数に対して，関係を把握すべき関数数は指数関数的に増えるため，人手でベースソフト全体の機能とデータの関係を完全に把握をするには，多くの工数が必要となる．そのため,短期間での開発が求められる場合には,調査する範囲を絞り込んでいる．絞り込む調査範囲の判断を誤れば，把握漏れが発生する．
- 設計制約の変化による影響箇所の抽出漏れ
  変更の要否を判断する対象となる影響箇所の抽出が漏れていた．設計制約の変化による影響箇所は，担当者の知識・経験をもとに抽出している．入力としている知識が不足していれば，影響箇所の抽出漏れが発生する．また，調査結果のレビューでは，抽出された影響箇所に対して問題を検出することはできても，影響箇所が抽出できていないことを問題として検出することは困難である．

2.2 課題提起

本研究では,「設計制約の変化から影響を受ける箇所」を対象に，変更箇所の特定漏れ防止を目的として，担当者の能力・知識に依存しないベースソフト調査手法を開発する．

開発した調査手法が,「事前調査における機能とデータ関係の把握漏れ」と「設計制約の変化による影響箇所の抽出漏れ」の 2 つの問題を解決することを確認する.

2.3 先行研究

(1) XDDP への DRBFM の導入

安田[5]により，変更要求仕様書（USDM）を活用し，DRFBFM を取り入れた未然防止手法が提案されている．変更要求から心配点を漏れなく抽出するために，機能と過去の不具合情報（心配点）の関係を入力として，従来の DRBFM では発見できなかった心配点や原因が抽出できた成果が報告されている．

本手法を参考として，開発手法では，変更要求から発生し得る欠陥とその要因を抽出し，ベースソフトに欠陥の要因となる個所があるかを確認し，変更が必要となる箇所を特定するアプローチを適用する．

(2) ソフトウェア欠陥予測アルゴリズム

SQiP 研究会第 7 分科会[6]により，欠陥混入メカニズムを表現・蓄積・利用する手法が提案されている．この手法は,「失敗の知識を活用するには失敗のメカニズムを示す要因と結果の要素が必要」という考え方をもとにしている．欠陥混入メカニズムは，誘発因子・過失因子・欠陥・不具合の関係を示す．蓄積している欠陥混入メカニズムの知識から，誘発因子キーに，同一条件下で発生する可能性のある欠陥を予測することが可能となることが報告されている．

本手法を参考として，変更要求から発生し得る欠陥とその要因を抽出するために，過去の不具合情報を蓄積・活用する仕組みを構築する．

(3) CRUD マトリクスを用いたソフトウェア設計影響分析手法

加藤[7]により，ソースコードから関数と変数の依存関係に関連する情報を自動的に抽出して，CRUD マトリクスとして可視化する手法が提案されている．この手法は，グローバル変数を介した処理間の連携動作を把握し，ソフトウェアを変更した場合の影響を特定しやすくすることを課題としている．影響が波及しそうな機能を絞り込み，人手で設計書やソースコードを調査していた作業が効率化できたことが報告されている．

本手法を参考として，データと機能の関係を把握するための仕組みを開発する．

3. 解決策の提案

3.1 課題の解決方針

(1) 機能とデータ関係の把握漏れの解決方針

「機能とデータ関係の把握漏れ」については，静的コード解析を活用し，担当者の能力を補い解決する．

「機能とデータの関連」は，CRUD マトリクスを拡張・変更し表現する．関数と変数の関係のみではなく，関数コールツリーも表現し，公開 IF と変数の関係を把握可能とする．また，データアクセスの種類は，変更箇所特定の目的で必要となる W：書き込み（Write）と R：読み込み（Read）の 2 種類のみとする．

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．

(2) 設計制約の変化による影響箇所の抽出漏れの解決方針

「設計制約の変化による影響箇所の抽出漏れ」については，変更箇所が問題なく特定できているときに利用している知識を明らかにし，その知識を蓄積し，担当者の知識を補い

解決する．変更箇所が問題なく特定できているとき，開発担当者は欠陥が混入するメカニズムを理解しており，変化点から発生し得る欠陥を予測したうえで，ベースソフトを調査し，変更箇所を特定していた．このことから，欠陥が混入するメカニズムの知識を蓄積し利用することができればで，影響箇所の抽出漏れを防止できると考えた．

影響箇所の抽出漏れを防止するために蓄積が必要な知識は，「設計制約の変化から影響を受ける箇所」が特定できているときのベースソフト調査手順から明らかにした．変更箇所が特定できているとき，開発担当者は，表 2 の手順で調査をしていた．

