ソフトウェアテストにおけるバグ推測によるテスト設計手法の提案

- 過去バグ事例を活用した経験が浅いメンバーのバグ推測能力の向上 -

主査　　：奥村　有紀子（有限会社デバッグ工学研究所）
副主査　　：秋山　浩一（富士ゼロックス株式会社）
　　喜多　義弘（東京工科大学）
アドバイザー　：堀田　文明（有限会社デバッグ工学研究所）
研究員　　：川上　祐司（ブラザー工業株式会社）
　　大島　祥吾（株式会社東海理化）
　　榎本　和也（株式会社神戸製鋼所）
　　岩尾　隆弘（株式会社日立製作所）

研究概要
仕様書にしたがって作成されたテストケースによる結合テストやシステムテストが完了している前提で行なうテストにおいては，仕様書に書かれていない部分についてはバグを推測してテストすることが必要になる．ただし，テスト設計の経験が浅いテスト技術者では，仕様書に記載されている情報を基にしたテストの設計はできても，テストベースとなる仕様の情報が無い状況ではバグを推測したテストを設計することが難しい．
研究員のテスト現場では，テスト設計経験が異なるメンバーで構成されるテストチームでテストを行なうケースが多く，テスト設計の経験が浅いメンバーでもバグを推測したテスト設計ができるようにすることが課題となっている．そこで本研究では，その課題の解決策として，既存機能と過去の最終確認テストでバグを摘出した時の観点を関連付けることによりバグを推測することを基にしたテスト設計手法を考案した．その内容について報告する．

1　はじめに

研究員の職場では，仕様書に記載された通りの動作確認を行うテストケース(以下仕様書にしたがったテストケースと呼ぶ)による統合テストやシステムテストが完了している前提で，出荷前の最終確認として品質を評価する（以下，この評価のことを最終確認テストと呼ぶ)．最終確認テストで取り切れず，出荷後に露見されるバグを減少させるのが職場の課題である．
出荷後のバグを仕様書に記載があるか否かで分けると，出荷後のユーザーの使用環境や使用方法などの仕様書に記載されていない部分に関するバグが多い（研究員の１名が所属する部署の実態は参考文献[1]で報告されている)．そのため，最終確認テストでは，仕様書に記載されていない部分についてはバグを推測してテストすることが必要になる．
テスト設計の経験が浅いメンバーでも，テスト設計技法を用いることで仕様書にしたがったテストケースを作成することはできる．しかし，仕様書に記載されていない部分に関するバグを推測することが難しい．研究員のテスト現場では，テスト設計の経験量が異なるメンバーで構成されるテストチームで最終確認テストを行なうケースが多い．ソフトウェア開発において限られた開発期間の中でより多くのバグを摘出するためには，経験が浅いメンバーでもバグを推測したテスト設計ができるようにすることが課題となっている．
先行研究[1]～[4]では，過去のバグに関する情報を開発工程やテスト設計に活用する取り組みが行われている．本研究では，経験による属人性を低減するために，経験が浅いメンバーでもバグを推測したテスト設計ができるようにすることに絞った．そして，上記課題の解決策として，経験が浅いメンバーに対して先行テストの中でバグが摘出できたテス

トケースの情報を手早く伝えるために，既存機能と過去の最終確認テストでバグを摘出した時の観点を関連付けることによるバグを推測することを基にしたテスト設計手法を考案した．

本論文の構成は以下のとおりである．2 章ではバグ推測の現状と課題を述べる．3 章ではバグ推測によるテスト設計手法について説明する．4 章では考察を述べる．5 章では今後の課題を述べる．

2　バグ推測の現状と課題

2.1　実施しているテスト内容

研究員の職場では，仕様書にしたがったテストケースの後に，出荷後のユーザーの使用環境や使用方法などの仕様書に記載されていない条件に関するバグの摘出を目的とした最終確認テストを実施している．さらに開発期間が限られた中でより多くのバグを摘出するテストを実施することが求められているため，網羅よりもバグを推測してピンポイントにバグを摘出することに主眼を置いている．

なお本論文では，内部構造を参照せずに行う，機能的または非機能的な製品の振る舞いに対するブラックボックステストのみを研究対象とする．

2.2　最終確認テストを実施するメンバーの現状

研究員のテスト現場では，複数人のテストチームで機能単位にテストを分担してテストプロセス（テスト分析，テスト設計，テスト実装，テスト実施）を一通り実施する．

テストチームは，大きく分けて次の 2 種類のメンバーで構成される．

- 経験が浅いメンバー
  - 新入社員
  - ソフトウェアとは異なる部門から異動した人
- 経験豊富なメンバー
  - 担当するソフトウェアテストにおけるテスト設計を複数年実施した人

