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～

Software defect prediction algorithm

- Proposed method of predicting defects using "Defect Injection mechanism" modeling -

主査	：細川 宣啓	日本アイ・ビー・エム(株)
副主査	：永田 敦	ソニー(株)
リーダー	：柏原 一雄	(株)デンソークリエイト
研究員	：岡本 晃	農中情報システム(株)
	鈴木 裕一郎	(株)日立製作所
	田村 光義	サイバートラスト(株)
	東久保 理江子	アンリツ(株)
	保栖 真輝	日本電子(株)

研究概要

ソフトウェア開発における欠陥の情報は，多くの組織で蓄積されている．混入し得る欠陥を予測し，欠陥の混入・流出を防止するためには，この蓄積された欠陥情報を活用することが不可欠である．本研究では，欠陥の混入を予測するためには，欠陥が混入するメカニズムを理解し，欠陥を引き起こす要因を検知する必要があるということに着目し，その方法を検討した．本稿では，欠陥の予測を目的に，欠陥混入メカニズムのモデリング手法と欠陥データベースの構築・利用手法を提案し，有効性を確認する．

Abstract

Software defect information that harvested from several software development projects is one of the assets to share. Software defect information is indispensable information for the software defect prediction / prevention activity. In this research activity, we set the focus onto that to understand and to predict the defect injection, we have to establish the way to represent the "Defect Injection mechanism", furthermore, defect management process (modeling, store and share) have to be defined. In this paper, we propose "Defect Injection Mechanism" modeling and the method for establish the defect management database for the software defect prediction.

1　はじめに

ソフトウェア開発において，各組織では膨大な欠陥情報が蓄積され，ビッグデータ化している．また，人は同じ過ちを繰り返すため，ソフトウェア開発で発生している不具合は，同じメカニズムで繰り返し発生するものが多い．欠陥の混入・流出を防止し，品質向上に繋げるためには，この蓄積した欠陥情報をいかに有効活用するかが鍵となる．

研究員が所属している組織でも，欠陥情報は蓄積されている．しかし，情報を蓄積するだけで，欠陥の混入・流出を防止するために，有効活用はできていない．我々は，欠陥の混入・流出を防止するために，蓄積した欠陥情報を利用する方法を研究の対象とした．

欠陥の混入・流出を防止するためには，まず対策の対象とする欠陥を特定しなければならない．また，過去の欠陥情報をもとに，同一条件下で混入し得る欠陥を予測する方法が必要となる．混入し得る欠陥を予測するためには，混入した欠陥とそれによって引き起こされる不具合の情報だけでなく，欠陥混入の要因・背景等も含めた欠陥混入に至るメカニズムを示す情報が必要となる．欠陥混入メカニズムが表現され，その情報が整理され蓄積されていることで，欠陥の混入要因から発生する欠陥を予測することが可能となる．

本研究では，混入し得る欠陥を予測するためには，欠陥混入メカニズムが理解できるよ

うに表現され，その情報を資産として蓄積し，利用できるようにする必要があると考え，手法を検討した．

本稿では，ソフトウェア開発において，混入し得る欠陥を予測することを目的に，欠陥混入メカニズムのモデリング手法と欠陥データベースの構築・利用手法を提案する．提案手法をまとめ，欠陥予測アルゴリズムと呼ぶ．

提案手法の有効性を確認するために，実験では以下を確認した．

- 欠陥混入メカニズムが理解・同意できるように表現される
- 欠陥混入メカニズムの情報をもとに，発生し得る誤り・欠陥を予測できる

本研究により，以下の貢献が期待できる．

- テストやレビューの観点抽出のために利用することで，限られた工数で混入の可能性のある欠陥をピンポイントで検出できる．
- 欠陥混入メカニズムを理解し，知識として利用することで，プロジェクト開始前に，欠陥を混入させる要因を除去することが可能となり，混入する欠陥数を減らし品質向上とコスト削減に貢献できる．
- 欠陥情報を欠陥混入メカニズムという抽象化された知識にすることで，ソフトウェア業界全体で資産として共有できる．

本稿の以降の構成は次の通りである．

以降，2 章で解決する課題を示し，3 章で課題解決のアイデアを得た関連研究を示す．4 章で提案手法の詳細を説明し，5 章で提案手法の有効性を確認した実験の結果と考察を示す．6 章で，まとめと今後の展望を示す．

本稿では，不具合，欠陥，誤りという言葉を表 1 のように定義し，使い分ける．

表 1. 用語と意味

用語	意味
不具合	ソフトウェアが期待結果を果たせていない状態.
欠陥	仕様書やソースコードなどのプロダクトに含まれる不正確な記述箇所.
誤り	不正確な結果を生み出した人間の行為.

