(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7384054号
(P7384054)

(45)発行日　令和5年11月21日(2023.11.21)

(24)登録日　令和5年11月13日(2023.11.13)

(51)国際特許分類
G06F 11/36 (2006.01)
G06F 8/70 (2018.01)

FI
G06F 11/36 164
G06F 11/36 124
G06F 11/36 108
G06F 11/36 112
G06F 8/70

請求項の数　10　（全42頁）

(21)出願番号　特願2020-12638(P2020-12638)
(22)出願日　令和2年1月29日(2020.1.29)
(65)公開番号　特開2020-129371(P2020-129371A)
(43)公開日　令和2年8月27日(2020.8.27)
審査請求日　令和4年10月6日(2022.10.6)
(31)優先権主張番号　16/270509
(32)優先日　平成31年2月7日(2019.2.7)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100107766
弁理士　伊東 忠重
(74)代理人　100070150
弁理士　伊東 忠彦
(72)発明者　吉田 浩章
アメリカ合衆国，カリフォルニア州 ９４０８５，サニーヴェイル，イースト アークス アヴェニュ １２４０番 フジツウ ラボラトリーズ アメリカ内
(72)発明者　プラサド・ムクル アール
アメリカ合衆国，カリフォルニア州 ９

最終頁に続く

(54)【発明の名称】　自動化されたソフトウェアプログラム修復

(57)【特許請求の範囲】

【請求項１】

複数のイベント対応を決定し、各イベント対応は、第１のソフトウェアプログラムの第１のソースコードに対してなされた複数の変更のうちそれぞれの変更と前記第１のソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示すステップであり、前記複数のイベント対応は、複数のイベントタイプ推論動作のうち２つ以上のイベントタイプ推論動作を実行することにより決定され、前記複数のイベントタイプ推論動作は、

不良導入イベントが前記複数の変更のうち第１の変更に対応することを、前記第１の変更が前記第１のソースコードに含まれないで合格し、かつ前記第１の変更が前記第１のソースコードに含まれて不合格になった前記第１のソースコードの第１のソフトウェアテストを識別することに基づいて、決定することと、

不良訂正イベントが前記複数の変更のうち第２の変更に対応することを、前記第２の変更が前記第１のソースコードに含まれないで不合格になり、かつ前記第２の変更が前記第１のソースコードに含まれて合格した前記第１のソースコードの第２のソフトウェアテストを識別することに基づいて、決定することと、

欠陥導入イベントが前記複数の変更のうち第３の変更に対応することを、前記第３の変更が前記第１のソースコードに含まれないで前記第１のソースコードに対して実行された第１の静的解析から第１の欠陥が識別されないことに基づいて、及び前記第３の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第２の静

的解析から前記第１の欠陥が識別されることに基づいて、決定することと、
　欠陥訂正イベントが前記複数の変更のうち第４の変更に対応することを、前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第３の静的解析から第２の欠陥が識別されることに基づいて、及び前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第４の静的解析から前記第２の欠陥が識別されないことに基づいて、決定することと、
　特定のプラットフォームの第１のバージョンから前記特定のプラットフォームの第２のバージョンへのプラットフォーム移行イベントが前記複数の変更のうち第５の変更に対応することを、前記第５の変更が中に含まれた前記第１のソースコードの第１のビルドが、前記第５の変更が中に含まれた前記第１のソースコードの第２のビルドに関して省略されるエラーを有することに基づいて、決定することであり、前記第１のビルドは前記特定のプラットフォームの前記第１のバージョンを使用して実行され、前記第２のビルドは前記特定のプラットフォームの前記第２のバージョンを使用して実行される、ことと、
　を含む、ステップと、
　第２のソフトウェアプログラムの第２のソースコードにおける１つ以上のエラーを、前記第２のソースコードに関してテストスイートを実行することに基づいて識別するステップと、
　前記第１のソースコードから決定された前記複数のイベント対応を使用して前記１つ以上のエラーに関して前記第２のソースコードに対して修復動作を実行するステップと、
　を含む方法。
【請求項２】
　前記複数の変更のうち特定の変更についてのコミットメッセージを生成し、前記コミットメッセージは前記特定の変更に関して決定されたそれぞれのイベント対応を示すステップ、をさらに含む請求項１に記載の方法。
【請求項３】
　前記不良導入イベントが前記第１の変更に対応すると決定することは、
　前記第１の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第１のソフトウェアテストを含む、ことと、
　前記第１の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、
　前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別することと、
　前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することと、
　前記不良導入イベントが前記第１の変更に対応することを、前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別すること及び前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することに応じて決定することと、
　を含む、請求項１に記載の方法。
【請求項４】
　前記不良訂正イベントが前記第２の変更に対応すると決定することは、
　前記第２の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第２のソフトウェアテストを含む、ことと、
　前記第２の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、
　前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別することと、
　前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別する

ことと、
前記不良訂正イベントが前記第２の変更に対応することを、前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別すること及び前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別することに応じて決定することと、
を含む、請求項１に記載の方法。
【請求項５】
前記欠陥導入イベントが前記第３の変更に対応すると決定することは、
前記第３の変更を含む前記第１のソースコードの部分に関して前記第２の静的解析を実行することであり、前記第２の静的解析は前記第１の欠陥を識別する、ことと、
前記第３の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第１の静的解析を実行することであり、前記第１の静的解析は前記第１の欠陥を識別しない、ことと、
前記欠陥導入イベントが前記第３の変更に対応することを、前記第２の静的解析が前記第１の欠陥を識別すること及び前記第１の静的解析が前記第１の欠陥を識別しないことに応じて決定することと、
を含む、請求項１に記載の方法。
【請求項６】
前記欠陥訂正イベントが前記第４の変更に対応すると決定することは、
前記第４の変更を含む前記第１のソースコードの部分に関して前記第３の静的解析を実行することであり、前記第３の静的解析は前記第２の欠陥を識別しない、ことと、
前記第４の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第４の静的解析を実行することであり、前記第４の静的解析は前記第２の欠陥を識別する、ことと、
前記欠陥訂正イベントが前記第４の変更に対応することを、前記第４の静的解析が前記第２の欠陥を識別すること及び前記第３の静的解析が前記第２の欠陥を識別しないことに応じて決定することと、
を含む、請求項１に記載の方法。
【請求項７】
前記プラットフォーム移行イベントが前記第５の変更に対応すると決定することは、
前記第５の変更を含む前記第１のソースコードの部分に関して前記第１のビルドを実行することであり、前記第１のビルドは前記エラーを識別する、ことと、
前記第５の変更を中に含ませた前記第１のソースコードの前記部分に関して前記第２のビルドを実行することであり、前記第２のビルドは前記エラーを識別しない、ことと、
前記プラットフォーム移行イベントが前記第５の変更に対応することを、前記第１のビルドが前記エラーを識別すること及び前記第２のビルドが前記エラーを識別しないことに応じて決定することと、
を含む、請求項１に記載の方法。
【請求項８】
命令を記憶するように構成された１つ以上の非一時的コンピュータ読取可能記憶媒体であって、前記命令は、実行されたことに応じてシステムに動作を実行させ、前記動作は、
複数のイベント対応を決定し、各イベント対応は、第１のソフトウェアプログラムの第１のソースコードに対してなされた複数の変更のうちそれぞれの変更と前記第１のソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示すことであり、前記複数のイベント対応は、複数のイベントタイプ推論動作のうち２つ以上のイベントタイプ推論動作を実行することにより決定され、前記複数のイベントタイプ推論動作は、
不良導入イベントが前記複数の変更のうち第１の変更に対応することを、前記第１の変更が前記第１のソースコードに含まれないで合格し、かつ前記第１の変更が前記第１のソースコードに含まれて不合格になった前記第１のソースコードの第１のソフトウェアテストを識別することに基づいて、決定することと、

不良訂正イベントが前記複数の変更のうち第２の変更に対応することを、前記第２の変更が前記第１のソースコードに含まれないで不合格になり、かつ前記第２の変更が前記第１のソースコードに含まれて合格した前記第１のソースコードの第２のソフトウェアテストを識別することに基づいて、決定することと、

欠陥導入イベントが前記複数の変更のうち第３の変更に対応することを、前記第３の変更が前記第１のソースコードに含まれないで前記第１のソースコードに対して実行された第１の静的解析から第１の欠陥が識別されないことに基づいて、及び前記第３の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第２の静的解析から前記第１の欠陥が識別されることに基づいて、決定することと、

欠陥訂正イベントが前記複数の変更のうち第４の変更に対応することを、前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第３の静的解析から第２の欠陥が識別されることに基づいて、及び前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第４の静的解析から前記第２の欠陥が識別されないことに基づいて、決定することと、

特定のプラットフォームの第１のバージョンから前記特定のプラットフォームの第２のバージョンへのプラットフォーム移行イベントが前記複数の変更のうち第５の変更に対応することを、前記第５の変更が中に含まれた前記第１のソースコードの第１のビルドが、前記第５の変更が中に含まれた前記第１のソースコードの第２のビルドに関して省略されるエラーを有することに基づいて、決定することであり、前記第１のビルドは前記特定のプラットフォームの前記第１のバージョンを使用して実行され、前記第２のビルドは前記特定のプラットフォームの前記第２のバージョンを使用して実行される、ことと、

を含む、ことと、

第２のソフトウェアプログラムの第２のソースコードにおける１つ以上のエラーを、前記第２のソースコードに関してテストスイートを実行することに基づいて識別することと、

前記第１のソースコードから決定された前記複数のイベント対応を使用して前記１つ以上のエラーに関して前記第２のソースコードに対して修復動作を実行することと、

を含む、１つ以上の非一時的コンピュータ読取可能記憶媒体。

【請求項９】

前記動作は、前記複数の変更のうち特定の変更についてのコミットメッセージを生成し、前記コミットメッセージは前記特定の変更に関して決定されたそれぞれのイベント対応を示すことをさらに含む、請求項８に記載の１つ以上のコンピュータ読取可能記憶媒体。

【請求項１０】

システムであって、

命令を記憶するように構成された１つ以上のコンピュータ読取可能記憶媒体と、

前記１つ以上のコンピュータ読取可能記憶媒体に通信上結合され、前記命令の実行に応じて当該システムに動作を実行させるように構成された１つ以上のプロセッサと、

を含み、前記動作は、

複数のイベント対応を決定し、各イベント対応は、第１のソフトウェアプログラムの第１のソースコードに対してなされた複数の変更のうちそれぞれの変更と前記第１のソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示すことであり、前記複数のイベント対応は、複数のイベントタイプ推論動作のうち２つ以上のイベントタイプ推論動作を実行することにより決定され、前記複数のイベントタイプ推論動作は、

