ソフトウェアの不具合箇所のパターン群を用いた検出法

多田 啓太† 福原 和哉†† 猪股 俊光† 新井 義和† 曽我 正和‡

† 岩手県立大学ソフトウェア情報学部

†† 岩手県立大学大学院ソフトウェア情報学研究科

‡ 岩手県立大学地域連携センター

1 はじめに

不具合を起こす製品が市場へ出回ることが数多く発生している．その原因の一つとして，テスト等で製品のソースコードに含まれた不具合を検出し切れなかったことが挙げられる．この問題を解決するため，筆者らはソースコード中に含まれた不具合となるコードを「禁止すべきコードパターン（以下，パターン）」として表し，コード中のパターンでパターンと照合する箇所を自動的に検出する手法を提案した [1].

これまでの手法では，パターンの記述が複雑になると誤検出が増えていた．そこで，本研究では，複数個の単純なパターンの組み合わせ (以下，パターン群) を用いて表すことで，誤検出の可能性を減らし，パターン群全てと合致したコードを含むソースファイルの検出 (以下，絞込み) を容易に行えるようにした．また，パターン群の設定にかかる操作を簡略化するために検出システムの GUI の拡張を行った．

2 不具合箇所の検出法

2.1 概 要

過去の事例をもとに，ソフトウェア製品に含まれている不具合の原因となるコードを，パターンとして，パターン言語で記述する．パターン言語は文献 [2] に基づくもので，対象としたプログラミング言語の構文要素を拡張したものである．たとえば，「任意の複数の文」は「@*;」，「式」は「#」と記述される．

検出システムは，ソースコード中にパターンと照合する記述があるかどうかを検出する．照合は文字列照合ではなく，対象プログラミング言語の構文規則に基づいて行われる．

図 1 に示すようにパターンを記述した「パターンファイル」と，ソースコードが記述された「ソースファイル」を検出システムに入力すると，検出システムの内部ではパターンは CPA(Code Pattern Automata)[2] に，ソースコードは AST(Attributed syntax Trees)[2] にそれぞれ変換され，CPA の状態遷移にもとづきながら照合が行われる．

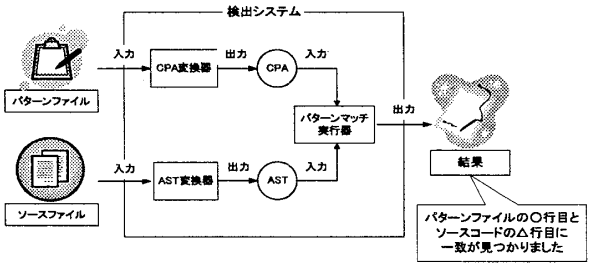

図 1: 不具合箇所検出法

2.2 問題点

この検出システムはパターンを CPA へ変換し，初期状態から AST の属性を状態遷移の条件として，受理状態への到達を目指して遷移する．このため，検出したいパターンがソースコード中に出現する順番も考慮しながら記述しなければならず，複数個のパターンを含んだ不具合の記述は難しい．

たとえば「パターン A との一致があり，その後にパターン B との一致がある」という記述は容易だが，「パターン A との一致と，パターン B との一致が，順不同で出現する」という不具合を表すときは，正規表現なども併用してパターンを記述しなければならなかった．

3 パターン群を用いた検出システム

3.1 GUI の拡張

既存の検出システムの GUI では，パターンファイルは 1 つしか設定できなかった．そのため複数個のパターンファイルを照合する場合，一つのパターンファイルの照合が終るたびに，次に調べるべきパターンファイルを設定し直さなければならない．

そこで，複数個のパターンファイル群を設定できるように，図 2 のような GUI を実装した．ユーザーは検出に用いるパターンファイル群，照合したいソースファイル群をそれぞれ登録し，実行ボタンを押すことで検出システムを使用する．

また，設定したパターンファイル群を用いて絞込み照合を行うのか，絞込みを行わない通常の照合を行うのかを区別するためのチェックボックスを設けた．

3.2 処理系の動作

本検出法では，パターンファイル群とソースファイル群を用いて，図 3 のような流れで照合を行っていく．

処理が終ったとき，結果として出力されたソースファイルには，パターン群で表される複雑な不具合が全て含まれているものと判断される．

A Detection Method For Fault Program Patterns Detection

† Keita TADA,Toshimitsu INOMATA,Yoshikazu ARAI

†† Kazuya FUKUHARA

‡ Masakazu SOGA

Faculty of Software and Information Science, Iwate Prefectural University (†)

Graduated School of Software and Information Science, Iwate Prefectural University (††)

Iwate Prefectural University Regional Cooperative Reserch Center(‡)

