ソフトウェアテストシンポジウム 2006
JaSST'06: Japan Symposium on Software Testing 2006

バグのパターンを用いたテストの提案
ーバグのナレッジマネジメントー

河野 哲也† 西 康晴‡

†電気通信大学大学院電気通信学研究科システム工学専攻 〒182-8585 東京都調布市調布ヶ丘 1 - 5 - 1
E-mail: †{kouno, nishi}@se.uec.ac.jp

あらまし 我々の生活に欠かせない携帯電話や家電製品は、内部に組み込まれたソフトウェアによって多機能化され、付加価値を高めている。現在、ソフトウェアのテストには莫大な工数がかかっているが、その多くの原因は増加した機能を組み合わせてテストすることにより、指数関数的にテスト項目数が増加してしまうからである。よって、テスト項目数の急激な増加を抑えつつ、多くのバグを検出することが急務となっている。そこで、本研究ではバグを分析することでテストを改善させる方法を提案する。まず初めに、類似のバグに分類しバグのパターンを抽出する。そして、バグのパターンをテストの観点とすることにより、バグの検出率を向上させる。

キーワード ソフトウェアテスト、フィードバックテスト、パターン、ナレッジマネジメント

The Pattern of Bugs Using for Testing Improvement
ーKnowledge Management of Bugsー

TetsuyaKOUNO† YasuharuNISHI†

†Department of Systems Engineering, The University of Electro-Communications,

1-5-1 Chofugaoka, Chofu, Tokyo 182-8585 Japan
E-mail: †{kouno, nishi}@se.uec.ac.jp

Abstract Our lives depend on software. Most electronic products such as mobile phones and DVD recorders are controlled by embedded software. Nowadays, testing phase takes enormous effort. We should avoid combinations of functions. This paper proposes the method improving testing. First, we extract a pattern of bugs. Second, we detect a lot of bugs for using the pattern.

Keyword Software Testing, Feedback Testing, Pattern of Bugs, Knowledge Management

1. はじめに

我々の生活に欠かせない携帯電話や家電製品は、内部に組み込まれたソフトウェアによって多機能化され、付加価値を高めている。しかし、身の回りにあるソフトウェアには多数のバグが存在し、我々の生活は脅かされている。例えば、自動車は組み込みソフトウェアによって数十の電子制御ユニットが制御されている[1]。もし、エンジンを制御しているソフトウェアにバグがあれば、生活に支障をきたすのは説明するまでもない。

しかし、ソフトウェアの品質が十分に管理できているとは到底言うことはできない。実際に組み込みソフトウェアのバグによるリコール問題が後を絶たない[1]。とはいえ、開発現場ではテストに全体の 3〜5 割の工数をさき、血のにじむような努力をしているのも事実である。

テストに莫大な工数がかかる多くの原因は、増加した機能を組み合わせてテストすることにより、指数関数的にテスト項目数が増加してしまうからである。それが顕著に表れているのが、携帯電話である。テスト項目が 50 万件を超えているという極めて深刻な事例もある[1]。よって、テスト項目数の急激な増加を抑えつつ、多くのバグを検出することが急務となっている。

そこで、すべての組み合わせテストを実施しようとするのではなく違うアプローチが必要である。Davis[2] は「バグは偏在する」という経験則を示し、Kaner[3]は実際的なテスト戦略について「バグの出方を深く理解したうえで、バグが出そうなテストを行うべきである」と述べている。Craig[4]はソフトウェアのリスクに基づくテストを示し、Black[5]は不具合モードを品質リスクとして捉えた FMEA 手法を提案している。西[6]はテストを網羅的に行うのではなく、ピンポイント化することが重要であると主張している。また、河野[7]はバグのパターンに着目したテストの改善を提案しているが、設計したテストケースがどれくらい