不良導入イベントが前記複数の変更のうち第１の変更に対応することを、前記第１の変更が前記第１のソースコードに含まれないで合格し、かつ前記第１の変更が前記第１のソースコードに含まれて不合格になった前記第１のソースコードの第１のソフトウェアテストを識別することに基づいて、決定することと、

不良訂正イベントが前記複数の変更のうち第２の変更に対応することを、前記第２の変更が前記第１のソースコードに含まれないで不合格になり、かつ前記第２の変更が前記第１のソースコードに含まれて合格した前記第１のソースコードの第２のソフトウェア

テストを識別することに基づいて、決定することと、
　　欠陥導入イベントが前記複数の変更のうち第３の変更に対応することを、前記第３の変更が前記第１のソースコードに含まれないで前記第１のソースコードに対して実行された第１の静的解析から第１の欠陥が識別されないことに基づいて、及び前記第３の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第２の静的解析から前記第１の欠陥が識別されることに基づいて、決定することと、
　　欠陥訂正イベントが前記複数の変更のうち第４の変更に対応することを、前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第３の静的解析から第２の欠陥が識別されることに基づいて、及び前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第４の静的解析から前記第２の欠陥が識別されないことに基づいて、決定することと、
　　特定のプラットフォームの第１のバージョンから前記特定のプラットフォームの第２のバージョンへのプラットフォーム移行イベントが前記複数の変更のうち第５の変更に対応することを、前記第５の変更が中に含まれた前記第１のソースコードの第１のビルドが、前記第５の変更が中に含まれた前記第１のソースコードの第２のビルドに関して省略されるエラーを有することに基づいて、決定することであり、前記第１のビルドは前記特定のプラットフォームの前記第１のバージョンを使用して実行され、前記第２のビルドは前記特定のプラットフォームの前記第２のバージョンを使用して実行される、ことと、
　を含む、ことと、
　前記複数の変更のうち特定の変更についてのコミットメッセージを生成し、前記コミットメッセージは前記特定の変更に関して決定されたそれぞれのイベント対応を示すことと、
　を含む、システム。
【発明の詳細な説明】
【技術分野】
【０００１】
　本開示で論じられる実施形態は、自動化されたソフトウェアプログラム修復に関する。
【背景技術】
【０００２】
　ソフトウェアプログラムは、その中にエラー（一般に「バグ」と呼ばれる）をしばしば有し、これらは、意図したとおりに動作しない可能性がある。ソフトウェアプログラム内のエラーを識別し、訂正しようとして、しばしば、自動化された修復システムが使用される。
【０００３】
　本開示において請求される対象事項は、何らかの欠点を解決し又は上述されたような環境においてのみ動作する実施形態に限定されない。むしろ、この背景技術は、本開示に記載されるいくつかの実施形態が実施され得る１つの例示的な技術分野を示すためにのみ提供される。
【発明の概要】
【０００４】
　一実施形態に態様によれば、動作が複数のイベント対応を決定することを含み得る。各イベント対応は、第１のソフトウェアプログラムの第１のソースコードに対してなされた複数の変更のうちのそれぞれの変更と、第１のソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示し得る。複数のイベント対応は、複数のイベントタイプ推論動作のうち２つ以上のイベントタイプ推論動作を実行することにより決定されてもよい。複数のイベントタイプ推論動作は、不良導入イベントが複数の変更のうち第１の変更に対応することを、第１の変更が第１のソースコードに含まれないで合格し、かつ第１の変更が第１のソースコードに含まれて不合格になった第１のソースコードの第１のソフトウェアテストを識別することに基づいて、決定することを含み得る。複数のイベントタイプ推論動作は、不良訂正イベントが複数の変更のうち第２の変更に対応すること

を、第２の変更が第１のソースコードに含まれないで不合格になり、かつ第２の変更が第１のソースコードに含まれて合格した第１のソースコードの第２のソフトウェアテストを識別することに基づいて、決定することをさらに含んでもよい。さらに、複数のイベントタイプ推論動作は、欠陥導入イベントが複数の変更のうち第３の変更に対応することを、第３の変更が第１のソースコードに含まれないで第１のソースコードに対して実行された第１の静的解析から第１の欠陥が識別されないことに基づいて、及び第３の変更が第１のソースコードに含まれて第１のソースコードに対して実行された第２の静的解析から第１の欠陥が識別されることに基づいて、決定することを含んでもよい。さらに、複数のイベントタイプ推論動作は、欠陥訂正イベントが複数の変更のうち第４の変更に対応することを、第４の変更が第１のソースコードに含まれて第１のソースコードに対して実行された第３の静的解析から第２の欠陥が識別されることに基づいて、及び第４の変更が第１のソースコードに含まれて第１のソースコードに対して実行された第４の静的解析から第２の欠陥が識別されないことに基づいて、決定することを含んでもよい。複数のイベントタイプ推論動作は、特定のプラットフォームの第１のバージョンから特定のプラットフォームの第２のバージョンへのプラットフォーム移行イベントが複数の変更のうち第５の変更に対応することを、第５の変更が中に含まれた第１のソースコードの第１のビルドが、第５の変更が中に含まれた第１のソースコードの第２のビルドに関して省略されるエラーを有することに基づいて、決定することであり、第１のビルドは特定のプラットフォームの第１のバージョンを使用して実行され、第２のビルドは特定のプラットフォームの第２のバージョンを使用して実行される、ことをさらに含んでもよい。

【０００５】

実施形態の目的及び利点は、少なくとも特許請求の範囲において特に指し示された要素、特徴、及び組み合わせにより実現され、達成される。

【０００６】

前述の一般的な説明及び以下の詳細な説明の双方が例として与えられ、説明的であり、請求される発明の限定ではない。

【図面の簡単な説明】

【０００７】

例示的な実施形態が、添付図面の使用を通してさらなる特定性及び詳細と共に記載され、説明される。

【図１】ソフトウェアプログラムに関して生じるイベントの推論に関連した一例示的な環境を表す図である。

【図２】ソフトウェアプログラムからの二次的な修正の除去に関連した一例示的な環境を表す図である。

【図３】ソフトウェアプログラムの修復に関連した一例示的な環境を表す図である。

【図４】一例示的なコンピューティングシステムのブロック図を示す。

【図５】ソフトウェアプログラムに関して生じるイベントを推論する一例示的な方法のフローチャートである。

【図６】不良導入イベント及び不良訂正イベント推論動作を実行する一例示的な方法のフローチャートである。

【図７】欠陥導入イベント及び欠陥訂正イベント推論動作を実行する一例示的な方法のフローチャートである。

【図８】ソフトウェアプログラムのソースコードから二次的な修正を除去する一例示的な方法のフローチャートである。

【図９Ａ】ソースコードから二次的な修正を除去する際に使用され得る例示的な抽象構文木（ＡＳＴ）を示す。

【図９Ｂ】図９ＡのＡＳＴの例示的な部分木を示す。

【図９Ｃ】図９Ｂの部分木の例示的なテキスト表現を示す。

【図９Ｄ】図９Ｃのテキスト表現の、例示的な異なるテキスト表現を示す。

【図９Ｅ】修正されたＡＳＴを含むＡＳＴを示す。

【図９Ｆ】図９ＥのＡＳＴの例示的な部分木を示す。
【発明を実施するための形態】
【０００８】
本開示に記載されるいくつかの実施形態は、ソフトウェアプログラムを修復する方法及びシステムに関する。ソフトウェアプログラムは、ソフトウェアプログラムを意図されない方法で挙動させる可能性があるエラー（一般に「バグ」と呼ばれる）をしばしば含む。さらに、エラーを検出及び訂正してソフトウェアプログラムを修復するために、しばしば、自動化された修復システム及び手法が使用される。
【０００９】
さらに、既存のソフトウェアプログラムのリポジトリが使用されて、開発又はテストされているコード内のエラーを識別及び／又は訂正するために使用され得る対応するコード内のパターンを識別することがある。いくつかの例において、既存のソフトウェアプログラムのリポジトリは、多数のソフトウェアプログラム（例えば、数千、数万、数十万、数百万等のソフトウェアプログラム）のソースコードを含み得る。さらに、リポジトリは、ソースコードに対してなされた１つ以上の変更を通してなされるそれぞれのソフトウェアプログラムのソースコードの異なるイテレーション（iterations）を含むことがある。本開示において、このようなリポジトリに記憶され、他のソフトウェアプログラムの開発に役立つよう使用され得る既存のソフトウェアプログラム及び対応するソースコードは、「ビッグコード」と呼ばれることがある。いくつかの例において、特定のソフトウェアプログラムのソースコードの異なるイテレーション間でなされた変更は、エラーを訂正するものである可能性があり、かつ／あるいはエラーを導入する可能性がある。さらに又は代わりに、変更は、プラットフォーム移行、追加された挙動、除去された挙動、ソースコードのリファクタリング等に起因する可能性がある。ソースコードの変更を引き起こし、あるいはソースコードの変更により引き起こされるイベントを理解することは、変更がいつ及びどこで必要とされ、かつ／あるいは問題となる可能性があるかを識別するのに役立つことにより、他のソフトウェアの開発に役立つ可能性がある。
【００１０】
いくつかの例において、ソースコードの１つ以上の変更は、対応する変更がなぜなされたのかのいくらかの洞察を提供するために（例えば、対応する変更を引き起こし又は対応する変更により引き起こされたイベントに関する洞察を提供するために）生成された対応するメッセージ（「コミットメッセージ」と呼ばれる）を有することがある。しかしながら、既存のソースコードを用いて含まれるコミットメッセージは、しばしば、意味のある洞察を提供するには過度にあいまいであり、かつ／あるいは幅広い。さらに、それぞれの変更に対応するイベントもまた、一般に、容易に明らかではない。そのようなものとして、既存のソフトウェアプログラムのリポジトリにおける変更に対応するイベントは、開発中であるか又はテストされているが容易に利用できないソフトウェアプログラムを向上させる際にかなり役立つ可能性がある、活用されていないリソースである。
【００１１】
本開示の１つ以上の実施形態によれば、ソフトウェア開発の技術分野は、コンピューティングシステムが既存のソフトウェアプログラムの異なるバージョン間でなされた変更に対応するイベントを推論することができるようにコンピューティングシステムを構成することにより、向上し得る。さらに、いくつかの実施形態において、コンピューティングシステムは、推論イベントに基づいて、ソフトウェアプログラムに関連づけられたコミットライブラリに典型的に含まれるものより詳細で説明的なコミットメッセージを生成するように構成されてもよい。
【００１２】
これら又は他の実施形態において、コンピューティングシステムは、他のソフトウェアプログラムをテストすること及び／又は開発することに関して、推論イベント及び／又は向上したコミットメッセージを使用するように構成されてもよい。例えば、コンピューティングシステムは、ソフトウェアプログラムのテスト中コードに関して発生したイベント