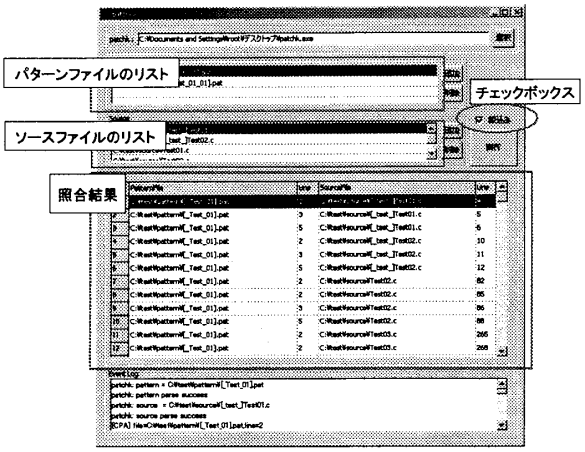

図 2: 実装した GUI

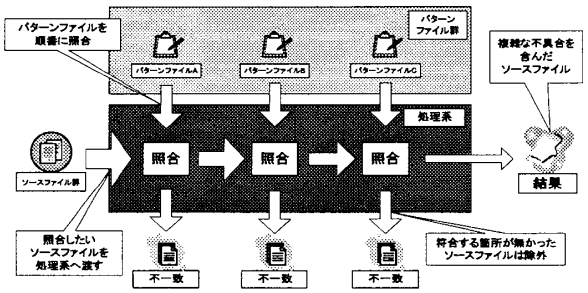

図 3: パターンファイル群を用いた絞込み検出の流れ

パターン P1

```
fp1 = fopen("abc.txt", "r");
@*;
for ( #; #; #) {
    @*;
    strncpy(List[i], buf, MAXLEN);
    @*;
}
@*;
fclose(fp1);
```

パターン P2

```
fp2 = fopen("xyz.txt", "w");
@*;
for (; #; #){
    fputs(List[i], fp2);
    @*;
}
@*;
fclose(fp2);
```

図 4: パターン P1 と P2

表 1: 各ソースファイルに含まれる不具合コード数

		パターンファイル群			
		D	E	F	G
ソースファイル群	F1	1	7	1	6
	F2	1	0	1	6
	F3	1	7	1	0
	F4	1	7	0	6
	F5	0	7	1	6

4 評価

4.1 複数個の単純なパターンを用いた検出

図 4 のパターン P1 と P2 を記述したパターンファイル A，パターン P1 のみを記述したパターンファイル B，パターン P2 のみを記述したパターンファイル C を用いて，パターン P1 と P2 を含んだ複数個のソースコードに対して照合を行った．

パターンファイル C を用いた場合は，パターンが A，B の順にソースコード中に含まれていた場合にのみ検出できた．パターンファイル A と B の両方を用いた場合は，それぞれのパターンが記述されていれば，出現順にかかわらず正しく検出されることが確認できた．

4.2 絞込み

4 つのパターンファイル D，E，F，G を用意し，複数個の不具合コードを含んだソースファイルに対し照合を行った．ソースファイルは 5 個用意し，それぞれ F1〜F5 とした．表 1 に，各ソースファイルに含まれている不具合コードのパターンファイルごとの数を示す．全パターンファイルにあてはまる不具合コードを含むのは F1 だけであり，他は一部を含む．

そして，これらのファイル群に通常の照合を行ったときと，絞込みを行ったときとの結果を表 2 に示す．

通常照合の結果は，一部のパターンにあてはまるソースファイルを全て出力していた．これに対し，絞り込み照合の結果では，不具合をすべて含んだソースファイルだけが正しく検出できているのが確認できた．

5 おわりに

本研究で考案した検出法を用いれば，単純なパターン群を用いた照合を行うことで，より正確に不具合の原因となるコードを検出できることが確かめられた．

今後の課題として，パターン群を用いた照合をより柔軟に行うことが挙げられる．たとえば「パターン A，またはパターン B に一致するとき，パターン C と照合」といったたとえば「パターン A，またはパターン B に一致するとき，パターン C と照合」といった不具合の実装が課題となる．

参考文献

[1] 福原和哉，猪股俊光，新井義和，曽我正和：ソフトウェア製品の不具合原因となるコードパターンを用いた検証手法，第 4 回システム検証の科学技術シンポジウム，2007.

[2] Santanu Paul, Atul Prakash : A Framework for Source Code Search Using Program Patterns., IEEE Transctions on software engineering, Vol.20.pp.463-475(1994).

表 2: 通常照合と絞込みの比較

	通常照合	絞込み
検出した不具合箇所数	60	15
検出したソースファイル	F1〜F5	F1