表 2 変更箇所を特定しているときのベースソフト調査手順

No	手順	入力	出力
1	変更要求をもとに，変化する設計制約を特定する	・変更要求	・設計制約の変化
2	設計制約が変化する場合，その変化に伴い発生の可能性のある欠陥を予測する	・設計制約の変化	・発生し得る欠陥
3	予測した欠陥は，ベースソフトの設計がどのような場合に混入するか特定する	・発生し得る欠陥	・欠陥混入に繋がる設計
4	ベースソフトの設計を把握するために利用する設計の表現技法（例：シーケンス，DFD，フローチャート等）を決める	・欠陥混入に繋がる設計	・設計把握のための表現技法
5	4 で決めた技法でベースソフトの設計を表現する	・設計把握のための表現技法	・ベースソフトの設計
6	5 で作成した成果物を活用し，ベースソフトの設計を把握し，予想した欠陥が発生し得る影響箇所を絞り込む	・ベースソフトの設計 ・発生し得る欠陥	・影響箇所
7	6 で絞り込んだ影響箇所に対して，ソースコードの変更の要否を判断する	・影響箇所 ・発生し得る欠陥	・変更箇所

表 2 の手順から，影響箇所の抽出には，「設計制約の変化」と「ベースソフトの設計」を要因として発生する「欠陥」の関係を示した情報が必要となることがわかった．本稿では，図 1 に示すように，変更箇所の特定漏れを引き起こすトリガとなる「設計制約の変化」「ベースソフトの設計」と，変更箇所の特定漏れにより引き起こされる「欠陥」の関係を欠陥混入メカニズムと呼ぶ．この欠陥混入メカニズムの知識を入力とすることで，担当者の知識に依存しない調査が可能となると考えた．

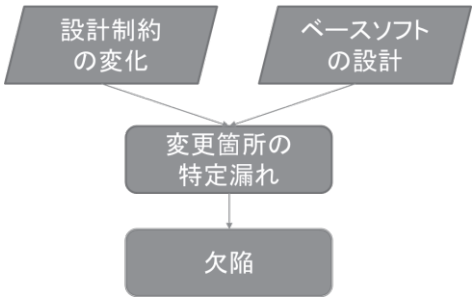

図 1 欠陥混入メカニズム

3.2 欠陥混入メカニズムをもとにしたベースソフト調査手法の定義

提案手法は，「データと機能の関係情報」と「欠陥混入メカニズムの知識」を利用するベースソフト調査手法である．今回の手法は開発担当者が実施することを前提としている．

図 2 にベースソフト調査の手順と利用する「データと機能の関係情報」と「欠陥混入メカニズムの知識」の関係を示す．「欠陥混入メカニズムの知識」は，以下の作業で利用する．

・変更要求から設計制約の変化を特定する
・設計制約の変化から発生し得る欠陥を予測する
・欠陥を顕在化させる可能性のある箇所（影響箇所）を抽出する

また，ソースコード解析により「データと機能の関係情報」を確実に把握し，影響箇所と変更箇所を抽出するための入力とする．

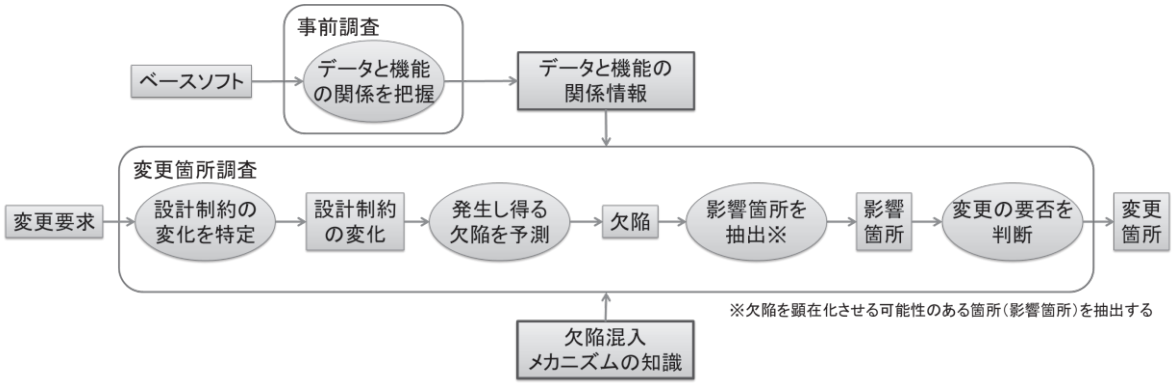

図 2 欠陥混入メカニズムをもとにしたベースソフト調査手順

本手法は，調査の入力とする「データと機能の関係情報」と「欠陥混入メカニズムの知識」の表現方法を技術要素とする．

(1) データと機能の関係情報の表現方法

「データと機能の関係」を表現した成果物は，CTD マトリクス（Call tree with data マトリクス）と呼ぶ．CTD マトリクスのイメージを図 3 に示す．