のバグを検出できるかは示せていない。

一方で、現場の経験豊富なテスト技術者は、ある程度バグが出ればノウハウ的にバグの傾向をつかみ、明示的ではなく勘と経験で多くのバグを見つけている場合もある。つまり、バグの特徴や性質を把握し、バグの多そうな部分が理解できれば効率的なテストが実施できる。バグの傾向を表出化させ、それをテスト工程へフィードバックさせるのである。つまり、バグのナレッジマネジメントを行うのである。

そこで本研究では、バグの特徴、性質の傾向に着目し、バグのパターンを用いたテストを提案する。そして、実際のソフトウェアのバグに適用する。適用で得られたパターンについてプロダクト・アフォーダンスという概念を用いて考察する。

2. パターン抽出に用いる情報

バグのパターン抽出にはあらゆる情報を用い検討することが望まれるが、テスト工程で扱える情報には限りがある。本研究では次の情報が扱えるものしてパターン抽出を行う。

・バグレポート

・仕様書(要件定義書)

バグレポートとはテスト技術者がバグを検出した時、データベースにバグに関する情報を登録するものである。データベースによりバグレポートの一元管理が行える。通常バグレポートの情報には、バグの概要、再現性、バグ内容及び再現方法、担当者、修正のコメント等[4]が記載されている。バグの概要、バグの内容及び再現性、担当者についはテスト技術者が記入し、修正のコメントは設計の技術者、プログラマが記入する。開発側からの唯一の情報は修正のコメントである。この項目の情報が最も重要と考えるテストチームも多い[8]。

3. バグのパターンを用いたテストの提案

バグのパターンを用いたテストでは、図1のプロセス[7]を提案する。バグの特徴、性質をパターンとして抽出し、得られた複数のパターンをバグのパターンツリーとして構造化を図る。次に、バグのパターンツリーを観点としテスト設計を行う。テスト実施では、テストケースに従い製品をテストする。

3.1. バグのパターン抽出

一言にバグのパターンといっても様々なパターンが考えられるが、本研究ではバグを作り込むメカニズムによるパターンを抽出する。

バグレポートの修正のコメントから、どのようにしてバグが作り込まれたのかのメカニズムを明らかにする。そして、バグを同じようなメカニズムまたは似たようなメカニズムに分類する。本質的に何が同じなのか、何が似ているのか分析し、抽象化の概念を用いパターンを抽出する。ここで、修正のコメントの記述をそろえるとパターン抽出が容易になると言われている[9]。以上の手順より複数のパターン(バグのパターン群)を抽出する。

3.2. バグのパターンの構造化

テストケース設計時に一つ一つのパターンを観点として扱うのではなく、パターン群を構造化しそれを観点として扱う方が適切である。なぜなら、パターン群の中には似ているパターンもあれば、全く異なるパターンも存在する。それらが混在した状態で扱うよりは構造化を図った状態で扱う方が分かり易いからである。本研究では、パターンを構造化したバグのパターンツリーを提案する。

バグのパターン群を似ているパターンに分類し、抽象化の概念を用い新たなパターンを抽出する。つまり、パターンの集まりからパターンを抽出しているのである。そして、新たなパターン群よりパターンを抽出する。このように子(似ているパターン)が親(新たなパターン)を作り、また親がその親を作るといった逆親子関係が成立する。この逆親子関係を明示したものがバグのパターンツリーである。