2　背景

2.1　欠陥の予測方法

本研究における欠陥予測は，「プロジェクトマネージャや SQA が，プロジェクトの入力となる情報（要求，入力成果物の特徴，ベースソフトの特徴，プロジェクトメンバの特徴等）をもとに，過去の欠陥情報から，同一条件下で発生する可能性のある誤りと混入する可能性のある欠陥を特定すること」と定義する．

2.2　欠陥の予測に必要なこと

欠陥は人間の誤り（過失因子）によって生み出される．人間が誤りを犯したときには，その誤りを引き起こした要因（誘発因子）が存在する．この誘発因子の存在を検知することで，誘発因子により発生する過失因子と，過失因子により混入する欠陥を予測することが可能となる．

欠陥を予測するためには，不具合と欠陥を示すだけでなく，欠陥を混入させる過失因子，人間の誤り引き起こす誘発因子も示す必要があると考えた．この不具合・欠陥・過失因子・誘発因子の関係を欠陥混入メカニズムと呼ぶ．

本研究では，欠陥混入メカニズムをどのように表現，蓄積するかを検討した．情報を表現しなければ，蓄積もできないため，特に重要となるのが，表現の方法である．

『失敗知識データベース構築の試み』[2]でも，潜在的な失敗の予測などに失敗知識を活用するためには，最低限，失敗のメカニズムを示す制約・要因・特性・結果の要素は必要であると考えられている．

2.3　解決する問題

研究員が所属している組織では，不具合を解決まで管理し，教訓として共有するために，不具合の一覧（不具合 DB）が存在している．しかし，不具合 DB に蓄積している情報をもとに，欠陥の予測をすることはできていない．

不具合 DB が以下の状態になっていることが多く，欠陥混入メカニズム表現する情報が理解できない，また欠陥混入予測のために利用しにくい．

・欠陥混入メカニズムが理解できない
　表現が一般化されていないため，不具合の未体験者は，内容を理解できない．また，内容が理解できないため，「自分には関係ない」と判断してしまい，予測のために情報を利用しない．
　また，同一条件下で発生した欠陥を特定するときに必要となる欠陥混入の条件（原因・背景）が記録されていないことが多い．具体的には，欠陥を混入させた過失因子，人間の誤り引き起こす誘発因子が記録されていない．

・欠陥 DB が欠陥混入予測のために利用しにくい
　不具合，欠陥，過失因子，誘発因子が区別し表現されていない．そのため，誘発因子が存在しているかを判断することや，過失因子，欠陥を抽出することが難しい
　また，欠陥を予測する手順は確立されていない．欠陥情報は増加する一方であり，多数の欠陥情報から効率よく必要な情報を抽出できるデータベースとその利用手順が必要．

この問題の主な原因は，欠陥情報を利用し欠陥を予測するという用途が考慮されてことである．想定していない用途で必要となる情報は，表現・蓄積されない．

不具合 DB に表現・蓄積されている情報は，主に以下を目的に，必要な情報のみ記載されていることが多く，不具合の対策が終わった後は，ほとんど利用されない．

・プロジェクトマネージャによる不具合対応の進捗監視
・経営者，プロジェクトマネージャへの対策結果の報告

目的や利用方法が変われば，表現・蓄積する必要のある情報も変わる．欠陥の予測を可能とするためには，過去の欠陥情報を利用し欠陥を予測する方法を明確にした上で，必要となる情報を明らかにする必要がある．

2.4　研究課題

本研究では，欠陥の混入予測を目的に，欠陥混入メカニズムモデリング手法と欠陥データベースの構築・利用手法を開発し，以下の 2 点の問題が解決されることを確認する．

・欠陥混入メカニズムが理解できない
・不具合 DB が欠陥混入予測のために利用しにくい

3　欠陥混入メカニズムの表現手法

本章では，欠陥混入メカニズムの表現手法に関してアイデアを得た関連研究を示す．

3.1　欠陥モデリング

Project Fabre[1]は，欠陥の情報はモデリングによって欠陥の可視化・抽象化が可能となり，共有・移転が容易になると提唱している．

また，欠陥のモデリング手法が提案されている．提案された欠陥のモデリング手法では，過失（人間の思考や判断の誤り）に着目し，欠陥情報を汎化し構造化し表現することを提案している．

欠陥混入メカニズムに必要な要素を漏れなく表現し，理解性を向上させるために，今回提案では，このモデリング手法をもとに，表記ルールを明確化し，記述ステップを定義した．