機能名	関数コールツリー	外部変数		
		外部変数1	外部変数2	外部変数3
A機能	FuncA	-	-	-
	FuncB	-	-	WR
	FuncC	W	-	-
	FuncD	-	-	-
	FuncE	-	W	-
B機能	FuncF	-	-	-
	FuncG	-	-	-
	FuncH	-	-	-
	FuncI	R	R	-

図 3 CTD マトリクス

CTD マトリクスは，関数コールツリーと外部変数を 2 軸にもつマトリクスである．外部変数へのアクセス仕様を W：書き込み（Write）と R：読み込み（Read）の 2 種類で表現し，対象となる関数と外部変数の交点箇所に記載する．

CTD マトリクスは，ソースコードに加えて，表 3 に示した設計情報を入力として静的コード解析により作成する．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．

表 3 CTD マトリクス作成に必要な設計情報

名前	説明	イメージ
関数呼び出し関係リスト	呼出し元関数と呼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出し関係のみ示す．	呼出し元関数 / 呼出し先関数 FuncA / FuncB FuncA / FuncC FuncB / FuncC FuncD / FuncE
データ-アクセス関数マトリクス	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．	データアクセス関数 / データ（DataA, DataB, DataC） FuncA: WR, , R FuncB: , W, FuncC: R, , W FuncD: , R, ※W･･･データの更新処理あり ※R･･･データの参照処理あり

(2) 欠陥混入メカニズムの知識の表現方法

欠陥混入メカニズムの知識は，表 4 に示した要素で表現する．欠陥混入メカニズムの知

識を蓄積した成果物を，DIM 辞書（Defect injection mechanism 辞書）と呼ぶ．

表 4 欠陥混入メカニズムの知識に含める要素

要素	説明
設計制約の変化	ベースソフトの設計制約の変化を示す．欠陥混入メカニズムの知識を検索するためのキーとなる情報である．
欠陥	設計制約の変化により，引き起こされる可能性のある欠陥を示す．ソースコードの欠陥を表現する．
ベースソフトの設計	欠陥の要因となるベースソフトの設計を示す．予想した欠陥が発生し得る箇所を絞り込むために利用する知識である． 変更箇所特定時に注目すべき点を示す．CTD マトリクスから特定すべき「複数機能からアクセスされる外部変数」「保持されているユニットとは別ユニットから書き込みがされる外部変数」等を示す．
設計の表現技法	ベースソフトの設計を把握するために利用する表現技法（例：シーケンス，DFD，フローチャート等）を示す．CTD マトリクス以外に利用すべき技法を示す．
ベースソフトの調査手順	予想した欠陥が発生し得る箇所を絞り込み，ソースコードの変更要否を判断する手順を示す．

4. 解決策の評価

4.1 評価方法

図 4 に示した DIM 辞書を準備し，CTD マトリクスの生成ツールを開発したうえで，本手法を適用し，評価を実施した．評価観点と評価方法は，以下の 2 点である．

- DIM 辞書を使用し「設計制約の変化による影響箇所」の抽出が可能か？？
  開発経験（所属プロジェクト，開発経験年数）の異なる複数の技術者（対象者：5 人）に対して，アンケート方式で確認を実施する．DIM 辞書の情報を入力に，「発生し得る欠陥が予測」と「欠陥を顕在化させる設計の特定」が可能かを質問した．
- 過去に発生した変更箇所の特定漏れを防止することが可能か？
  DIM 辞書を入力に，ベースソフト調査の再実施を行い，変更箇所が特定できるかを確認した．変更箇所特定漏れが発生した 2 件の事例を対象とした．

No	設計制約の変化	欠陥	ベースソフトの設計	設計の表現技法	ベースソフトの調査手順
1	Before) 機能が並列実行不可能 After) 機能が並列実行可能	共有データの排他処理漏れ	・機能間で共有しているデータ（外部変数）あり	-	1. 並行実行可能となった機能で、共有している外部変数を特定する 2. 1で特定した外部変数に対して、排他処理の追加要否を判断する
2	Before) オーバーランが発生する可能性のないバッファサイズとする After) バッファサイズの最小値の制約なし	バッファオーバーランのガード処理漏れ	・同じ種類のデータが格納されるバッファ(配列)が複数あり ・バッファ(配列)が保持されているユニットとは別ユニットから書き込みがされる箇所あり	・DFD	1. バッファオーバーランの可能性が発生したバッファを特定する 2. 1で特定したバッファを起点としてDFDを作成する 3. 2で作成したDFDから関連するバッファを特定する 4. 3で特定したバッファに対して、オーバーランのガード処理の追加要否を判断する