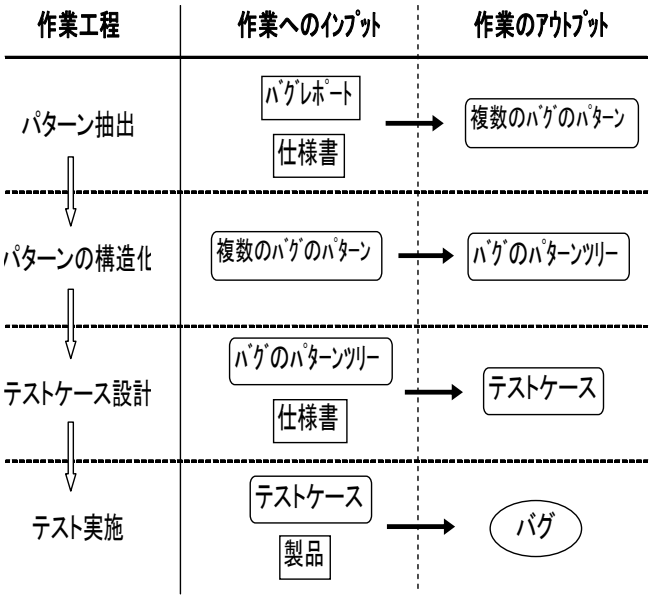

図1　バグのパターンを用いたテストプロセス

3.3. テストケース設計

バグのパターンツリーを観点とし、仕様書の中からバグを作りこんでいるであろう部分を検索し、それをテストするテストケースを設計する。

3.4. テスト実施

テストケースに従い製品をテストし、バグが検出されればバグレポートを登録する。この作業は通常のテスト実施と同様である。

4. 適用

組み込みソフトウェアのGUI部分であるメニュー機能でのバグを対象とし、バグのパターンを用いたテストを適用する。また、テストケースレビューにてテストの精度を検討した。

4.1. バグのパターン抽出

バグが作り込まれたメカニズムを明らかにした結果、数件が同じようなメカニズムとして分類できた。図2のようにパターン抽出を容易にするため、バグレポートの修正のコメントをそろえた。その結果「～時・・・場合があることを考慮していなかった」と書き換えることができた。

ここで抽象化を行う。「～時・・・場合がある」を「・・・・・する場合がある」と置き換える。

「・・・・・する場合がある」のだから通常は「・・・・・する」とは違うことをするのである。つまり、「・・・・・する場合がある」のは「例外」の時なのである。よって図3に示すように、ここで抽出できたパターンは「例外」である。バグを作り込んだメカニズムは、”「例外」を考慮していなかった”といえる。

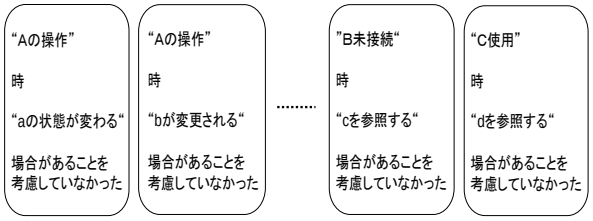

図2　修正のコメントの記述をそろえた例

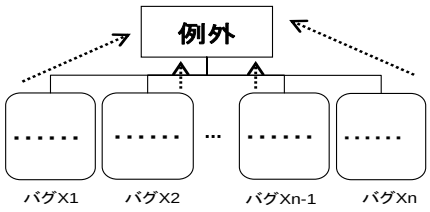

図3　「例外」パターンの抽出

4.2. バグのパターンの構造化

本適用で抽出したパターン「例外」は、親のパターンに該当するものであった。よって、「例外」より子のパターン、孫のパターンと順に導出した。図4に示すようなバグのパターンツリーが構築できた。

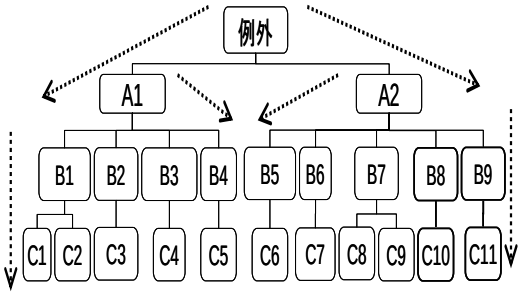

図4　パターンツリー

4.3. テストケース設計

バグのパターンのツリーを観点として、テスト技術者によりテストケース設計を行った。仕様書の中からバグのパターンツリーに該当する部分を検索し、それをテストするテストケースを設計した。