上記経験豊富なメンバーは，テストだけでなく出荷後に障害などのトラブルが発生した際に，ログの調査などを実施している．これにより，既存機能におけるバグに関する情報や出荷後の製品の使われ方や製品の内部仕様などの仕様書に記載されていない部分に関する情報を得ることができる．このようなトラブル対応の経験から，最終確認テストでは，同様のバグを推測したテストを実施し，効率的にバグを摘出することができる．

2.3　最終確認テストにおける問題点

参考文献[1]より，研究員 1 名の職場では出荷後に発生しているバグの内訳として仕様書に記載されていない部分に関するバグが 86%を占めている．それに対して，仕様書に記載されていない部分に関するバグの摘出を目的としたテストケースの割合が 10%と少ない．

最終確認テストの内容のうち，仕様書に記載されていない部分に関するバグの摘出を目的としたテストケースは基本的に経験豊富なメンバーによって作成されている．ただし，プロジェクト内でテスト設計およびテスト実施の人数に余裕がなく，すべてのテスト対象に経験豊富なメンバーを割り当てることは難しい場合がある．そのときには，一部のテスト対象に対して経験が浅いメンバーのみで担当しなければならない．よって，仕様書に記載されていない部分については，バグの摘出を目的とする必要がある．経験が浅いメンバーでも適切なバグ推測によるテストを設計できるようになることが望まれている．

2.4　最終確認テストにおける課題

最終確認テストのテスト設計時にインプットとなるデータは，「設計仕様書」と「先行テストの情報」がある．さらにテスト設計の経験豊富なメンバーは図 2.1 に示すような「経験ベース」と「知識ベース」によってバグを推測してテストを実施している．「経験ベース」とは過去のプログラミング経験からプログラム構造を意識した情報や出荷後のトラブル対応からの情報である．「知識ベース」とは製品の位置づけや使われ方の知識や過去に摘出されたバグの知識からの情報である．

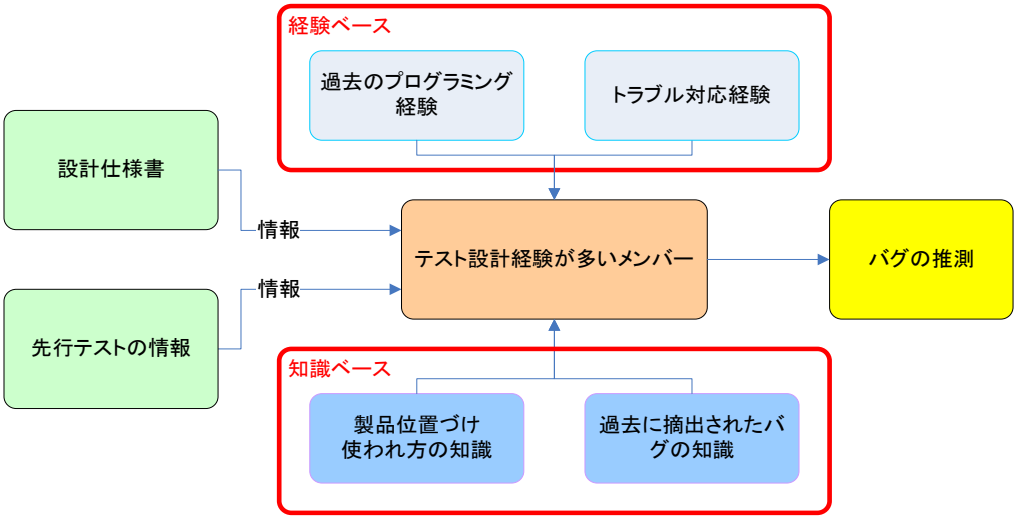

図 2.1　経験豊富なメンバーのテスト設計時のインプット情報

ところが，経験が浅いメンバーがバグを推測したテストを実施するためには，「経験ベース」や「知識ベース」の情報が不足している．

研究員の職場では，過去のテストからの観点リストや過去のバグに関する情報(以下，バグ事例と呼ぶ)を蓄積し利用可能となっているが，観点リストやバグ事例の数が多く，効果的に利用できるように整理されていないことも多い．そのため経験豊富なメンバーにとっては有用な情報であるが，経験の浅いメンバーでは観点リストやバグ事例を参照しても担当する機能に適用可能か判断できなかったり内容を理解できずにテストが不十分になってしまうことがあり課題となっている．