3.2　特性要因図と 5M1E

ソフトウェア開発において過失を誘発する要因は複数あり，欠陥混入メカニズムを表現

する場合，要因を系統的に（漏れなく，重複なく，矛盾なく）示す必要がある．

今回提案する欠陥混入メカニズムのモデリング手法では，特定の結果と要因系との関係を系統的に示す特性要因図[4]の表現方法と要因の分類方法を参考にしている．要因の分類方法としては，5M1E（Man(人),Method(方法),Measurement(測定),Material(原料・材料),Machine(機械・設備)，Environment(環境)）を利用している．

4　提案

本章では欠陥予測アルゴリズムを説明する．欠陥予測アルゴリズムの技術的要素は「欠陥混入メカニズムモデリングの手法」と「欠陥予測 DB の構造」,「過失・欠陥予測手順」の３つである．

4.1　欠陥混入メカニズムのモデリング手法

ソフトウェア欠陥混入メカニズム表記ルールとソフトウェア欠陥混入メカニズムの記述ステップを説明する．

4.1.1　欠陥混入メカニズム表記ルール

欠陥混入メカニズム表記ルールとして，ノードの種類とノードの繋ぎ方を示す．

(1)ノードの種類

ソフトウェア欠陥混入メカニズム表記法では，発生し得る「誤り」と混入し得る「欠陥」を予測するために，表 2 のノードにより，「不具合」「欠陥」「過失因子」「誘発因子」の関係を示す．特に欠陥の作り込みに着目するため,「過失因子」は欠陥を見逃し流出させたレビューやテストの過失は除き，欠陥を混入させた過失のみ示すことにした．

表 2. ノードの種類

名称	ノード	解説
不具合	不具合 (Incident)	ソフトウェアの振舞いが正しくないという現象のこと． 欠陥によって引き起こされる．
欠陥	欠陥 (Defect)	不具合をもたらした原因である成果物の記述のこと． 人間の誤りによって混入，流出する．
過失因子	過失因子 (Negligence Factor)	人間の思考や判断の誤りそのもののこと． 欠陥を混入させた誤り． 欠陥を見逃し流出させたレビューやテストにおける誤りではない．
誘発因子	誘発因子 (Induction Trigger)	欠陥を混入させた誤りを誘発する“トリガー”となる要素のこと． 誘発因子は，Man(人),Method(方法),Measurement(測定),Material(原料・材料),Machine(機械・設備),Environment(環境)に分類できる．

(2)ノードの繋ぎ方

ノードの繋ぎ方のルールを示す．メカニズムを示す主要な要因に絞ることによって，人に伝えやすいと考え，この表記法としている．

- 「欠陥」と「不具合」は 1 対 1 で繋げる．
- 「欠陥」と「過失因子」は 1 対 1 で繋げる．
- 1 つの「過失因子」と 1 つ以上の「誘発因子」を繋げる．

図 1 に上記ルールで記載した欠陥モデルの例を示す．

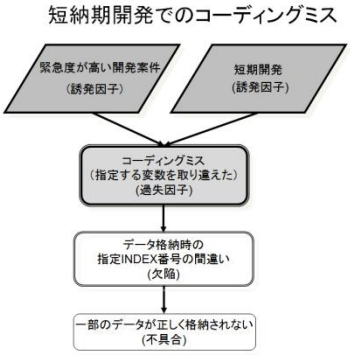


図 1. 欠陥混入メカニズムの一例

4.1.2 欠陥混入メカニズムの記述ステップ

(1)モデルの記述ステップ

1. 欠陥情報をもとに，欠陥混入メカニズム表記ルールに従い，「不具合」「欠陥」「過失因子」「誘発因子」の関係を示す．
2. 誘発因子を，5M1E の観点で系統的に（漏れなく，重複なく，矛盾なく）示されているか確認する．
3. 欠陥混入メカニズムを共有するため，業務や製品固有の言葉を避け，一般化して表現する．
4. モデリングした欠陥混入メカニズムをレビューする．

(2)欠陥混入メカニズムのレビュー観点

モデリングした欠陥混入メカニズムは，レビューによる妥当性確認を行う．レビューの観点を以下に示す．

- 欠陥の未体験者が理解できるか？（理解容易性）
  欠陥混入メカニズムの表現が一般化されていることを確認する．
- 欠陥混入メカニズムに納得（同意）できるか？（同意性）
  過失因子に対して必要十分な誘発因子が挙げられていることを確認する．欠陥混入メカニズムに納得がいかない場合には，客観的分析が不足し必要な誘発因子が欠けていると考え，再度分析する．
- 開発開始時に該当／非該当の誘発因子を選択可能か？（該当性）
  誘発因子を欠陥予測 DB の検索条件として利用可能かを確認する．該当／非該当の判断が難しい場合は，抽象的な表現になり過ぎている可能性がある．または，そもそも誘発因子としている要因が不適切な可能性もある．