4.4. テストケースレビュー

パターンを用いずに設計したテストケース(テストケース群 A)と本適用で設計したテストケース(テストケース群 B)の精度を検討する。

テストケース群 A は 1834 件、テストケース群 B は 208 件であった。また、テストケース群 A で検出したバグは 39 件、テストケース群 B で検出するであろうバグは 18 件であった。検出率はテストケース群 A で 2.1%、テストケース群 B で 8.7%となった。よって、約 4 倍の精度向上となり、良好な結果が得られた。

5. パターン「例外」の考察

4 章の適用で抽出したパターン「例外」について考察する。バグを作り込むメカニズムを整理すると「パターンはエラーモードを誘発する」といえる。ここで、エラーモードとは人間の作業ミスを一般化したものである。本適用では、”パターン「例外」がエラーモード「考慮漏れ」を誘発している”と説明できる。そこで本章では、「プロダクト・アフォーダンス」[10]という概念を用いた時、「例外」とはどういったものなのか検討し、パターンとエラーモードの関係を明らかにする。

5.1. プロダクト･アフォーダンス

「プロダクト・アフォーダンス」とは、米国の心理学者 Gibson が提案したアフォーダンス[11]を拡張したものである。

アフォーダンスとは、afford(アフォード)の意味する「～ができる、～を与える」からきた造語であり、「ものをどう使うかは、そのものが情報をアフォードしている(与えている)」という概念である。例えば、椅子は“座る”という情報をアフォードし、床は“立つ”、あるいは“歩く”という情報をアフォードしている。

アフォーダンスという概念は、デザインの良し悪しとしてよく議論される[12]。また、安全性においては、製品を扱う場合のヒューマンエラーの起こし易さをアフォーダンスの概念を用いて検討している[13]。

本研究では、ヒューマンエラーを引き起こす原因は、製品がアフォードしている情報と製品の実際の取り扱いとの間に生じるギャップによるものであるとする。例えばドアに取り付けられた水平のバーは引くことをアフォードしている。しかし、実際の取り扱いは押して開ける場合はヒューマンエラーを引き起こしやすいだろう。「引く」と「押す」というギャップが生じ、それが原因でヒューマンエラーを引き起こしたのである。

プロダクト・アフォーダンスとは、「どのように実

装するかは､仕様書自身が情報をアフォードしている」という概念である。つまり、仕様書はどのように実装するのかといった情報をプログラマにアフォードしている。そして、実際に実装すべき情報とアフォードしている情報との間にギャップが生じると、プログラマにはエラーモードが誘発されバグを作り込んでしまうのである。

例えば、一般によく見かけられるバグに、機能が使われなくなったときの処理(非機能要件)の抜けがある。これを、プロダクト・アフォーダンスの概念で説明する｡仕様書というものは､「使う機能の処理を実装すること」をアフォードしている｡つまり､「機能が使われなくなった時の処理を実装すること」はアフォードされず、そこにギャップが生じ、エラーモード「抜け」を誘発させているのである。

5.2. プロダクト･アフォーダンスにおいての「例外」

仕様書にたくさんの同じ処理の項目があり、その中に 1 つだけ例外的に違う処理が含まれていた場合があるとする｡この仕様書は､「たくさんの同じ処理を実装すること」をアフォードし､「1 つだけ例外的に違う処理を実装すること」はアフォードされないのではないだろうか。

例えば､50 項目中 49 項目は同じ処理を割り当て、1 つの項目だけが例外的に違う処理の割り当てが仕様書にあるとする。この仕様書は「50 項目すべてを同じ処理で実装すること」をアフォードしている。しかし、実際に実装すべき情報は｢同じ処理を 49 項目に割り当て､1 項目は違う処理を割り当てる」である。ここで、アフォードしている情報と実際に実装すべき情報との間にギャップが生じ、エラーモード「考慮漏れ」を誘発させているのである。

