2Z-1 探針型デバッグ手法の実現(1) – 概要と評価

岡本 渉　田村 文隆　植木 克彦　平山 雅之
株式会社東芝 研究開発センター

1 はじめに

ソフトウェアにバグが見つかったとき、通常、プログラマはプログラムを最初から最後まで読み直すことはせず、疑わしい部分から優先的に調査してバグの原因を探る。この疑わしい部分の見当の付け方にはノウハウがあり、それがデバッグの技量に直結している。このような従来方式のデバッグでは、経験の差がデバッグ効率に大きく寄与し、経験の浅いプログラマのデバッグ効率は低くなるという問題点があった。

我々は探針型デバッグ手法という、プログラム中の疑わしい部分、つまり優先的に調査すべき領域を絞り込む手法を研究・開発し、その実装を進めている [2]。この手法の目的は、経験豊かなプログラマが暗黙に行っている作業をツール化することにより、経験の浅いプログラマのデバッグ効率を向上させることにある。

本手法の特徴は、異常動作、正常動作の実行の違いから、バグ原因の領域を絞り込む点にある。これは、異常動作時における特徴的な関数呼び出しや分岐は、それだけバグ原因である可能性が高いことを利用したものである。本論文では、探針型デバッグ手法の概要と、その評価結果を報告する。

2 探針型デバッグの概要

探針型デバッグ手法は、異常動作時と正常動作時のプログラムの振舞いをログとして記録し、それらの比較に基いてバグ原因を探る手法である。本手法の大まかな流れは以下のようなものである。

(1) ログ記録　異常・正常動作のログを収集
(2) ログ選択　比較に適した正常動作ログを選択
(3) 比較　両者の差分を重要度に従って調査

これらの手順の詳細については後述する。

本手法は、正常動作時のログも必要とするため、プログラムが実行可能なレベルまで仕上がっていることが前提条件である。また、概ね正常に動作する中で稀に発生するバグを主なデバッグ対象としている。本手法によるメリットとして、疑わしい領域を絞り込むことでプログラム全体の動作を理解しなくてもデバッグ作業を進められる点が挙げられる。この点は、特に大規模なソフトウェア中のバグ原因の特定に効果を発揮する。

我々は現在、本手法のツール化を進めている。比較の観点としては、(1) プログラムの実行制御の流れに着目 (2) データの違いに着目、という2種類が考えられるが、最初のステップとして、現在のバージョンでは (1) の実行の流れに着目する手法のみ実装している。また、対象は逐次プログラムに限定している。

2.1 探針型デバッグの手順

1. 実行ログの記録

探針型デバッグ手法の最初のステップは、プログラムの動作をログとして記録しながら実行することである。ログは、プログラムの制御が分岐する可能性のある位置(条件文やループなど)にプローブを埋め込み、実行の流れを一意に定められるレベルで記録する。また、ログは異常動作時のものだけでなく、正常動作時のログも併せて記録する。

2. 比較ログ選択

記録されたログが異常動作を示したものか、正常動作を示したものかはユーザが指定する。さらに異常動作ログについては、ログ中のどの時点で異常動作が発現したか、その位置も指定する。それらの情報に基いて、本手法は異常動作ログとの比較に適した正常動作ログを選び出し、ユーザに提示する。選ばれた正常動作ログを「比較対象ログ」と呼ぶ。

望ましい比較対象ログとは、異常動作との差がそのまま異常の原因となっている場合である。したがって、できる限り「似ている」正常動作ログを選択することが本手法において重要なポイントとなる。ログ同士の類似性については第 2.2 節を参照して頂きたい。

またユーザが、比較する領域を指定することで、より効率的な比較を行うことができる。例えばユーザが既に、ある関数内 (そこから呼び出されている関数も含めて) に、バグ原因があるのではないかと見当を付けている場合、その関数をログの「開始点」と指定する。これにより、関係のない箇所の類似性を考慮する必要がなくなる。この事からわかるように、ログは必ずしもプログラム実行の開始から終了までを1つの単位としない。つまり、1本のログを複数の部分ログに切り分けて、別のログとして扱える。

3. 比較

比較対象ログが選択されたら、それと異常動作ログとの差分情報をもとにバグ原因を探る。複数の差分箇所が

"An Implementation of Probe Debugging Method(1) - Outline and Evaluation", Wataru Okamoto, Humitaka Tamura, Katsuhiko Ueki, and Masayuki Hirayama, TOSHIBA corp. R&D Center
e-mail: wa.okamoto@toshiba.co.jp