3　バグ推測によるテスト設計手法の提案

3.1　提案手法適用後の目標

2.4 で述べた経験が浅いメンバーの課題に対する解決策として，経験が浅いメンバーが，単独でバグを推測することを基にしたテスト設計手法を考案した．提案手法の適用効果の目標を，経験が浅いメンバーが適用した際の前後の状態を設定し，表 3.1 に示す．

表 3.1　提案手法の適用効果の目標

	説明
適用前	製品の既存機能については理解している
	仕様書にしたがったテストケースは作成できる
適用後	既存機能の変更に対して，過去のバグと同様のバグを推測したテスト設計ができる

3.2　解決策のポイント

2.4 の課題に対して,「経験ベース」の情報を活用したテスト設計時のバグ推測能力の向上は難しいと考える．そこで本研究では「知識ベース」による情報として過去のバグ事例を活用したバグ推測によるテスト設計手法に着目した．

研究員の職場で，経験豊富なメンバーに対してテスト設計時に意識する箇所のアンケートを実施した．ここで対象とした経験豊富なメンバーとは，仕様書に記載されていない部分についてはテストケースが作成できて，そのテストケースでバグを摘出していることを条件とした．その結果を図 3.1 に示す．

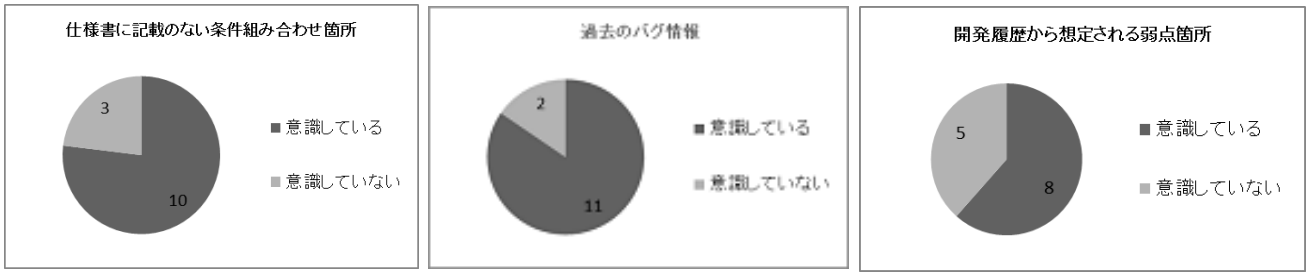

図 3.1　経験豊富なメンバーがテスト作成時に意識すること

経験豊富なメンバーのうち，テスト設計時に過去のバグ情報を意識するメンバーの割合が，他の質問に比べても多かった．

このことから知識ベースの情報として，既存機能の一覧とその機能での過去の最終確認テストでバグを摘出したテスト観点を関連付けた情報をインプットとすることが，経験が浅いメンバーに対してバグを推測する方法として有効という仮説を立てた．

3.3　前提条件

提案手法における，組織やテスト対象の前提条件を表 3.2 に示す．

表 3.2　前提条件

#	前提条件	前提条件とした理由
1	各プロジェクトもしくは各社で過去の最終確認テストで摘出したバグ事例を蓄積し利用できるようになっている	過去のバグ事例を正確に把握するため
2	蓄積したバグ事例に対して内容を理解しているメンバーが存在する	提案手法が有効に機能するようにマトリクスを作成する必要があるため
3	テスト対象が既存機能への変更,もしくは既存機能に類似した機能である	研究員の職場の事情として,経験の浅いメンバーはこのような部分のテストを担当することが多いため

3.4　解決策

既存機能の一覧とその機能での過去の最終確認テストでバグを摘出したテスト観点を関連付けるために，表 3.3 に示すような『既存機能』と『過去のバグ摘出時の観点』からなるマトリクス表を用意する．下記マトリクス表の下線部の数字はバグ表の帳票番号を表す．

表 3.3　既存機能と過去のバグ摘出時の観点からなるマトリクス表

既存機能	過去のバグ摘出時の観点							
	中止後の操作	素早い操作	同時	初期化	冗長化構成	通信負荷	資源競合	・・・
機能 A			11	3, 13, 61	1			
機能 B	15, 28	19	12, 51				30, 45	
機能 C			14			4		
機能 D						22		

ただし，経験が浅いメンバーでは，観点だけでは具体的にテストケースを考え出すのは難しい．そこで，既存機能に対して過去のバグを摘出したテスト観点に該当する箇所に対して，過去の最終確認テストで摘出したバグに関して表 3.4 に示すような現象・発生メカニズム・対策の情報が含まれるバグ事例とのリンク付けをおこなった．