をビッグコードから推論されたイベントと比較して、テスト中コードにおいて導入され又は直された可能性のある潜在的なエラーを識別するように構成されてもよい。さらに又は代わりに、コンピューティングシステムは、ビッグコードにおいて、推論イベントに関連づけられたコードパターンを識別するように構成されてもよい。これら又は他の実施形態において、識別されたコードパターンに関連づけられた推論イベントが、テスト中コードに関連づけられたイベントと同じであるか又は類似することに応じて、識別されたコードパターンが、テスト中コードに関する修復を選択又は実現するために（例えば、類似のコードパターンを有する修復を選択又は実現するために）使用されてもよい。例えば、２０１７年１１月２４日に出願された米国特許出願第１５／８２２，１０６号及び２０１８年３月８日に出願された米国特許出願第１５／９１５，８９４号は、双方がその全体を参照により本明細書に組み込まれ、他のソースコードにおける修復を実現するための、既存のソースコードにおけるコードパターンの使用を論じている。

【００１３】

さらに又は代わりに、ソースコードの変更は、対応するソフトウェアプログラムの根本的な機能性に影響しない１つ以上の二次的な修正（secondary modifications）を時に含むことがある。既存のソースコード（例えば、ビッグコード）に、他のソースコードの開発に役立つよう使用され得る二次的な修正を含めることは、既存のソースコードのうちどの部分が他のソースコードの開発又はテストに役立ち、あるいは役立たない可能性があるかを識別することを、困難にする可能性がある。

【００１４】

本開示の１つ以上の実施形態によれば、動作が、ソースコード内の二次的な修正を識別し、識別された二次的な修正を除去するために実行されてもよい。二次的な修正の除去は、他のソースコードの開発及びテストに有用であり得る既存のソースコード内のコードパターンの識別を容易にするのに役立つ可能性がある。これら又は他の実施形態において、イベントの推論は、以下に詳述されるように、二次的な修正を識別するのに役立つよう使用されてもよい。本開示において、「二次的な」修正は、対応するソースコードの根本的な機能性を変更し得ない修正を参照し得る。例えば、「二次的な」修正は、（例えば、可読性を向上させるための）表面的な修正、又はデバッグ目的で使用され得る修正を含む。

【００１５】

さらに又は代わりに、二次的な修正の除去は、どの修正がエラーを含み得るかを識別するのに役立つ可能性がある。例えば、特定のソースコードは、特定のソースコードに１つ以上のエラーを導入し得る特定の変更を、特定のソースコードに対してなされる可能性がある。変更は、特定のソースコードに対してなされた複数の修正を含むことがあり、複数の修正のうち１つ以上が、それを含むことがエラーに寄与していない可能性がある二次的な修正であり得る。ゆえに、二次的な修正の除去は、変更のうちどの修正がエラーに寄与したかを識別するのに役立つ可能性がある。ゆえに、向上したエラーの識別は、エラーの訂正を容易にする可能性があり、結果的に、特定のソースコードを向上させるのに役立つ可能性がある。

【００１６】

本開示の実施形態は、添付の図面を参照して説明される。

【００１７】

図１は、本開示に記載される少なくとも１つの実施形態に従って配置された、ソフトウェアプログラムに関して生じるイベントの推論に関連した一例示的な環境１００を表す図である。環境１００は、解析モジュール１０６を含んでもよく、解析モジュール１０６は、ソフトウェアプログラムのソースコード１０４を解析して、ソフトウェアプログラムに関して生じ、かつソースコード１０４に対してなされた１つ以上の変更に関連づけられた１つ以上の推論イベント（inferred events）１０８を決定するように構成される。これら又は他の実施形態において、環境１００は、コミットメッセージモジュール１１０を含んでもよく、コミットメッセージモジュール１１０は、推論イベント１０８を取得し、対応するコミットメッセージ１１２を生成するように構成される。各コミットメッセージ１

１２は、それぞれの変更の理由又は原因である可能性があり、あるいはそれぞれの変更の結果であった可能性がある、それぞれの推論イベント１０８の指標を含んでもよい。

【００１８】

ソースコード１０４は、例えば、ソフトウェアプログラム、ソフトウェアプログラムのコード、ライブラリ、アプリケーション、スクリプト、又は処理デバイスによる実行のための他の論理若しくは命令などの電子データを含んでもよい。いくつかの実施形態において、ソースコード１０４は、ソフトウェアプログラムの完全なインスタンスを含んでもよい。さらに又は代わりに、ソースコード１０４は、ソフトウェアプログラムの一部分を含んでもよい。ソースコード１０４は、ソフトウェアプログラムに使用され得る任意の適切なタイプのコンピュータ言語で書かれてもよい。

【００１９】

いくつかの実施形態において、ソースコード１０４は、ソフトウェアプログラムのコードの複数のイテレーションを含んでもよい。例えば、アップデート、パッチ、プラットフォーム移行、バグフィックス、表面的な再配置等として、ソースコード１０４に対して複数の変更がなされることがある。ゆえに、複数の変更は、ソースコード１０４の２つ以上の異なるイテレーションを結果としてもたらし得る。これら又は他の実施形態において、ソースコード１０４は、ソフトウェアプログラムのコードのそのようなイテレーションのうち２つ以上を含むことがあり、イテレーション間でなされた変更に起因して、異なるイテレーション間に１つ以上の差異が存在し得る。さらに、本開示におけるソースコードに対する「変更」への参照は、１つ以上のコード行に対してなされ得る任意数の修正を含んでもよい。例えば、変更は、単一のコード行における単一の要素名変更、要素追加、又は要素削除のような簡素なものから、多数のコード行を導入又は削除するような複雑なものに及んでもよい。

【００２０】

解析モジュール１０６は、コンピューティングデバイスがソースコード１０４に関して１つ以上の動作を実行して推論イベント１０８を取得することを可能にするように構成されたコード及びルーチンを含んでもよい。さらに又は代わりに、解析モジュール１０６は、プロセッサ、マイクロプロセッサ（例えば、１つ以上の動作を実行し又はその実行を制御するため）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、又は特定用途向け集積回路（ＡＳＩＣ）を含むハードウェアを使用して実現されてもよい。いくつかの他の例において、解析モジュール１０６は、ハードウェアとソフトウェアの組み合わせを使用して実現されてもよい。本開示において、解析モジュール１０６により実行されると記載された動作は、解析モジュール１０６が対応するシステムに実行するよう指示し得る動作を含んでもよい。

【００２１】

解析モジュール１０６は、ソースコード１０４の複数のイテレーションを含むソースコード１０４を取得するように構成されてもよい。いくつかの実施形態において、解析モジュール１０６は、ソースコード１０４をビッグコードデータベース１０２から取得するように構成されてもよい。ビッグコードデータベース１０２は、既存のソフトウェアプログラム及びそれらのそれぞれのソースコードのリポジトリであり得る。これら又は他の実施形態において、ビッグコードデータベース１０２は、対応するソースコードの異なるイテレーションを含んでもよい。さらに又は代わりに、ビッグコードデータベース１０２は解析に対して利用可能にされてもよく、これは、他のソフトウェアプログラムの開発及びデバッグを容易にするのに役立つ可能性がある。

【００２２】

解析モジュール１０６は、ソースコード１０４に関するイベント対応（event correspondences）を決定するために使用され得るソースコード１０４に関する一連の動作を実行するように構成されてもよい。いくつかの実施形態において、各イベント対応は、ソースコード１０４に対してなされたそれぞれの変更と、ソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示し得る。これら又は他の実施形態におい

て、解析モジュール１０６は、推論イベント１０８を出力するように構成されてもよい。
【００２３】
上述のように、推論イベント１０８は、ソフトウェアプログラムに関して生じ、かつソースコード１０４の異なるイテレーション間でソースコード１０４に対してなされた変更により引き起こされたか又は該変更の原因であったイベントを含んでもよい。例として、推論イベント１０８は、エラー訂正、エラー導入、プラットフォーム移行、追加された挙動、除去された挙動、ソースコード１０４のリファクタリング等を含んでもよい。
【００２４】
いくつかの実施形態において、解析モジュール１０６は、ソースコード１０４に対してなされた変更に関してイベントタイプ推論動作を実行して、それぞれの変更に対応するイベントタイプを決定するように構成されてもよい。これら又は他の実施形態において、解析モジュール１０６は、推論イベント１０８を、どのイベントタイプがどの変更に対応するかについての指標として出力するように構成されてもよい。例えば、推論イベント１０８は、特定の変更、ソースコード１０４における特定の変更の位置、特定の変更に対応すると決定された特定のイベント、及び特定のイベントと特定の変更との間の対応の指標を含んでもよい。いくつかの実施形態において、推論イベント１０８を決定するために解析モジュール１０６により実行され得るイベントタイプ推論動作は、以下で詳細に論じられる図５～図１１の方法５００、６００、７００、８００、９００、１０００、及び１１００それぞれに関して以下に記載される１つ以上の動作を含んでもよい。
【００２５】
いくつかの実施形態において、環境１００は、コミットメッセージモジュール１１０を含んでもよい。コミットメッセージモジュール１１０は、コンピューティングデバイスが推論イベント１０８に関して１つ以上の動作を実行してコミットメッセージ１１２を取得することを可能にするように構成されたコード及びルーチンを含んでもよい。さらに又は代わりに、コミットメッセージモジュール１１０は、プロセッサ、マイクロプロセッサ（例えば、１つ以上の動作を実行し又はその実行を制御するため）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、又は特定用途向け集積回路（ＡＳＩＣ）を含むハードウェアを使用して実現されてもよい。いくつかの他の例において、コミットメッセージモジュール１１０は、ハードウェアとソフトウェアの組み合わせを使用して実現されてもよい。本開示において、コミットメッセージモジュール１１０により実行されると記載された動作は、コミットメッセージモジュール１１０が対応するシステムに実行するよう指示し得る動作を含んでもよい。
【００２６】
コミットメッセージモジュール１１０は、コミットメッセージ１１２が特定のコード変更がなぜなされたかの理由又は特定のコード変更の影響に関するより多くの洞察を提供するように、コミットメッセージ１１２を生成するように構成されてもよい。例えば、コミットメッセージモジュール１１０は、推論イベント１０８を使用してコミットメッセージ１１２を生成するように構成されてもよく、それにより、コミットメッセージ１１２は、特定の変更、変更に対応する特定のイベント、及び特定のイベントの特定の変更への対応を記述するメッセージを提供する。
【００２７】
例えば、特定のイベントが、特定のタイプのエラーのエラー訂正イベントとして、推論イベント１０８に示されてもよい。さらに、特定の変更が、エラーを訂正するためになされた修正及び修正のソースコード１０４内の位置により、推論イベント１０８に示されてもよい。これら又は他の実施形態において、コミットメッセージモジュール１１０は、特定のタイプのエラーが特定の変更により訂正されたことを記述する特定のコミットメッセージを生成するように構成されてもよい。いくつかの実施形態において、コミットメッセージモジュール１１０は、特定のコミットメッセージをソースコードに、特定の変更の位置において含めるように構成されてもよい。さらに又は代わりに、コミットメッセージモジュール１１０は、特定のコミットメッセージをイベントレポートに含めるように構成さ