図 4 DIM 辞書

4.2 評価結果

各評価観点に対して，評価結果を以下に示す．

- DIM 辞書を使用し「設計制約の変化による影響箇所」の抽出が可能か？
  「発生し得る欠陥が予測」と「欠陥を顕在化させる設計の特定」が可能であると回答した技術者は 4 名，不可能と回答した技術者は 1 名という結果であった．不可能と回答した技術者は，ソフトウェア開発経験 1 年以下のメンバである．
- 過去に発生した変更箇所の特定漏れを防止することが可能か？
  ベースソフト調査を再実施した結果，2 件の事例共に，「設計制約の変化から影響を受ける箇所」を特定できた．

4.3　結果の考察

提案手法により，担当者の能力・知識を補い，設計制約の変化による影響箇所の抽出が可能となった．DIM 辞書に欠陥混入メカニズムの知識を蓄積していくことで，変更漏れを防止できるケースが増え，欠陥の流出数が削減できると考えられる．

ただし，提案手法により，完全に担当者の能力・知識・経験に依存しない調査が可能となったとは言い切れない．DIM 辞書に示されている欠陥を経験していない技術者の場合は，ベースソフトの調査内容を決めることができない．提案手法による効果を得るには，DIM 辞書の内容を理解できていることが条件であることがわかった．

5.　おわりに

5.1　研究の成果

派生開発において，変更要求に対応することで設計制約が変化することがある．この設計制約の変化から影響を受ける箇所で発生する，変更漏れが防止できていないことがわかった．ベースソフト調査方法に「機能とデータ関係の把握漏れ」「設計制約の変化による影響の特定漏れ」という 2 つの問題があり，変更漏れに繋がっていることを明らかにした．

本研究では，この問題を解決するために，「データと機能の関係情報」と「欠陥混入メカニズムの知識」を利用したベースソフト調査手法を開発した．そして，本手法の技術要素として，「CTD マトリクス」と「DIM 辞書」の 2 つを定義した．

このベースソフト調査手法を適用することで，過去に発生した変更箇所の特定漏れが再現しないことを確認した．XDDP と影響波及パス分析法に加えて本手法を実開発に適用することで，派生開発における変更漏れ確実に防止する効果が期待できる．更に，本手法によるベースソフト調査作業を自動化することで，派生開発に必要となる工数を削減し，新たな価値を生み出す新規開発に技術者の工数を投入することが可能となる．

5.2　今後の進め方

過去のレビュー指摘，DRBFM 結果，テストでの検出欠陥等の情報をもとに，欠陥混入メカニズムの知識を蓄積するプロセスを考案する．考案した方法で DIM 辞書を拡充したうえで，以下の評価を行い，手法を改善する．

- DIM 辞書の情報から，設計制約の変化を特定可能となるか確認する．
- 異なる複数の製品開発において，欠陥混入メカニズムの知識を利用可能か確認する．

更に，組織全体への手法の展開・導入を可能とするために，「DIM 辞書」を利用したベースソフト調査プロセスを定義し，「DIM 辞書」を共有するための教育の実施を検討する．

参考文献

[1]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 Tokyo，2009

[2]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

[3]柏原一雄，「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」，ソフトウェア品質シンポジウム 2017，2017

[4] SQiP 研究会第 6 分科会（A グループ），「変更の影響範囲を特定するための「標準調査プロセス」の提案」，ソフトウェア品質管理研究会 第 30 年度分科会成果報告，2015

[5]安田隆司，「変更要求仕様書を活用した未然防止方法の提案-XDDP への DRBFM 導入とその効果-」，ソフトウェア品質シンポジウム 2011，2011

[6]2014 年度 SQiP 研究会第 7 分科会，「ソフトウェア欠陥予測アルゴリズム-欠陥混入メカニズムのモデリング手法を利用した欠陥予測方法の提案-」，ソフトウェア品質シンポジウム 2015，2015

[7]加藤正恭，小川秀人，「CRUD マトリクスを用いたソフトウェア設計影響分析手法」，情報処理学会全国大会講演論文集，巻：73rd，号：1，ページ：1.249-1.250，2011