表 3.4　バグ事例に必要な項目

	説明
帳票番号	最終確認テストで摘出したバグに関する情報が一意に特定できる番号
現象	バグによりユーザーから見たテスト対象の振る舞いの情報
発生メカニズム	現象を発生させる原因となるプログラムの動作等の情報
対策	テスト対象が本来あるべき振る舞いをするための修正方法の情報

上記マトリクスとバグ事例を参照することで，経験が浅いメンバーが既存機能の変更に対してどういったバグが摘出できるか推測する．

『過去のバグ摘出時の観点』については，操作や環境条件や発生しうる現象を列挙する。このテスト観点を具体性のある項目に落とし込むと，テスト観点数が膨大になるため作成にかかる工数や適用時の工数が大きい．しかも，経験が浅いメンバーが記載されたテスト観点しか実施しなくなり，バグを推測する効果を低下させてしまう可能性がある．そのため，製品に応じて抽象化を行なったテスト観点にするなどの工夫が必要である．

過去のバグ摘出時のテスト観点を抽象化する例を表 3.5 に示す．まず，バグ事例の現象と発生メカニズムから該当のバグを摘出できるテストケースを考える．次に，そのテストケースから操作内容もしくは環境条件における特徴となるキーワードを導出する．

表 3.5　過去のバグ摘出時のテスト観点を抽象化する例

現象	発生メカニズム	バグを摘出できるテストケース	抽象化したテスト観点
SW A を ON 後に SW B を ON したら，SW B の信号は出力されたが，SW A の信号は出力されなかった	変数への上書き	SW A と SW B を両方 ON し，SW A SW B のどちらも出力されることを確認する	同時
SW の ON/OFF を素早く操作したら，異常を検出した	異常検出条件が過敏な条件であった	SW の ON/OFF を素早く操作し異常が出力されないことを確認する	素早い操作
SW A を ON 後に信号 A の出力を確認し，電源を OFF/ON してから再度 SW A を ON したら信号 A が出力されなかった	初期化処理の不整合	SW A を ON し，電源 ON-OFF 後に SW A が ON されることを確認する	初期化

3.5　提案手法の適用に向けた準備プロセス

経験豊富なメンバーが以下の手順にしたがってマトリクス表を作成する.

① テスト対象の既存機能の一覧を作成する.
② バグ事例から，それらのバグを摘出できるテストケースから抽象化したテスト観点の一覧を作成する.
③ ①と②からマトリクス表を作成する.
④ 既存機能に対してバグを摘出できたテスト観点に該当する箇所にバグ事例の番号などでリンク付けする.

3.6　提案手法を適用したバグ推測のプロセス

経験が浅いメンバーは以下の手順にしたがってバグを推測する.

① 仕様書から機能などの変更(追加・削除を含む)情報を把握する.
② マトリクス表からテスト対象の機能または類似機能を選択する.
③ 選択した機能の行からリンク付けされているテスト観点を確認する.
④ 確認したテスト観点で，どのようなバグが摘出できるか推測する.
⑤ バグが推測できない，もしくは拡充したい場合は，バグ事例を参照する．そして参照したバグ事例と同様のバグを推測する.

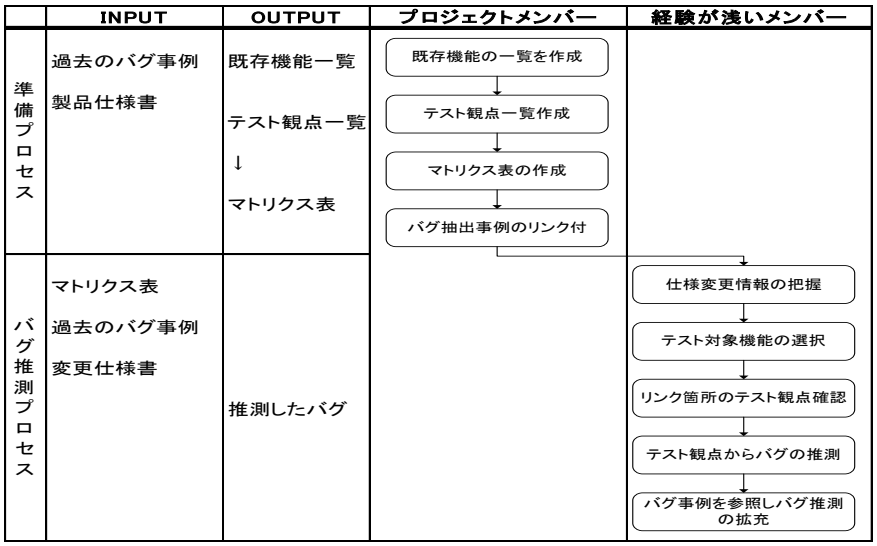

図 3.2　マトリクス表を使用したテスト設計フロー

3.7　提案手法適用の具体例

マトリクス表とそのバグ事例の例を表 3.6 と図 3.3 に示す．想定している製品は，組込み系のスイッチ(以降 SW と示す)の付いた製品とする.縦軸は上記製品の既存機能の一覧，横軸は過去の最終確認テストで摘出した時のテスト観点とし，既存機能とテスト観点の交点はバグ事例の番号を表している.