れてもよく、イベントレポートの特定のコミットメッセージが、特定の変更の位置を示してもよい。

【００２８】

本開示の範囲から逸脱することなく、図１に対して修正、追加、又は省略がなされてもよい。例えば、環境１００は、本開示において図示及び説明されるものより多くの又は少ない要素を含んでもよい。例えば、いくつかの実施形態において、環境１００は、解析モジュール１０６を含むがコミットメッセージモジュール１１０を含まなくてもよく、他の実施形態において、環境１００は、コミットメッセージモジュール１１０を含むが解析モジュール１０６を含まなくてもよい。さらに、いくつかの実施形態において、解析モジュール１０６及びコミットメッセージモジュール１１０の１つ以上のルーチン、１つ以上の命令、又はコードの少なくとも一部分が、それらが同じ要素とみなされ得るように組み合わせられてもよく、あるいは解析モジュール１０６及びコミットメッセージモジュール１１０の一部とみなされ得る共通セクションを有してもよい。

【００２９】

図２は、本開示に記載される少なくとも１つの実施形態に従って配置された、ソフトウェアプログラムからの二次的な修正の除去に関連した一例示的な環境２００を表す図である。環境２００は、トリミングモジュール２０２を含んでもよく、トリミングモジュール２０２は、ソフトウェアプログラムのソースコード２０４を解析して、ソースコード２０４に対してなされた１つ以上の二次的な修正を除去し、修正されたソースコード２０６を取得するように構成される。

【００３０】

ソースコード２０４は、例えば、ソフトウェアプログラム、ソフトウェアプログラムのコード、ライブラリ、アプリケーション、スクリプト、又は処理デバイスによる実行のための他の論理若しくは命令などの電子データを含んでもよい。いくつかの実施形態において、ソースコード２０４は、ソフトウェアプログラムの完全なインスタンスを含んでもよい。さらに又は代わりに、ソースコード２０４は、ソフトウェアプログラムの一部分を含んでもよい。ソースコード２０４は、ソフトウェアプログラムに使用され得る任意の適切なタイプのコンピュータ言語で書かれてもよい。

【００３１】

いくつかの実施形態において、ソースコード２０４は、ソフトウェアプログラムのコードの複数のイテレーションを含んでもよい。例えば、アップデート、パッチ、プラットフォーム移行、バグフィックス、表面的な再配置等として、ソースコード２０４に対して複数の変更がなされることがある。ゆえに、複数の変更は、ソースコード２０４の２つ以上の異なるイテレーションを結果としてもたらし得る。これら又は他の実施形態において、ソースコード２０４は、ソフトウェアプログラムのコードのそのようなイテレーションのうち２つ以上を含むことがあり、イテレーション間でなされた変更に起因して、異なるイテレーション間に１つ以上の差異が存在し得る。

【００３２】

さらに又は代わりに、いくつかの実施形態において、ソースコード２０４は、開発中か又はテストされているソフトウェアコードを含んでもよい。これら又は他の実施形態において、ソースコード２０４は、開発中か又はテストされているコードの２つ以上のイテレーションを含んでもよい。

【００３３】

トリミングモジュール２０２は、コンピューティングデバイスがソースコード２０４に関して１つ以上の動作を実行して、修正されたソースコード２０６を取得することを可能にするように構成されたコード及びルーチンを含んでもよい。さらに又は代わりに、トリミングモジュール２０２は、プロセッサ、マイクロプロセッサ（例えば、１つ以上の動作を実行し又はその実行を制御するため）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、又は特定用途向け集積回路（ＡＳＩＣ）を含むハードウェアを使用して実現されてもよい。いくつかの他の例において、トリミングモジュール２０２は、ハードウェアと

ソフトウェアの組み合わせを使用して実現されてもよい。本開示において、トリミングモジュール２０２により実行されると記載された動作は、トリミングモジュール２０２が対応するシステムに実行するよう指示し得る動作を含んでもよい。

【００３４】

トリミングモジュール２０２は、ソースコード２０４の複数のイテレーションを含むソースコード２０４を取得するように構成されてもよい。いくつかの実施形態において、トリミングモジュール２０２は、図１のビッグコードデータベース１０２などのビッグコードデータベースからソースコード２０４を取得するように構成されてもよい。

【００３５】

トリミングモジュール２０２は、ソースコード２０４の異なるイテレーションに関して一連の動作を実行して、イテレーション間でなされたどの修正が一次的な（primary）修正及び二次的な修正であるかを決定するように構成されてもよい。上述のように、「二次的な」修正への参照は、ソースコード２０４の根本的な機能性を変更し得ない修正を参照し得る。逆に、「一次的な」修正への参照は、ソースコード２０４の根本的な機能性を変更し得る修正を参照し得る。

【００３６】

これら又は他の実施形態において、トリミングモジュール２０２は、二次的な修正であると決定された修正をソースコード２０４から除去するように構成されてもよい。トリミングモジュールにより出力され得る修正されたソースコード２０６は、決定された二次的な修正が除去されたソースコード２０４を含んでもよい。いくつかの実施形態において、トリミングモジュール２０２は、以下で詳細に論じられる図５～図７の方法５００、６００、及び７００それぞれに関して以下に記載される１つ以上の動作を使用して、修正されたソースコード２０６を生成するように構成されてもよい。

【００３７】

本開示の範囲から逸脱することなく、図２に対して修正、追加、又は省略がなされてもよい。例えば、環境２００は、本開示において図示及び説明されるものより多くの又は少ない要素を含んでもよい。例えば、いくつかの実施形態において、環境２００は、図１の環境１００に含まれてもよい。さらに、いくつかの実施形態において、トリミングモジュール２０２の１つ以上のルーチン、１つ以上の命令、又はコードの少なくとも一部分が、解析モジュール１０６及びコミットメッセージモジュール１１０のうち１つ以上と組み合わせられてもよい。

【００３８】

図３は、本開示に記載される少なくとも１つの実施形態に従って配置された、ソフトウェアプログラムの修復に関連した一例示的な環境３００を表す図である。環境３００は、修復モジュール３０６を含んでもよく、修復モジュール３０６は、テスト中コード（code under test）３０４をエラーについて解析するように構成される。修復モジュール３０６は、修正されたテスト中コード３０８を出力するようにさらに構成されてもよく、修正されたテスト中コード３０８は、修復モジュール３０６により実行された修復動作によりテスト中コード３０４になされた１つ以上の修正を含み得る。

【００３９】

テスト中コード３０４は、ソフトウェアプログラム、ソフトウェアプログラムのソースコード、ライブラリ、アプリケーション、スクリプト、又は処理デバイスによる実行のための他の論理若しくは命令などの電子データを含んでもよい。いくつかの実施形態において、テスト中コード３０４は、ソフトウェアプログラムの完全なインスタンスを含んでもよい。さらに又は代わりに、テスト中コード３０４は、ソフトウェアプログラムの一部分を含んでもよい。テスト中コード３０４は、ソフトウェアプログラムに使用され得る任意の適切なタイプのコンピュータ言語で書かれてもよい。いくつかの実施形態において、テスト中コード３０４は、ソフトウェアプログラムのソースコードの１つ以上のイテレーションを含んでもよい。さらに又は代わりに、テスト中コード３０４は、図１及び図２のソースコード１０４又はソースコード２０４それぞれを含んでもよい。

【００４０】
修復モジュール３０６は、コンピューティングデバイスがテスト中コード３０４の１つ以上の修正を実行して、修正されたテスト中コード３０８を生成することを可能にするように構成されたコード及びルーチンを含んでもよい。さらに又は代わりに、修復モジュール３０６は、プロセッサ、マイクロプロセッサ（例えば、１つ以上の動作を実行し又はその実行を制御するため）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、又は特定用途向け集積回路（ＡＳＩＣ）を含むハードウェアを使用して実現されてもよい。いくつかの他の例において、修復モジュール３０６は、ハードウェアとソフトウェアの組み合わせを使用して実現されてもよい。本開示において、修復モジュール３０６により実行されると記載された動作は、修復モジュール３０６が対応するシステムに実行するよう指示し得る動作を含んでもよい。
【００４１】
修復モジュール３０６は、テスト中コード３０４内の１つ以上のエラーを修復する（訂正するとも呼ばれる）ために使用され得るテスト中コード３０４に関する一連の修復動作を実行するように構成されてもよい。いくつかの実施形態において、修復モジュール３０６は、修復テンプレート３１２及び１つ以上のテストスイート３１１に基づいて、修復動作のうち１つ以上を実行するように構成されてもよい。
【００４２】
修復テンプレート３１２は、任意の適切なタイプの命令又はルーチンを含んでもよく、該命令又はルーチンは、実行されたときに、テスト中コード３０４におけるエラーの存在に応じてテスト中コードに関する１つ以上の修正を実現するように構成されてもよい。修正は、エラーを修復又は修復しようと試みることができるテスト中コード３０４の変更を含んでもよい。本開示において、実行され得る修正は、「修復候補」又は「修復」と呼ばれることがある。
【００４３】
テストスイート３１１は、テスト中コード３０４のテストケースとして機能し得る１つ以上のルーチンを含んでもよい。テストスイート３１１は、テスト中コード３０４が指定された方法で挙動するかどうかを決定するように構成されてもよい。テストスイート３１１は、任意の適切な手法に従って構成されてもよい。
【００４４】
修復モジュール３０６は、テスト中コード３０４に関してテストスイート３１１のうち１つ以上を適用して、テスト中コード３０４における１つ以上のエラー及び対応するエラー位置を検出又は決定するように構成されてもよい。いくつかの実施形態において、修復モジュール３０６は、テストスイート３１１に含まれる１つ以上のテストを実行するように構成されてもよく、これは、テスト実行を実行すると呼ばれることがある。合格する（passes）テスト実行は「合格テスト実行」と呼ばれることがあり、不合格になる（fails）テスト実行は「不合格テスト実行」と呼ばれることがある。いくつかの実施形態において、テスト中コード３０４のエラー位置及び対応するエラーは、エラー位置に現れるコードを不合格テスト実行が実行することに基づいて識別されてもよい。
【００４５】
いくつかの実施形態において、修復モジュール３０６は、図２に関して説明したトリミングモジュール２０２などのトリミングモジュールを含んでもよい。これらの実施形態において、修復モジュール３０６は、トリミングモジュールを使用してエラー位置をより正確に識別するように構成されてもよい。例えば、修復モジュール３０６は、特定のエラー位置が、特定の変更が実現された後に導入された特定のエラーを含むことを識別するように構成されてもよい。いくつかの実施形態において、修復モジュール３０６は、図１の解析モジュール１０６などの解析モジュールを含み、特定のエラーが特定の変更により導入されたことを識別してもよい。さらに又は代わりに、修復モジュール３０６は、特定の変更が特定のエラーを引き起こしたことを、二分手法などの任意の他の適切な手法を使用して識別してもよい。