	バグ内容	種類	ログ数	結果
1	Fix bug with macro name appearing immediately after L'x'	△	10	B+
2	Properly skip // style comments between a function macro name and '(', as well as \ newlines in comments there.	△	9	A
3	In #include directives, \ does not escape "	○	6	A−
4	For '#include "file', don't bother expanding into temp buffer	△	8	A−
5	Handle / \ newline * between # and directive name	○	9	C
6	\" does not end a string	○	9	A+
7	Newline space is not a special marker inside a string	○	9	B−
8	Discard \ newline properly when discarding comments	○	6	B+

「種類」とは、バグが流れによって特定可能かどうかを示す
○： バグ発生時と同じパスの正常動作は存在し得ない
△： バグ発生時と同じパスの正常動作は存在し得る
×： バグ発生時と異なるパスの正常動作が存在し得ない

表 1: cccp.c 内の不具合

存在する場合、「どの差分から調査すべきか」ということがデバッグ効率化のための重要な要素である。本手法では、どの差分が異常動作ログにおいて特徴的 (稀なイベント) であるかを示し、優先的に調査すべき位置をユーザに示す。これにより、複数の差分箇所があったとしても、より異常動作に特徴的な位置からバグ原因の調査を進めることができる。

2.2 ログの類似性

2つのログの何をもって「似た」ログとみなすかは、本手法において最も重要な判断基準である。我々は、単純にログ間の差分の量 (異なるイベントの数) を計るのではなく、イベントの「情報量」を重視して重み付けを行っている。これは、異常動作ログで発生した稀なイベントは、発生自体がバグの原因に直結している可能性が高いことに基いている。そこで、まずログ中の各イベントの出現確率を算出し、各ログにおける各イベントの特徴値を求める。そして、それらの特徴値をもとにログ同士の類似性を計算するという方法を用いた [1]。

3 評価実験

本手法およびツールにおいて用いたアルゴリズム [1] の有効性を評価するため、gcc 2.8.0 の cccp.c (プリプロセッサ部分) をサンプルに実験を行った。不具合情報は gcc 2.8.1 へのバージョンアップ時の差分をもとにしている。ここでは 8 個の不具合をサンプルに用いた (表 1)。また、cccp.c は約 10000 行から成るプログラムである。

この評価実験において重要となるのは、以下の 2 点である。

- 現実のプログラムに含まれるバグの中で、どの程度のバグが実行の流れだけで原因を特定できるか
- 本手法が絞り込んだ異常動作の特徴的位置は、実際のバグ原因と一致しているか

実験の手順として、まず、報告されているバグに対して、バグを再現するログ 1 本と実行条件をわずかに変えて正常に動作するログ数本を準備した。表 1 には、サンプルのバグのリストを挙げた。このサンプルに含まれるバグのうち 63%が、実行制御の流れに正常動作時との違いを発生させる類のものであった (バグ発生時とまったく同じ正常動作ログは存在し得ない)。もし正常動作ログが、異常動作と全く同じ実行パスを辿るならば、そのバグは何らかのデータ依存を持ったバグだということがわかる。

実験の結果、サンプルの 8 個のバグ中、4 個については、異常動作ログにおいて最も特徴的と判断された場所そのものにバグ原因が含まれていた (表 1 の結果 A)。さらに 3 個については、特徴的な位置の周辺を辿って行くことによって、バグ原因を発見できた (結果 B)。残りの 1 個は、異常動作に特徴的な位置からは、かなり離れた位置にバグ原因が存在した (結果 C)。

ただし、本手法は「比較」がポイントとなっている以上、各イベントの特徴を知る母体である正常動作ログ群にどのようなログを準備するかが非常に重要である。これは、あまりにもバイアスのかかったログを集めると、本来の特徴が特徴として読み取れなくなってしまうためである。

4 結論

本論文では、プログラムの実行ログの比較を基本とした探針型デバッグ手法の概要と評価内容について述べた。この手法の特徴は、(1) プログラムの実行ログを記録し (2) 異常動作ログとの比較に適した正常動作ログを選び出し (3) それらの差分のうち優先的にチェックすべき箇所を示す、ことでデバッグ時のバグ原因の調査領域を絞り込むことにある。また、本手法の有効性を評価するために行った実験では、8 個中 7 個のバグ原因の調査において有効という結果が得られた。

さらに本手法を進めて、より効率的なデバッグ支援を行うには、データの違いにも着目した比較をする方法が考えられる。

参考文献

[1] 田村 文隆 他, "探針型デバッグ手法の実現 (2) - アルゴリズム", 62 回情処全国大会 2Z-2 (2001-03)

[2] 植木 克彦 他, "探針型デバッグ手法の提案", 信学技報, Vol.100, No.186, pp.1-8, (2000-7)