5.3. パターンとエラーモード

プロダクト・アフォーダンスの概念で、パターンとエラーモードを説明する。仕様書の中でアフォードされない性質がパターンである。そのパターンによりギャップが生じ、それが原因でエラーモードを誘発させるのである。

6. あとがき

本研究では、バグの特徴、性質の傾向に着目し、バグのパターンを用いたテストを提案した。実際のソフトウェアのバグに適用し、パターン抽出、パターンの構造化、テストケース設計を行った。テストケースレビューにてテストの精度を示し良好な結果が得られた。

また、本適用で抽出できたパターンは「例外」であった。この「例外」をプロダクト・アフォーダンスという概念を用い考察した。その結果、アフォードされない性質がパターンであり、そのパターンによりギャップが生じ、エラーモードを誘発させているという結論が得られた。

パターンを明らかにしテスト工程へフィードバックさせることで多くのバグの検出が可能になる。パターンはテストでの大きな武器となり、テスト工程の改善が行える。

今後の課題としては、プロダクト･アフォーダンスに着目したパターン抽出プロセスの検討、そのほかのパターン抽出、新たなテストプロジェクトへの適用などが挙げられる。

謝　辞

本研究を進めるにあたり、(株)ベリサーブの江澤宏和様、浅岡正美様、山崎太郎様には大変お世話になりました。各位に対し心から感謝いたします。また、(株)ベリサーブの皆様にも感謝いたします。

文　献

[1] 日経ビジネス (2005):”ソフトが危ない 品質崩壊 クルマも電機も鉄道も”,日経ＢＰ社, 2005 年 4 月 25 日･5 月 2 日合併号, pp.30-44.

[2] Davis,A.(1995):201Principles of Software Development, McGraw-Hill.(松原友夫訳(1996):”ソフトウェア開発 201 の鉄則”, 日経 BP 社)

[3] Kaner, C. Bach , J . Pettichord, B.(2002): Lessons Learned in Software Testing, John Wiley & Sons. (テスト技術者交流会訳(2003):”ソフトウェアテスト 293 の鉄則”, 日経 BP 社)

[4] Craig, R. Jaskiel, S.(2002):“Systematic Software Testing”,Artech House. (宗雅彦監訳 成田光彰訳 (2004)： “体系的ソフトウェアテスト入門”, 日経 BP 社)

[5] Black, R. (2002):“Managing the Testing Process Second Edition”, John Wiley & Sons. (テスト技術者交流会監訳 トップスタジオ訳 (2004)： “基本から学ぶテストプロセス管理”, 日経 BP 社)

[6] 西康晴, 河野哲也(2005):”障害モードクラスツリーによるテストの改善とプロセスの改善”, 4th Workshop of Critical Software , pp.29-34.

[7] 河野哲也,西康晴(2005)： “バグのパターンに着目したテストの改善”, 日本品質管理学会 第 77 回研究発表会, pp.253-256.

[8] Kaner, C. (1999):“Testing Computer Software Second Edition”, John Wiley & Sons. (テスト技術者交流会訳 (2001)： “基本から学ぶソフトウェアテスト”, 日経 BP 社)

[9] 西康晴(2003)： “不具合の分析とフィードバックによるテストの設計”, Japan Symposium on Software Testing 2003 , pp.49-54.

[10] 西康晴, 河野哲也(2005):”ソフトウェア・ハードウェア複合システムにおける信頼性情報データベースのためのデータ構造の検討”, 電気通信大学大学院情報システム学研究科シンポジウム, pp.14-17.

[11] 佐々木正人(1994):”アフォーダンス-新しい認知の理論”, 岩波書店.

[12] Norman, Donald A(1988):”The Psychology of Everyday Things”,Basic Books.(野島久雄訳(1990):”誰のためのデザイン?認知科学者のデザイン原論”, 新曜社)

[13] 芳賀繁(2000):”失敗のメカニズム”,日本出版サービス