【００４６】
トリミングモジュールを使用し、修復モジュール３０６は、特定の変更のどの修正が二次的な修正であり得るかを識別するように構成されてもよい。さらに又は代わりに、トリミングモジュールを使用し、修復モジュール３０６は、二次的な修正を除去して、特定のエラーを結果としてもたらした特定の変更の一次的な修正を識別するように構成されてもよい。いくつかの実施形態において、修復モジュール３０６は、図８の方法８００に関して以下に詳細に記載されるように、このような一次的な修正をピンポイントするように構成されてもよい。

【００４７】
いくつかの実施形態において、修復モジュール３０６は、検出されたエラーを修復するためになされ得る潜在的な修正として修復テンプレートから修復候補を取得するように構成されてもよい。いくつかの実施形態において、修復モジュール３０６は、修復候補の修復コードパターンを取得してもよい。例えば、いくつかの実施形態において、修復モジュール３０６は、米国特許出願第１５／８２２，１０６号に記載されているような修復コードパターンを取得するように構成されてもよい。

【００４８】
いくつかの実施形態において、修復モジュール３０６は、共通コードパターンデータベース３１４にアクセスするように構成されてもよい。共通コードパターンデータベース３１４は、ビッグコードから導出され得る共通コードパターンを含み得る。例えば、共通コードパターンは、既存のソフトウェアプログラムの１つ以上のリポジトリに記憶され得る既存のソフトウェアプログラムの既存のコードから導出されてもよい。いくつかの実施形態において、共通コードパターンは、米国特許出願第１５／８２２，１０６号に記載されているように取得されてもよい。

【００４９】
さらに又は代わりに、いくつかの実施形態において、既存のソフトウェアプログラムの既存のソースコードは、図２のトリミングモジュール２０２により取得され、既存のソースコードに含まれ得る二次的な修正を除去して、修正された既存のソースコードを生成してもよい。これら又は他の実施形態において、共通コードパターンは、修正された既存のソースコードから取得されてもよく、これは、共通コードパターンの識別の効率を向上させる可能性がある。

【００５０】
いくつかの実施形態において、修復モジュール３０６は、米国特許出願第１５／８２２，１０６号に記載されているように修復候補のコードパターンと共通コードパターンとの間の関連づけに基づいてエラーの修正のために修復候補を選択又は優先順位づけするように構成されてもよい。

【００５１】
さらに又は代わりに、いくつかの実施形態において、修復モジュール３０６は、ビッグコード推論イベント３１６を取得するように構成されてもよい。ビッグコード推論イベント３１６は、ビッグコードでなされた変更に関して推論されるイベントであり得る。いくつかの実施形態において、ビッグコード推論イベントは、図１の推論イベント１０８に類似してもよい。これら又は他の実施形態において、修復モジュールは、ビッグコード推論イベントに基づいて修復候補を優先順位づけ又は選択するように構成されてもよい。

【００５２】
例えば、ビッグコード推論イベントは、特定のタイプのエラーの訂正のイベントに対応する特定の変更を含んでもよい。さらに、エラー検出が、特定のエラータイプのものであるテスト中コード３０４の特定のエラーを識別してもよい。いくつかの実施形態において、修復モジュール３０６は、特定の変更に類似する修復候補を選択又は優先順位づけするように構成されてもよい。

【００５３】
いくつかの実施形態において、修復モジュール３０６はまた、修復候補の優先順位づけ

に基づいてテスト中コード３０４に対する修復を実行するように構成されてもよい。例えば、修復モジュール３０６は、潜在的な修復として、修復候補を最も高い優先順位から最も低い優先順位まで降順に実現してもよい。さらに又は代わりに、修復モジュール３０６は、修正されたテスト中コード３０８を出力するように構成されてもよく、修正されたテスト中コード３０８は、修復候補の優先順位づけに基づいて実現され得る１つ以上の修復を含み得る。

【００５４】

上述のように、いくつかの実施形態において、修復モジュール３０６は、解析モジュールを含み、テスト中コード３０４の異なるイテレーションに関して生じ得る１つ以上の推論イベントを識別してもよい。これら又は他の実施形態において、修復モジュール３０６は、図１のコミットメッセージモジュール１１０などのコミットメッセージモジュールを含んでもよい。いくつかの実施形態において、修復モジュール３０６は、図１に関して上述したように、コミットメッセージモジュール及び推論イベントを使用して、テスト中コード３０４に関してコミットメッセージを生成するように構成されてもよい。いくつかの実施形態において、テスト中コード３０４に対応するコミットメッセージは、修正テスト中コード３０８の生成においてテスト中コード３０４の開発及びテストに使用されてもよい。

【００５５】

本開示の範囲から逸脱することなく、図３に対して修正、追加、又は省略がなされてもよい。例えば、環境３００は、本開示において図示及び説明されるものより多くの又は少ない要素を含んでもよい。さらに、いくつかの実施形態において、修復モジュール３０６、テストスイート３１１、及び修復テンプレート３１２の１つ以上のルーチン、１つ以上の命令、又はコードの少なくとも一部分が、それらが同じ要素とみなされ得るように組み合わせられてもよく、あるいは修復モジュール３０６、テストスイート３１１、及び修復テンプレート３１２のうち２つ以上の一部とみなされ得る共通セクションを有してもよい。

【００５６】

図４は、本開示の少なくとも１つの実施形態による、一例示的なコンピューティングシステム４０２のブロック図を示す。コンピューティングシステム４０２は、解析モジュール（例えば、図１の解析モジュール１０６）、コミットメッセージモジュール（例えば、図１のコミットメッセージモジュール１１０）、トリミングモジュール（例えば、図２のトリミングモジュール２０２）、及び／又は修復モジュール（例えば、図３の修復モジュール３０６）に関連づけられた１つ以上の動作を実現又は指示するように構成され得る。コンピューティングシステム４０２は、プロセッサ４５０、メモリ４５２、及びデータ記憶装置４５４を含んでもよい。プロセッサ４５０、メモリ４５２、及びデータ記憶装置４５４は、通信上結合されてもよい。

【００５７】

一般に、プロセッサ４５０は、種々のコンピュータハードウェア又はソフトウェアモジュールを含む、任意の適切な専用若しくは汎用コンピュータ、コンピューティングエンティティ、又は処理デバイスを含んでもよく、任意の適用可能なコンピュータ読取可能記憶媒体に記憶された命令を実行するように構成されてもよい。例えば、プロセッサ４５０は、マイクロプロセッサ、マイクロコントローラ、デジタル信号プロセッサ（ＤＳＰ）、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、又はプログラム命令を解釈及び／又は実行するよう及び／又はデータを処理するように構成された任意の他のデジタル若しくはアナログ回路を含んでもよい。図４において単一のプロセッサとして示されているが、プロセッサ４５０は、本開示において説明された任意数の動作を個々又は集合的に実行し又はその実行を指示するように構成された任意数のプロセッサを含んでもよい。さらに、プロセッサのうち１つ以上が、異なるサーバなどの１つ以上の異なる電子デバイス上に存在してもよい。

【００５８】

いくつかの実施形態において、プロセッサ４５０は、メモリ４５２、データ記憶装置４

５４、又はメモリ４５２及びデータ記憶装置４５４に記憶されたプログラム命令を解釈及び／又は実行し、かつ／あるいはデータを処理するように構成されてもよい。いくつかの実施形態において、プロセッサ４５０は、データ記憶装置４５４からプログラム命令を取り出し、プログラム命令をメモリ４５２にロードすることができる。

【００５９】

例えば、いくつかの実施形態において、上述のモジュール（例えば、解析モジュール、コミットメッセージモジュール、トリミングモジュール、及び／又は修復モジュール）のうち１つ以上が、プログラム命令としてデータ記憶装置４５４に含まれてもよい。プロセッサ４５０は、データ記憶装置４５４から対応するモジュールのプログラム命令を取り出すことができ、対応するモジュールのプログラム命令をメモリ４５２にロードすることができる。対応するモジュールのプログラム命令がメモリ４５２にロードされた後、プロセッサ４５０はプログラム命令を実行することができ、それにより、コンピューティングシステムは、命令により指示されるとおり対応するモジュールに関連づけられた動作を実現できる。

【００６０】

メモリ４５２及びデータ記憶装置４５４は、記憶されたコンピュータ実行可能命令又はデータ構造を搬送し又は有するコンピュータ読取可能記憶媒体を含むことができる。そのようなコンピュータ読取可能記憶媒体は、プロセッサ４５０などの汎用又は専用コンピュータによりアクセスされ得る任意の利用可能な媒体を含んでもよい。限定でなく例として、そのようなコンピュータ読取可能記憶媒体は、ランダムアクセスメモリ（ＲＡＭ）、読取専用メモリ（ＲＯＭ）、電気的消去可能プログラマブル読取専用メモリ（ＥＥＰＲＯＭ）、コンパクトディスク読取専用メモリ（ＣＤ‐ＲＯＭ）若しくは他の光ディスク記憶装置、磁気ディスク記憶装置若しくは他の磁気記憶デバイス、フラッシュメモリデバイス（例えば、ソリッドステートメモリデバイス）、又はコンピュータ実行可能命令又はデータ構造の形式で特定のプログラムコードを搬送又は記憶するために使用でき、かつ汎用又は専用コンピュータによりアクセスできる任意の他の記憶媒体を含む、有形の又は非一時的なコンピュータ読取可能記憶媒体を含んでもよい。上記の組み合わせもまた、コンピュータ読取可能記憶媒体の範囲内に含まれてもよい。コンピュータ実行可能命令は、例えば、プロセッサ４５０に特定の動作又は動作のグループを実行させるように構成された命令及びデータを含んでもよい。