なお，バグ事例は開発現場で運用方法が異なるため，ここでは簡易的な一例として示す.

表 3.6　想定している製品におけるマトリクス表の例

既存機能	過去のバグ摘出時の観点							
	中止後の操作	素早い操作	同時	初期化	冗長化構成	通信負荷	資源競合	メモリリーク
初期化機能			11	3, 13, 61	1			
SW 入力機能	15, 28	19	12, 51				30, 45	
通信機能			14			4		55
出力機能				72		22, 41		65
検査機能		28			29		85	

バグ表番号	14
発生日時	2015/xx/xx
プロジェクト名	○○SWモジュール
現象	SW AとSW Bを同時にONしたら，SW Bの出力情報しか出力されなかった．
発生メカニズム	SW Bの更新処理がSW BのRAM領域を上書きするプログラムとなっていた．
対策	SW Bの更新処理をSW Aを上書きにしないように，論理和に修正する．

図 3.3　想定している製品におけるバグ事例の例

上記マトリクス表とバグ事例を用いた適用例を説明する．まず「通信機能」に関するテスト設計を行う場合，経験が浅いメンバーはマトリクス表から，テスト対象の機能として「通信機能」を選択する．次に，その行でリンク付けされている「同時」というテスト観点でテストを実施した場合に摘出できるバグを推測する．ここでバグが推測できない，もしくは推測したバグを拡充したい場合は，バグ事例を参照し，「A，B 同時条件成立時に C が動作する」や「複数の入力のタイミングによってデータがクリアされる」といったバグを推測し，これらのバグを摘出する具体的なテストケースを考える．

この提案内容を経験が浅いメンバーが担当する機能のテスト設計時に利用することにより，担当する機能と過去のバグ事例の関連を理解し類似するバグを推測しやすくなることが期待でき，経験豊富なメンバーと同様に効果的にバグを摘出できるようになると考える．

4　考察

最終確認テストで実施している仕様書に記載されていない部分に関するテストを効率的かつ効果的に行なうために，メンバーの経験に依存しないテストの実施が大きな課題であった．今回の提案手法では，既存機能と過去の最終確認テストでのバグ摘出時の観点によるマトリクス表により関連付けることで経験が浅いメンバーが過去にどういった観点でバグが摘出されるのか容易に理解することができる．また，バグ事例の情報を加えることで，具体的に摘出可能なバグも理解できるので，具体的なテストケースを考えることができる．そのため，研究員の職場においては経験が浅いメンバーによる最終確認テストの質

の改善が期待できると考える．

ただ，今回提案した解決策のベースとなるものは過去のバグ事例やそれを摘出したテスト観点であり，それでは新しい機能に対しての効果が期待できないことになるため，本提案手法を新機能にも適用できるような改善が必要である．

5　今後の課題

今回の研究では，研究員の職場における実際のテスト現場で検証ができておらず，提案手法の有効性の評価まで至らなかったため，まず各社での検証が課題となる．

その上で，本提案手法をさらに有効に機能させるための改善を行う．例えば，「経験ベース」の情報を明文化したデータによる拡充，テスト観点の見直し，テストプロセスへの組込みなど．今後はより効率的にバグを摘出するための継続的な改善などに取り組んでいく．

参考文献

[1]．星名 卓郎　他，製品検査における仕様書記載外の問題点摘出強化に向けたテスト観点拡充方法の検討と実践，ソフトウェア品質シンポジウム 2015，2015

[2]．細川 宣啓　他，Project Fabre presents 欠陥マスター情報構築ワークショップ－「悪さの知識」伝承によるバグ予防の為のバグ情報マスター化実践－，JaSST’12 Tokyo，2012

[3]．大谷 和夫　他，上流設計工程における未然防止プロセスの提案　－未然防止リストの活用と欠陥の発想－，JaSST’12 Tokyo，2012

[4]．光永 洋　他，故障事例によるテスト観点知識ベース構築とテスト設計への適用，ソフトウェア品質シンポジウム 2012，2012