上記の観点での確認を確実に行うために，不具合発生にかかわった当事者のみでは，レビューをしないことを推奨する．

4.2　欠陥予測 DB の構造

欠陥予測 DB の構造を定義する．欠陥予測 DB には，第 4.1 章で述べた 4 つの要素「誘因子」，「過失因子」，「欠陥」，「不具合」の関係を蓄積する．

欠陥予測 DB 構築のポイントは 2 つある．

1 つ目は，既存の不具合 DB からの移行・拡張を容易にするために，予測のために必要最低限のデータのみを蓄積することである．

2 つ目は，検索を容易にするために，「誘発因子」をグルーピングすることである．

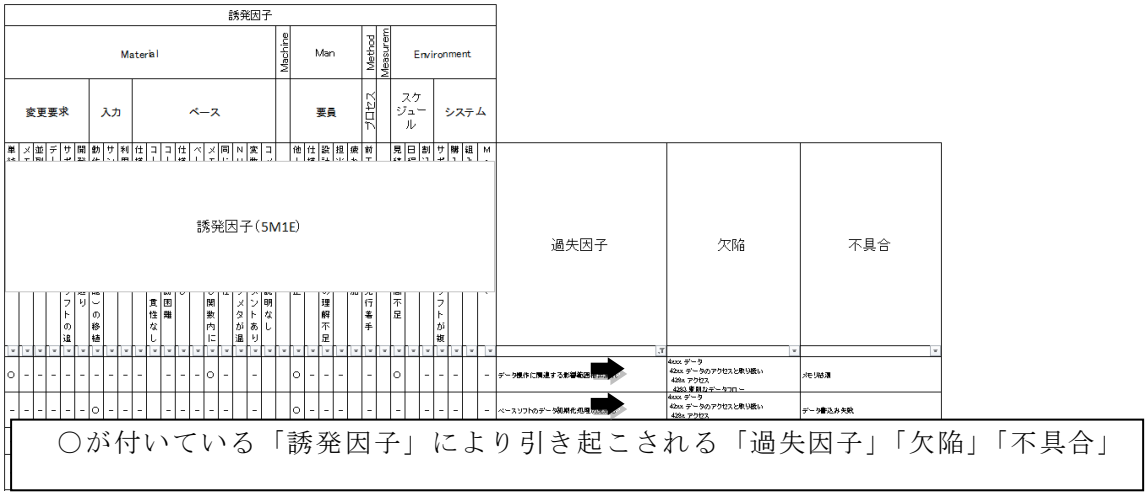


図 2. 欠陥予測 DB の一例

図 2 に，構築した欠陥予測 DB の一例を示す．

グルーピングした「誘発因子」を列に並べ，「過失因子」・「欠陥」・「不具合」を行に並べて，「誘発因子」から「過失因子」を引き起こした関連性を「○」で表記する．

4.3　過失・欠陥予測手順

欠陥予測 DB を構築し，現在のプロジェクトの状況をもとに，欠陥混入メカニズムを検索することで，起こり得る過失・欠陥を予測可能とする．過失・欠陥を予測する手順のイメージを図 3 に示す．

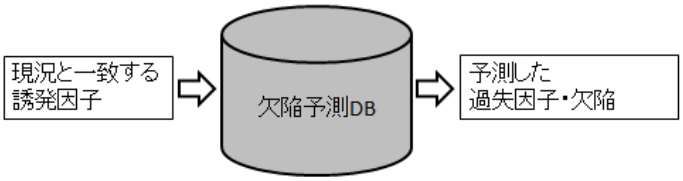


図 3．過失・欠陥を予測する手順のイメージ

DB では過去の誘発因子と過失および欠陥の関係性が蓄積されている．同一の誘発因子によって発生した過失・欠陥が検索可能であり，同一状況下で発生し得る過失・欠陥が予測できる．

以下に現在の状況下で混入し得る欠陥を予測する手順を述べる．

1. プロジェクトの状態および環境などを調査する
2. 調査結果をもとに，欠陥予測 DB のすべての「誘発因子」に対して該当／非該当を判断する
3. 該当しない誘発因子が含まれている欠陥混入メカニズムを発生候補から外す
4. 発生候補として残った欠陥混入メカニズムから，「過失」と「欠陥」を抽出する
5. 抽出された「過失」と「欠陥」が，発生する可能性がある