【００６１】

本開示の範囲から逸脱することなく、コンピューティングシステム４０２に対して修正、追加、又は省略がなされてもよい。例えば、いくつかの実施形態において、コンピューティングシステム４０２は、明示的に図示又は説明されない可能性のある任意数の他のコンポーネントを含んでもよい。

【００６２】

図５は、本開示に記載される少なくとも１つの実施形態による、ソフトウェアプログラムに関して生じるイベントを推論する一例示的な方法５００のフローチャートである。方法５００は、テスト中コードに関して任意の適切なシステム、装置、又はデバイスにより実行されてもよい。例えば、図１の解析モジュール１０６、図１のコミットメッセージモジュール１１０、及び図３の修復モジュール３０６のうち１つ以上、又は（例えば、１つ以上のモジュールにより指示される）図４のコンピューティングシステム４０２が、方法５００に関連づけられた動作のうち１つ以上を実行してもよい。別個のブロックで示されているが、方法５００のブロックのうち１つ以上に関連づけられたステップ及び動作は、特定の実装に依存してさらなるブロックに分割されてもよく、より少ないブロックに組み合わせられてもよく、あるいは消去されてもよい。

【００６３】

方法５００はブロック５０２で開始してもよく、ブロック５０２において、第１のソフトウェアプログラムの第１のソースコードが取得されてもよい。いくつかの実施形態において、第１のソースコードの複数のイテレーションが取得されてもよく、１つ以上の変更

（各々が１つ以上の修正を含み得る）が、イテレーション間で第１のソースコードに関して生じていてもよい。

【００６４】

いくつかの実施形態において、方法５００は、複数のイベント対応を決定することを含んでもよく、各イベント対応は、イテレーション間で第１のソースコードに対してなされたそれぞれの変更間の対応を示す。これら又は他の実施形態において、イベント対応は、イテレーションのうち２つ以上を使用して１つ以上のイベントタイプ推論動作を実行することにより決定されてもよい。いくつかの実施形態において、イベントタイプ推論動作は、以下に記載されるブロック５０４、５０６、５０８、５１０、５１２、５１４、５１６、及び５１８に従って実行されてもよい。

【００６５】

ブロック５０４において、最も古い変更が選択されてもよい。例えば、上述のように、第１のソースコードの各イテレーションは、前のイテレーションに対して１つ以上の変更がなされたことに起因し得る。そのようなものとして、ブロック５０２におけるソースコードの取得されたイテレーションに基づいて、第１のソースコードに対してなされた複数の変更が識別されてもよい。ブロック５０４で、いくつかの実施形態において、識別された変更のうち、最も古い変更、又は換言すれば他の変更のすべての前になされた変更である変更が、選択されてもよい。

【００６６】

いくつかの実施形態において、方法５００は、ブロック５０６において、選択された変更に関して不良導入（fault introduction）イベント推論動作を実行することを含んでもよい。不良導入イベント推論動作は、選択された変更が第１のソースコードに不良を導入したかどうかを決定するために使用され得る。そのようなものとして、いくつかの実施形態において、ブロック５０６で、不良導入イベントが選択された変更に対応するかどうかが決定されてもよい。

【００６７】

例えば、いくつかの実施形態において、ブロック５０６で、１つ以上のソフトウェアテストが、第１のソースコードに関して、選択された変更が第１のソースコードに含まれないで実行されてもよい。さらに、同じソフトウェアテストが、第１のソースコードに関して、選択された変更が第１のソースコードに含まれて実行されてもよい。いくつかの例において、第１のソースコードの第１のソフトウェアテストが、選択された変更が第１のソースコードに含まれないで合格した可能性があり、選択された変更が第１のソースコードに含まれて不合格になった可能性がある。いくつかの実施形態において、第１のソフトウェアテストは、それが合格し次いで不合格になったことに基づいて識別されてもよい。これら又は他の実施形態において、不良導入イベントが選択された変更に対応することは、第１のソフトウェアテストをそれが合格し次いで不合格になったことに基づいて識別することに応じて、決定されてもよい。いくつかの実施形態において、不良導入イベントが選択された変更に対応するかどうかに関する決定は、以下で論じられるように、図６の方法６００の１つ以上の動作を実行することによりなされもよい。

【００６８】

いくつかの実施形態において、方法５００は、ブロック５０８において、選択された変更に関して不良訂正（fault correction）イベント推論動作を実行することを含んでもよい。不良訂正イベント推論動作は、選択された変更が第１のソースコード内の不良を訂正したかどうかを決定するために使用され得る。そのようなものとして、いくつかの実施形態において、ブロック５０８で、不良訂正イベントが選択された変更に対応するかどうかが決定されてもよい。

【００６９】

例えば、いくつかの実施形態において、ブロック５０８で、ブロック５０６に関して説明したのと同様に、１つ以上のソフトウェアテストが、第１のソースコードに関して、選択された変更が第１のソースコードに含まれないで実行されてもよい。さらに、同じソフ

トウェアテストが、第１のソースコードに関して、選択された変更が第１のソースコードに含まれて実行されてもよい。いくつかの例において、第１のソースコードの第２のソフトウェアテストが、選択された変更が第１のソースコードに含まれないで不合格になった可能性があり、選択された変更が第１のソースコードに含まれて合格した可能性がある。いくつかの実施形態において、第２のソフトウェアテストは、それが不合格になり次いで合格したことに基づいて識別されてもよい。これら又は他の実施形態において、不良訂正イベントが選択された変更に対応することは、第２のソフトウェアテストをそれが不合格になり次いで合格したことに基づいて識別することに応じて、決定されてもよい。いくつかの実施形態において、不良訂正イベントが選択された変更に対応するかどうかに関する決定は、以下で詳細に論じられるように、図６の方法６００の１つ以上の動作を実行することによりなされてもよい。

【００７０】

いくつかの実施形態において、方法５００は、ブロック５１０において、選択された変更に関して欠陥導入（defect introduction）イベント推論動作を実行することを含んでもよい。欠陥導入イベント推論動作は、選択された変更が第１のソースコードに欠陥を導入したかどうかを決定するために使用され得る。そのようなものとして、いくつかの実施形態において、ブロック５１０で、欠陥導入イベントが選択された変更に対応するかどうかが決定されてもよい。本開示において、用語「欠陥」及び「不良」の使用は、双方、ソースコードにおいて見つけられ得るエラーを参照し得る。本開示においては、エラーが如何にして識別され得るかを区別するために、異なる用語が図５、図６及び図７に関して、及び請求項に関して使用される。特に、「不良」は、テストスイートを使用するエラー導入イベント及びエラー訂正イベントを識別する文脈において使用される。さらに、「欠陥」は、静的解析を使用するエラー導入イベント及びエラー訂正を識別する文脈において使用される。

【００７１】

例えば、いくつかの実施形態において、ブロック５１０で、第１の静的解析が、第１のソースコードに関して、選択された変更が第１のソースコードに含まれないで実行されてもよい。さらに、第２の静的解析が、第１のソースコードに関して、選択された変更が第１のソースコードに含まれて実行されてもよい。いくつかの例において、第１の欠陥が、第２の静的解析から識別され得る。さらに、第１の欠陥は、第１の静的解析から識別されなかった可能性がある。これら又は他の実施形態において、欠陥導入イベントが選択された変更に対応することは、第１の欠陥が第２の静的解析から識別されるが第１の静的解析から識別されないことに応じて、決定されてもよい。いくつかの実施形態において、欠陥導入イベントが選択された変更に対応するかどうかに関する決定は、以下で詳細に論じられるように、図７の方法７００の１つ以上の動作を実行することによりなされてもよい。

【００７２】

いくつかの実施形態において、方法５００は、ブロック５１２において、選択された変更に関して欠陥訂正（defect correction）イベント推論動作を実行することを含んでもよい。欠陥訂正イベント推論動作は、選択された変更が第１のソースコード内の欠陥を訂正したかどうかを決定するために使用され得る。そのようなものとして、いくつかの実施形態において、ブロック５１２で、欠陥訂正イベントが選択された変更に対応するかどうかが決定されてもよい。

【００７３】

例えば、いくつかの実施形態において、ブロック５１２で、ブロック５１４と同様に、第１の静的解析が、第１のソースコードに関して、選択された変更が第１のソースコードに含まれないで実行されてもよい。さらに、第２の静的解析が、第１のソースコードに関して、選択された変更が第１のソースコードに含まれて実行されてもよい。いくつかの例において、第２の欠陥が、第１の静的解析から識別され得る。さらに、第２の欠陥は、第２の静的解析から識別されなかった可能性がある。これら又は他の実施形態において、欠陥訂正イベントが選択された変更に対応することは、第２の欠陥が第１の静的解析から識

別されるが第２の静的解析から識別されないことに応じて、決定されてもよい。いくつかの実施形態において、欠陥訂正イベントが選択された変更に対応するかどうかに関する決定は、以下で詳細に論じられるように、図７の方法７００の１つ以上の動作を実行することによりなされてもよい。

【００７４】

ブロック５０６、５０８、５１０、及び５１２のうち１つ以上で実行される１つ以上の動作に続いて、別の変更が選択されてもよい。例えば、ブロック５０４において、最も古い変更は、上記で論じられたように選択された可能性がある。いくつかの実施形態において、ブロック５１４において、解析されていない、より新しい変更があるかどうかが決定されてもよく、該より新しい変更は、ブロック５０４において前に選択された変更より新しい。いくつかの実施形態において、より新しい変更があることに応じて、該より新しい変更が選択されてもよく、ブロック５０６、５０８、５１０、及び５１２のうち１つ以上の動作のうち１つ以上が、新たに選択された変更に関して実行されてもよい。いくつかの実施形態において、新たに選択された変更は、新たに選択された変更が前に選択された変更と比較して次の最も古い変更であることに応じて選択されてもよい。いくつかの実施形態において、ブロック５０６、５０８、５１０、５１２、及び５１４に関する上記の動作は、あらゆる変更がブロック５０６、５０８、５１０、及び５１２のうち１つ以上に関して解析されるまで繰り返されてもよい。

【００７５】

いくつかの実施形態において、上記に対して修正がなされてもよい。例えば、いくつかの例において、特定の変更に関してブロック５０６、５０８、５１０、又は５１２のうちの１つのみに関して動作が実行されてもよい。さらに又は代わりに、ブロック５０６、５０８、５１０、又は５１２のうちの１つに関して動作が実行されてもよく、特定のブロックに関して対応するイベント対応が識別されない場合、方法５００は、ブロック５０６、５０８、５１０、又は５１２のうち別の１つに進んでもよい。さらに又は代わりに、特定の変更についてブロック５０６、５０８、５１０、又は５１２のうち１つに関して特定のイベント対応が決定されることに応じて、そのようなブロックのうち残りは、特定の変更に関してスキップされてもよい。

【００７６】

いくつかの実施形態において、本方法５００は、ブロック５１２において、特定の変更に関してプラットフォーム移行イベント推論動作を実行することを含んでもよい。図５の図示された例示的な実装において、特定の変更は、第１のソースコードに対してなされた最も新しい（又は最も最近の）変更であり得る。さらに又は代わりに、特定の変更は、第１のソースコードに対してなされた可能性のある任意の他の変更であってもよい。

【００７７】

プラットフォーム移行推論動作は、特定の変更が第１のソースコードのプラットフォーム移行に対応するかどうかを決定するために使用され得る。そのようなものとして、いくつかの実施形態において、ブロック５１６で、プラットフォーム移行イベントが特定の変更に対応するかどうかが決定されてもよい。

【００７８】

プラットフォーム移行は、第１のソースコードが異なるプラットフォーム又はプラットフォームの異なるバージョンと共に使用され得るように第１のソースコードに対してなされた修正を含んでもよい。例えば、プラットフォーム移行イベントは、第１のプラットフォームから第２のプラットフォームへの移行を含んでもよい。さらに又は代わりに、プラットフォーム移行イベントは、特定のプラットフォームの第１のバージョンから特定のプラットフォームの第２のバージョンへの移行を含んでもよい。プラットフォームは、第１のソースコードをコンパイルするために使用されるコンパイラ、第１のソースコードに関して使用されるライブラリ、第１のソフトウェアプログラムを含むソフトウェアプログラムを実行するオペレーティングシステムを含んでもよい。いくつかの実施形態において、プラットフォーム移行イベントが特定の変更に対応するかどうかに関する決定は、以下で

詳細に論じられるように、図７の方法７００の１つ以上の動作を実行することによりなされてもよい。いくつかの実施形態において、ブロック５１６の動作は、第１のソースコードに対応し得る各々のあり得る異なるプラットフォームについて繰り返されてもよい。

【００７９】

ブロック５１６に続いて、別の変更が選択されてもよい。例えば、ブロック５１４において、最も新しい変更は、プラットフォーム移行イベント推論動作を実行するために選択された可能性がある。いくつかの実施形態において、ブロック５１８において、解析されていない、より古い変更があるかどうかが決定されてもよく、該より古い変更は、ブロック５１４において前に選択された変更より古い。いくつかの実施形態において、より古い変更があることに応じて、該より古い変更が選択されてもよく、ブロック５１６の動作のうちの１つ以上が、新たに選択された変更に関して実行されてもよい。いくつかの実施形態において、新たに選択された変更は、新たに選択された変更がブロック５１４において前に選択された変更と比較して次の最も新しい変更であることに応じて選択されてもよい。いくつかの実施形態において、ブロック５１４、５１６、及び５１８に関する上記の動作は、あらゆる変更がブロック５１６に関して解析されるまで繰り返されてもよい。

【００８０】

いくつかの実施形態において、ブロック５０４、５０６、５０８、５１０、５１２、５１４、５１６、及び５１８で実行され得るイベント対応に基づいて、１つ以上の動作が実行されてもよい。例えば、いくつかの実施形態において、上述したような決定されたイベント対応に基づいて、変更のうち１つ以上の各々についてイベント推論が識別されてもよい。これら又は他の実施形態において、同様に上述したような識別されたイベント推論に基づいて、１つ以上のコミットメッセージが生成されてもよい。

【００８１】

別の例として、いくつかの実施形態において、イベント対応に基づいて、第２のソフトウェアプログラムの第２のソースコードに関して１つ以上の修復動作が実行されてもよい。いくつかの実施形態において、図３のテスト中コード３０４は、第２のソースコードの一例であり得る。

【００８２】

例えば、いくつかの実施形態において、方法５００は、ブロック５２０、５２２、及び５２４を含んでもよい。ブロック５２０において、１つ以上のエラーが、第２のソースコードにおいて識別されてもよい。エラー識別は、第２のソースコードに関して任意の適切な手法を使用して実行されてもよい。例えば、いくつかの実施形態において、エラー識別は、第２のソースコードに関して１つ以上のテストスイートのテスト実行を実行することに基づいて実行されてもよい。

【００８３】

ブロック５２２において、識別されたエラーの修復は、第１のソースコードに関して決定され得るイベント対応に基づいて実行されてもよい。例えば、上述のように、いくつかの実施形態において、イベント対応から取得され得る第１のソースコードに関する推論イベントは、図３に関して上述したビッグコード推論イベントに類似し得る。これら又は他の実施形態において、第１のソースコード推論イベントのうち１つ以上が使用されて、第２のソースコードに対してなされ得る１つ以上の修復候補を識別又は優先順位づけし、図３に関して上述したように、該推論イベントに含まれるものと同様であり得るエラーを訂正してもよい。これら又は他の実施形態において、推論イベントに対応する修復候補及び変更の間の類似度は、修復候補及び変更について決定され得るコードパターンに基づいて決定されてもよい。これら又は他の実施形態において、第１のソースコードに対応する推論イベントに関して生成され得る１つ以上のコミットメッセージが使用されて、推論イベントに含まれる変更と第２のソースコードに対してなされた変更との間の類似度に基づいてエラーのあり得る原因を識別してもよい。

【００８４】

ブロック５２４において、修正された第２のソースコードが出力されてもよい。修正さ

れた第２のソースコードは、上述の修復優先順位づけに基づいて実現され得る１つ以上の修復候補を含み得る修正を含んでもよく、それにより、修正された第２のソースコードは、第２のソースコードの修正されたバージョンを含み得る。

【００８５】

方法５００は、ソフトウェアプログラムのテスト及び修復の効率及び有効性を向上させる可能性がある。例えば、記載された修復候補の優先順位づけの決定は、修復候補の有効性に関するより良い決定を行うのに役立つ。

【００８６】

本開示の範囲から逸脱することなく、方法５００に対して修正、追加、又は省略がなされてもよい。例えば、方法５００の動作は、異なる順序で実現されてもよい。さらに又は代わりに、２つ以上の動作が同時に実行されてもよい。さらに、概説された動作及びアクションは例として提供されているに過ぎず、動作及びアクションのいくつかは、開示される実施形態の本質を損なうことなく、任意であってもよく、より少ない動作及びアクションに組み合わせられてもよく、あるいはさらなる動作及びアクションへ拡張されてもよい。

【００８７】

例えば、いくつかの実施形態において、イベントタイプ推論動作を行うことに関して記載された動作は、第２のソースコードに関して行われてもよい。さらに又は代わりに、１つ以上の第２のソースコード推論イベントが決定されてもよい。これら又は他の実施形態において、１つ以上の対応する第２のソースコードコミットメッセージが生成されてもよい。いくつかの実施形態において、第２のソースコード推論イベント及び／又はコミットメッセージは、第２のソースコードを修復するために使用されてもよい。例えば、第２のソースコード推論イベントは、第２のソースコードに対してなされた特定の変更に関して不良又は欠陥導入イベントが生じたと示すことがある。ゆえに、いくつかの実施形態において、該特定の変更が識別され、修正されて、第２のソースコードを訂正してもよい。さらに又は代わりに、第１のソースコード推論イベントから、類似の不良又は欠陥を訂正する不良又は欠陥訂正イベントが識別されてもよい。これら又は他の実施形態において、識別された不良又は欠陥訂正イベントに対応する第１のソースコードに対する変更が使用されて、修復候補を選択し、第２のソースコード推論イベントから識別された類似の不良又は欠陥を訂正してもよい。

【００８８】

図６は、本開示に記載される少なくとも１つの実施形態による、不良導入イベント及び不良訂正イベント推論動作を実行する一例示的な方法６００のフローチャートである。いくつかの実施形態において、方法６００は、ソフトウェアプログラムのソースコードの特定の変更が不良導入イベント又は不良訂正イベントに対応し得るかどうかを決定するために実行されてもよい。さらに、上述のように、いくつかの実施形態において、図５の方法５００に関して上述したブロック５０６及び５０８の動作のうち１つ以上が、方法６００に従って実行されてもよい。そのようなものとして、いくつかの実施形態において、図５の方法５００に関して上記で論じられた選択された変更は、方法６００の説明で参照される特定の変更であり得る。さらに又は代わりに、方法６００の説明で参照されるソースコードは、図５の方法５００に関して上記で論じられた第１のソースコード又は第２のソースコードであり得る。

【００８９】

方法６００は、テスト中コードに関して任意の適切なシステム、装置、又はデバイスにより実行されてもよい。例えば、図１の解析モジュール１０６及び図３の修復モジュール３０６のうち１つ以上、又は（例えば、１つ以上のモジュールにより指示される）図４のコンピューティングシステム４０２が、方法６００に関連づけられた動作のうち１つ以上を実行してもよい。別個のブロックで示されているが、方法６００のブロックのうち１つ以上に関連づけられたステップ及び動作は、特定の実装に依存してさらなるブロックに分割されてもよく、より少ないブロックに組み合わせられてもよく、あるいは消去されてもよい。