5　実験

5.1　実験内容

研究員が所属する組織で実際に発生した欠陥情報を用いて，4 章で提案した手法の有効性を以下の 2 つの実験で確認した．

【実験 1】欠陥混入メカニズムが理解・同意できるように表現されることを確認する．

【実験 2】「誘発因子」によって「過失因子」を予測できることを確認する．

【実験 1】では，4.1 節で述べた欠陥混入メカニズムの表記法を用いて記述したモデルを作成し，初めてモデルを見た人が事前説明なしに理解・同意を得られるか以下の選択肢を用いたアンケート方式で確認を行った．

○：全部理解できる・全部同意できる

△：一部の内容が理解できない・一部論理に同意できない

×：全部理解できない・全部同意できない

なお，2.1 節で定義している通り，欠陥予測の対象者は「プロジェクトマネージャや SQA」であるため，アンケートの対象はプロジェクトマネージャと設定した．また，「○」と回答しない場合はその理由をコメントしてもらった．

【実験 2】では，4.2 節で述べた欠陥予測 DB を構築し，4.3 節で提案した欠陥予測のアルゴリズムを用いて，プロジェクトで発生する「過失因子」を予測できるか確認した．今回は研究員が所属する各組織の終了したプロジェクトで発生した「欠陥」を確認対象とした．なお，欠陥予測 DB を構築する際に使用した欠陥情報は確認対象に含まれないよう留意した．欠陥予測 DB に登録した「欠陥」は 44 件であった．また，本来「過失因子」によって混入される「欠陥」についても確認すべきであるが，欠陥予測 DB の構築のうち，「過失因子」と「欠陥」の関係性について必要十分なデータを用意することができなかったため，

「過失因子」のみの確認とした．

5.2　実験結果

【実験 1】の確認結果を図 4 に記載する．5 つのモデルの平均値は理解度を「○」と回答した割合は 62.5%，同意度を「○」と回答した割合は 60.0%であった．特にモデル 3 およびモデル 4 については半数以下の理解度となった．アンケートの回答では理解できなかった理由として誘発因子，「過失因子」に記載している内容自体が理解できないという内容が多かった．またモデル 4 は同意度についても半数以下であった．アンケートの回答では「誘発因子」と「過失因子」の関係性について疑問を呈する内容が多かった．全体を通して「誘発因子」の過不足，「過失因子」の表現に関する不適切さを問う回答が多かった．

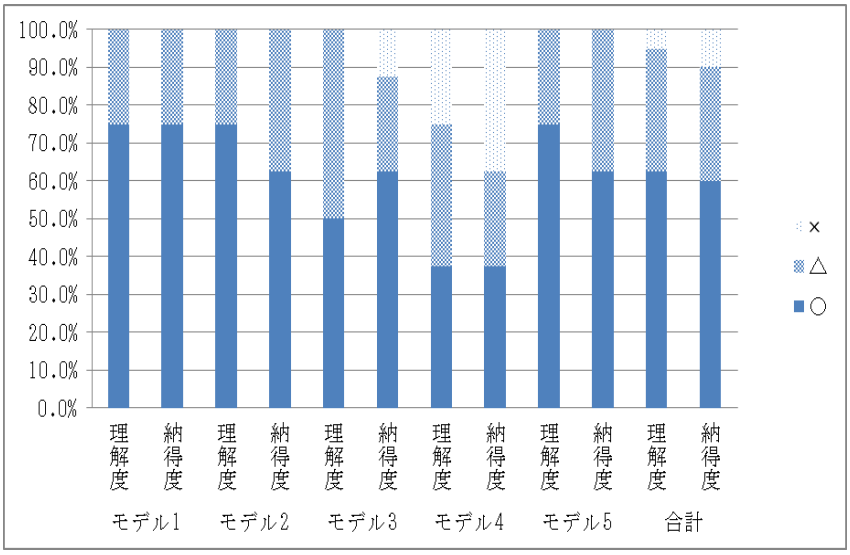


図 4.【実験 1】の結果

【実験 2】の結果を表 3 に記載する．この結果は欠陥予測 DB から導き出した過失とプロジェクトで発生した「過失因子」の一致率を表しており，全体としては 55%の確率で予測することを確認できた．このことから，そのプロジェクトにおいて発生する「過失因子」の候補を特定することが可能と判断できる．

また，一致した過失のうち多くの占めるのが「影響範囲抽出漏れ」，「コーディング漏れ」，「タイプミス」であった．

表 3.　【実験 2】の結果

プロジェクト	発生した欠陥数	予測で一致した過失因子数	一致率
A	28	11	39.3%
B	24	13	54.2%
C	52	36	69.2%
D	24	15	62.5%
E	43	30	69.8%
F	6	2	33.3%
G	6	4	66.7%
H	19	5	26.3%
I	16	6	37.5%
J	19	10	52.6%
合計	237	132	55.7%

5.3　考察

【実験 1】の結果から欠陥自体の説明をしなくても，4.1 節で述べた欠陥混入メカニズムの表記法を用いることで約 6 割の欠陥混入に関する理解･同意を得ることが確認できた．この結果より，従来は口頭または文章によって伝えられていた欠陥の情報が，混入の経緯まで含めたモデリングとして伝えられるようになった．従来と比較すると，このモデリン

グは情報量を抑え図式化することにより，初見で欠陥混入メカニズムを直感的に把握することが可能となった．このように欠陥を伝承しやすくなったことは，欠陥を理解することによって有効な改善活動へつながる大きな進歩であるといえる．

理解を得られなかったモデルについては，各因子に記載している表記方法が理解されていないこと，「欠陥」と「過失因子」の区別が明確でないことが理由として考えられる．

同意を得られないモデルについては，「誘発因子」と「過失因子」の関係性に対する同意が得られていないことに起因している．「誘発因子」と「過失因子」の関係性について実験のために用意した個々のモデルに対しての検証が不十分であると考えられ，モデル自体が欠陥混入メカニズムを適切に表現できていない可能性がある．

そのため，欠陥混入メカニズムのモデリング手法を用いる際に，誰が・どの欠陥を表現する場合でも同程度の質で記載できるルールと，表記したモデルの検証手法の確立は今後の課題である．

【実験 2】の結果から，提案した欠陥予測 DB の構造と過失・欠陥予測手順によって，「過失因子」の予測行為が可能になったといえる．ただし，今回の実験では一致した「過失因子」のみを集計しており，「誘発因子」と「過失因子」の関係性まで含めて一致したかどうかは確認できなかった．そのため，今回の実験で示した一致率がそのまま予測できる確率と言うことはできない．しかし，たとえ完全な予測とは言えなくともプロジェクトに潜む「誘発因子」から発生するであろう「過失因子」の候補を示すことで，プロジェクトメンバ間での共通のリスク認識，設計・開発時の注意喚起およびレビュー，テストの観点抽出として活用することができるといえる．今後は今回の実験で確認できなかった「誘発因子」と「過失因子」の関係性および「過失因子」と「欠陥」の関係性に焦点をあわせた実験を行い，予測の精度や品質向上における有効性の確認を進めることが課題である．

6　おわりに

我々は過去の誘発因子と過失因子は繰り返し発生すると考え，欠陥混入メカニズムを表現・蓄積し，その情報をもとにして欠陥を予測する「欠陥予測アルゴリズム」を提案した．実験により，本手法で欠陥混入メカニズムが理解・同意できるように表現されることを確認した．また，欠陥予測アルゴリズムが，プロジェクトで発生する「過失因子」の候補を特定できることを確認した．本研究により，過去の欠陥情報をもとに，混入し得る誤り・欠陥を予測する手法を新たに確立できた．実験は異なる複数の企業で実施し，本手法はソフトウェア業界全体に適用可能と確認できた．将来，ソフトウェア業界全体で欠陥混入メカニズムの情報を蓄積し，各組織で欠陥予測に利用することも可能と言える．今後は，欠陥予測 DB の拡充，今後開発を開始するプロジェクトへの適用実験，過失因子・欠陥と関連する混入・流出防止策（分析技法，設計技法，検証技法等）の整理等を実施し，より有益な手法となるよう改善していきたい．

参考文献

[1]細川宣啓，野中誠，西康晴，原佑貴子，嬉野綾「過失に着目した欠陥のモデリング」，JaSST2013 Tokyo

[2]畑村洋太郎・中尾政之・飯野謙次「失敗知識データベース構築の試み」，2003 年

[3] Atsushi Nagata・Nobuhiro Hosokawa「IMPROVEMENT of ROOT CAUSE ANALYSIS with ITERATIVE PROCESS and DEFECT MODELLING」，2014 年

[4]石川馨「品質管理入門」日科技連出版社，1956 年

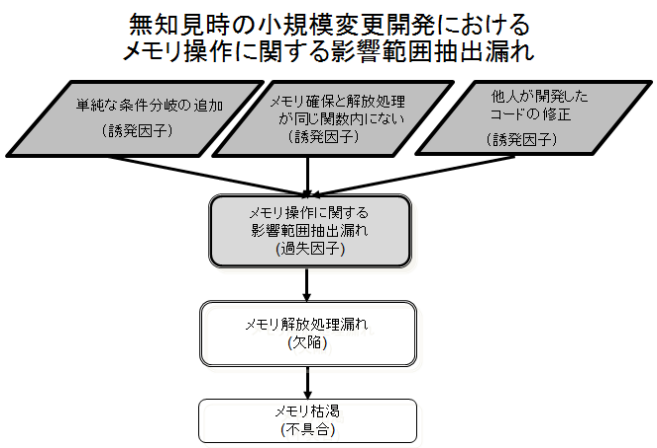
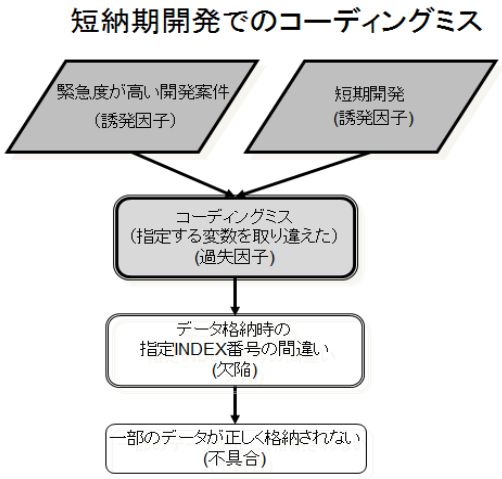
[5]ボーリスバイザー「ソフトウェアテスト技法」，日経 BP マーケティング

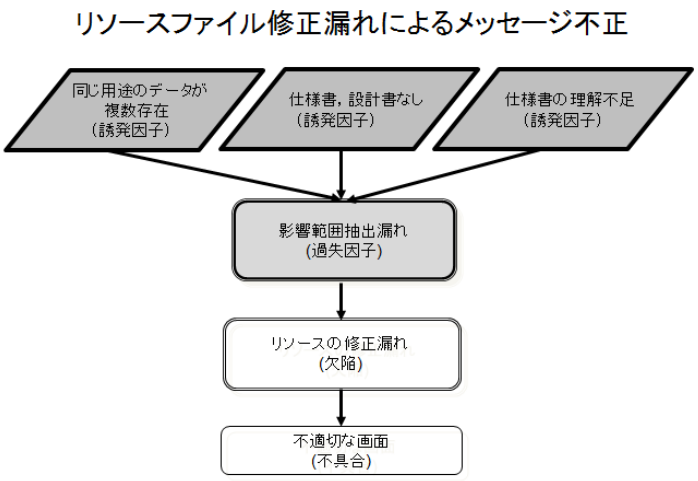
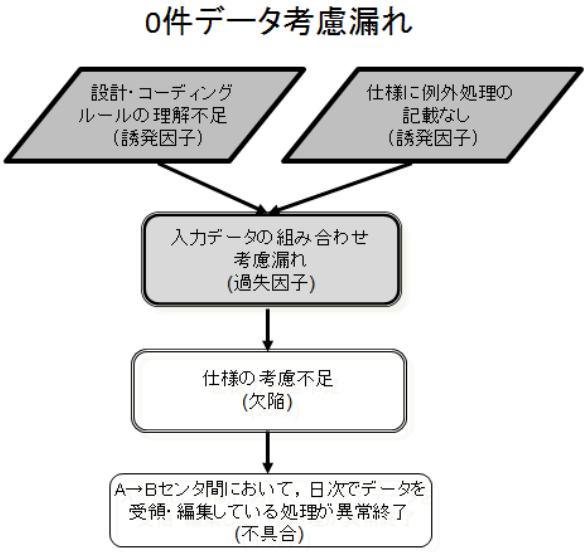
[6]野中誠，小池利和，小室睦「データ指向のソフトウェア品質マネジメント」日科技連出版社，2012 年

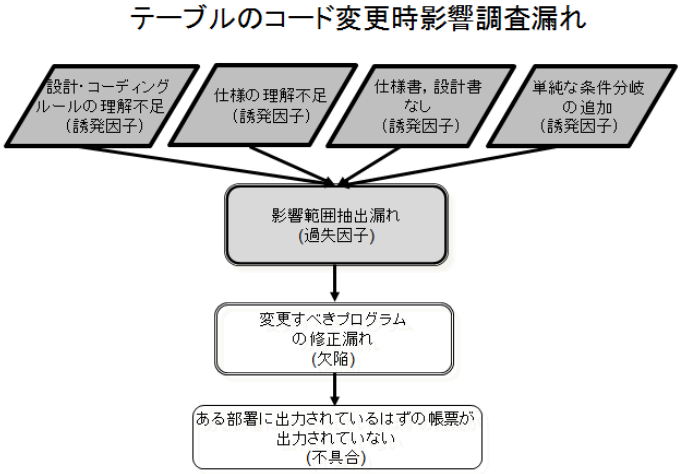
[7]IEEE610.12-1990(R2002)

付録

付録 1：【実験 1】に用いた欠陥混入モデリングとモデルに対するコメント

No	モデル	コメント
1		・誘発因子として，「設計・コーディングルールの理解不足」「仕様書・設計書なし」「理解不足」が他にもあると思う ・他にも要因がある気がする．確認漏れも，不具合流出の大きな要因ではないか？ ・過失因子と欠陥がどちらも誤りに見える． ・「単純な条件分岐の追加」が「影響範囲抽出漏れ」を誘発するのか？
2		・誘発因子として，「理解不足」も原因になるのでは．時間がなくて理解不足で修正してしまうため． ・変数の取り違えを発生させる条件が足りない気がする．「変数」と「指定 INDEX 番号」という単語が混在している．関係がわからない． ・「見積り工数に対して」という枕言葉は必要？ ・過失因子はもう少し抽象化したものだと思うので「仕様の取り違え」とかかな？また欠陥に“間違い”のような動機的な言葉ではなく「指定誤り」のような技術的な言葉の方が相応しい．

No	モデル	コメント
3		・「データ」と「リソース」という単語が混在している．関係がわからない． ・影響範囲の抽出漏れに，対象成果物の量や読みやすさは関係ないか？ ・影響範囲の抽出漏れだけが原因で，リソースの修正漏れが起きたのか？そもそも影響範囲の抽出とはどういう行為をやったのか？ ・欠陥が過失因子に見える． ・「同じ用途のデータが複数存在」が意味不明． ・「影響範囲抽出漏れ」よりは「修正範囲洗い出し漏れ」とかの方が分かりやすい？ ・「同じ用途のデータが複数存在」していても，「仕様書，設計書あり」「仕様の理解が十分」であれば，影響範囲抽出漏れは防げると思われる．
4		・「設計・コーディングルール」と「0 件データ時の動作」の関係性がわからない． ・「目次でデータを・・・」の意味がわからない． ・「設計・コーディングルールの理解不足」が，なぜ「入力データの組合わせ考慮漏れ」に繋がるのかわからない． ・欠陥の説明が曖昧． ・「仕様の考慮不足」は欠陥ではなく過失ではないか． ・誘発因子の“例外処理”がちょっとピンと来ない．逆に，インターフェース仕様の記載なしとか理解不足とかが誘発因子に出てこないだろうか？ ・過失因子の“組み合わせ”がどこから来たものかが不明． ・「仕様の考慮不足」は欠陥ではなく過失因子ではない．

No	モデル	コメント
5		・「設計・コーディングルールの理解不足」が，なぜ影響範囲の抽出漏れに繋がるかわからない． ・過失因子と欠陥がどちらも過失因子に見える． ・「影響範囲抽出漏れ」よりは「修正範囲洗い出し漏れ」とかの方が分かりやすい？

付録 2：欠陥予測 DB の誘発因子一覧

5M1E 分類	誘発因子分類	誘発因子
Material (原料･材料)	変更要求	単純な条件分岐の追加
		メモリ配置の変更
		並列処理を可能とする変更
		データ構造の変更
		サポート対象の機器・関連ソフトの追加
		開発途中での一部変更の見送り
	入力	動作実績のあるソフト(機能)の移植
		サンプルプログラムの利用
		利用実績のない OS の使用
	ベース	仕様書，設計書なし
		コードと仕様書，設計書に一貫性なし
		コードと関連する仕様が追跡困難
		仕様書に例外処理の記載なし
		ベースソフトの仕様が不統一
		メモリ確保と解放処理が同じ関数内にない
		同じ用途のデータが複数存在
		NULL の扱いが異なるパラメタが混在
		変数に誤解釈を誘発するコメントあり
		コメントにパラメタの値域説明なし
Machine (機械･設備)	機械・設備	負荷状況によりタイミングが変わる
Man (人)	要員	他人が開発したコードの修正
		仕様の理解不足
		設計・コーディングルールの理解不足
		担当の変更
		疲れ・焦りによるミスの増加
Method (方法)	プロセス	前工程が未完了で後工程に先行着手
		構成管理のルールが曖昧，または存在しない
Measurement (測定)	-	-
Environment (環境)	スケジュール	見積り工数に対して開発期間不足
		日程遅延
		割込み案件
	システム	サポート対象の機器・関連ソフトが複数
		購入ソフトとの結合
		組み込みソフト
		Manager と Agent