【００９０】
方法６００はブロック６０２で開始してもよく、ブロック６０２において、１つ以上のソフトウェアテストが、ソースコードに関して、特定の変更がソースコードに含まれないで実行されてもよい。いくつかの実施形態において、１つ以上のソフトウェアテストは、ソースコードの第１のイテレーションにおいてソースコードに関してテストスイートの第１の実行を実行することにより実行されてもよく、特定の変更はソースコードから省略されている。
【００９１】
ブロック６０４において、同じ１つ以上のソフトウェアテストが、ソースコードに関して、特定の変更がソースコードに含まれて実行されてもよい。いくつかの実施形態において、１つ以上のソフトウェアテストは、ソースコードの第２のイテレーションにおいてソースコードに関してテストスイートの第２の実行を実行することにより実行されてもよく、特定の変更はソースコードに含まれている。
【００９２】
ブロック６０６において、第２のテスト実行に関して何らかの不合格テストがあるかどうかが決定されてもよい。第２のテスト実行に関して何の不合格テストもないことに応じて、方法６００は、ブロック６０６からブロック６１４に進み得る。第２のテスト実行に関して少なくとも１つの不合格テストがあることに応じて、方法６００は、ブロック６０６からブロック６０８に進み得る。ブロック６０８において、第２のテスト実行の特定の不合格テストが選択されてもよい。
【００９３】
ブロック６１０において、第２のテスト実行の特定の不合格テストが第１のテスト実行に関して合格テストであったかどうかが決定されてもよい。第２のテスト実行の特定の不合格テストが第１のテスト実行に関して合格テストであることに応じて、方法６００はブロック６１０からブロック６１２に進み得る。ブロック６１２において、不良導入イベントが特定の変更に対応することが、第２のテスト実行の特定の不合格テストが第１のテスト実行に関して合格テストであることに基づいて決定されてもよい。
【００９４】
ブロック６１２に続いて、方法６００は、ブロック６０６に戻り得る。さらに、ブロック６１０を再び参照し、特定の不合格テストが第１のテスト実行に関して合格テストでないことに応じて、方法６００はブロック６１０からブロック６０６に戻り得る。
【００９５】
ブロック６１０又は６１２からブロック６０６に戻った後、ブロック６０８、６１０、又は６１２に関して解析されていない、第２のテスト実行に関する何らかの他の不合格テストがあるかどうかが決定されてもよい。ブロック６０８、６１０、又は６１２に関して解析されていない、第２のテスト実行に関する他の不合格テストがあることに応じて、方法６００はブロック６０６から再びブロック６０８に進み得る。いくつかの実施形態において、ブロック６０６、６０８、６１０、及び６１２の動作は、第２のテスト実行に関するすべての不合格テストがブロック６０８、６１０、及び６１２に関して解析されるまで繰り返されてもよい。
【００９６】
第２のテスト実行に関して不合格テストがないことに応じて、又は第２のテスト実行に関するすべての不合格テストがブロック６０８、６１０、及び６１２に関して解析されることに応じて、方法６００は、ブロック６０６からブロック６１４に進み得る。
【００９７】
ブロック６１４において、ソースコードの第１のテスト実行に関して何らかの不合格テストがあるかどうかが決定されてもよい。第１のテスト実行に関して何の不合格テストもないことに応じて、方法６００は終了し得る。第１のテスト実行に関して少なくとも１つの不合格テストがあることに応じて、方法６００はブロック６１４からブロック６１６に進み得る。ブロック６１６において、第１のテスト実行の特定の不合格テストが選択され

てもよい。

【００９８】

ブロック６１８において、第１のテスト実行の特定の不合格テストが第２のテスト実行に関して合格テストであったかどうかが決定されてもよい。第１のテスト実行の特定の不合格テストが第２のテスト実行に関して合格テストであることに応じて、方法６００はブロック６１８からブロック６２０に進み得る。ブロック６２０において、不良訂正イベントが特定の変更に対応することが、第１のテスト実行の特定の不合格テストが第２のテスト実行に関して合格テストであることに基づいて決定されてもよい。

【００９９】

ブロック６２０に続いて、方法６００は、ブロック６１４に戻り得る。ブロック６２０からブロック６１４に戻った後、ブロック６１６、６１８、又は６２０に関して解析されていない、第１のテスト実行に関する何らかの他の不合格テストがあるかどうかが決定されてもよい。ブロック６１６、６１８、又は６２０に関して解析されていない、第１のテスト実行に関する他の不合格テストがあることに応じて、方法６００はブロック６１４から再びブロック６１６に進み得る。いくつかの実施形態において、ブロック６１４、６１６、６１８、及び６２０の動作は、第１のテスト実行に関するすべての不合格テストがブロック６１６、６１８、及び６２０に関して解析されるまで繰り返されてもよい。ブロック６１４において、第１のテスト実行に関する全ての不合格テストがブロック６１６、６１８、及び６２０に関して解析されたと決定されることに応じて、方法６００は終了し得る。

【０１００】

本開示の範囲から逸脱することなく、方法６００に対して修正、追加、又は省略がなされてもよい。例えば、方法６００の動作は、異なる順序で実現されてもよい。例えば、いくつかの実施形態において、動作６１４、６１６、６１８、及び６２０は、いくつかの実施形態において動作６０６、６０８、６１０、及び６１２の前に実行されてもよい。別の例として、ブロック６０２及び６０４の動作は、記載されたものと異なる順序で実行されてもよい。さらに又は代わりに、２つ以上の動作が同時に実行されてもよい。例えば、いくつかの実施形態において、動作６０６、６０８、６１０、及び６１２は、いくつかの実施形態において動作６１４、６１６、６１８、及び６２０と同時に実行されてもよい。別の例として、ブロック６０２及び６０４の動作は、同時に実行されてもよい。さらに、概説された動作及びアクションは例として提供されているに過ぎず、動作及びアクションのいくつかは、開示される実施形態の本質を損なうことなく、任意であってもよく、より少ない動作及びアクションに組み合わせられてもよく、あるいはさらなる動作及びアクションへ拡張されてもよい。

【０１０１】

図７は、本開示に記載される少なくとも１つの実施形態による、欠陥導入イベント及び欠陥訂正イベント推論動作を実行する一例示的な方法７００のフローチャートである。いくつかの実施形態において、方法７００は、ソフトウェアプログラムのソースコードの特定の変更が欠陥導入イベント又は欠陥訂正イベントに対応し得るかどうかを決定するために実行されてもよい。さらに、上述のように、いくつかの実施形態において、図５の方法５００に関して上述したブロック５１０及び５１２の動作のうち１つ以上が、方法７００に従って実行されてもよい。そのようなものとして、いくつかの実施形態において、図５の方法５００に関して上記で論じられた選択された変更は、方法７００の説明で参照される特定の変更であり得る。さらに又は代わりに、方法７００の説明で参照されるソースコードは、図５の方法５００に関して上記で論じられた第１のソースコード又は第２のソースコードであり得る。

【０１０２】

方法７００は、テスト中コードに関して任意の適切なシステム、装置、又はデバイスにより実行されてもよい。例えば、図１の解析モジュール１０６及び図３の修復モジュール３０６のうち１つ以上、又は（例えば、１つ以上のモジュールにより指示される）図４の

コンピューティングシステム４０２が、方法７００に関連づけられた動作のうち１つ以上を実行してもよい。別個のブロックで示されているが、方法７００のブロックのうち１つ以上に関連づけられたステップ及び動作は、特定の実装に依存してさらなるブロックに分割されてもよく、より少ないブロックに組み合わせられてもよく、あるいは消去されてもよい。

【０１０３】

方法７００はブロック７０２で開始してもよく、ブロック７０２において、第１の静的解析が、特定の変更をソースコードから省略させたソースコードの第１のイテレーションに関して実行されてもよい。ブロック７０４において、第２の静的解析が、特定の変更を中に含ませたソースコードの第２のイテレーションに関して実行されてもよい。

【０１０４】

ブロック７０６において、第２の静的解析がソースコードの第２のイテレーションに関して何らかの欠陥を識別したかどうかが決定されてもよい。第２の静的解析が第２のイテレーションに関して何の欠陥も識別しないことに応じて、方法７００は、ブロック７０６からブロック７１４に進み得る。第２の静的解析が第２のイテレーションに関して少なくとも１つの欠陥を識別することに応じて、方法７００は、ブロック７０６からブロック７０８に進み得る。ブロック７０８において、第２の静的解析から識別された第２のイテレーションの特定の欠陥が選択されてもよい。

【０１０５】

ブロック７１０において、第２のイテレーションの特定の欠陥が、第１のイテレーションに関して実行された第１の静的解析から識別されたかどうかが決定されてもよい。特定の欠陥が第１の静的解析から識別されないことに応じて、方法７００はブロック７１０からブロック７１２に進み得る。ブロック７１２において、欠陥導入イベントが特定の変更に対応することが、特定の欠陥が第２の静的解析から識別されるが第１の静的解析から識別されないことに基づいて決定されてもよい。

【０１０６】

ブロック７１２に続いて、方法７００はブロック７０６に戻り得る。さらに、ブロック７１０を再び参照し、特定の欠陥が第２の静的解析から、及び第１の静的解析から識別されることに応じて、方法７００はブロック７１０からブロック７０６に戻り得る。

【０１０７】

ブロック７１０又は７１２からブロック７０６に戻った後、ブロック７０８、７１０又は７１２に関して解析されていない、第２のイテレーションに関する第２の静的解析から識別された何らかの他の欠陥があるかどうかが決定されてもよい。ブロック７０８、７１０、又は７１２に関して解析されていない、第２のイテレーションに関する第２の静的解析から識別された他の欠陥があることに応じて、方法７００はブロック７０６から再びブロック７０８に進み得る。いくつかの実施形態において、ブロック７０６、７０８、７１０、及び７１２の動作は、第２のイテレーションに関する第２の静的解析から識別された欠陥がブロック７０８、７１０、及び７１２に関して解析されるまで繰り返されてもよい。

【０１０８】

ブロック７０８、７１０又は７１２に関して解析されていない、第２のイテレーションに関する第２の静的解析から識別されたさらなる欠陥がないことに応じて、方法７００は、ブロック７０６からブロック７１４に進み得る。

【０１０９】

ブロック７１４において、第１の静的解析がソースコードの第１のイテレーションに関して何らかの欠陥を識別したかどうかが決定されてもよい。第１のイテレーションに関して第１の静的解析から識別された欠陥がないことに応じて、方法７００は終了し得る。第１のイテレーションに関して第１の静的解析から識別された少なくとも１つの欠陥があることに応じて、方法７００は、ブロック７１４からブロック７１６に進み得る。ブロック７１６において、第１の静的解析から識別された特定の欠陥が選択されてもよい。

【０１１０】

ブロック７１８において、第１の静的解析から識別された特定の欠陥が、第２のイテレーションに関して実行された第２の静的解析から識別されたかどうかが決定されてもよい。特定の欠陥が第１の静的解析から識別されるが第２の静的解析から識別されないことに応じて、方法７００はブロック７１８からブロック７２０に進み得る。ブロック７２０において、欠陥訂正イベントが特定の欠陥に対応することが、第１の静的解析が特定の欠陥を識別すること及び第２の静的解析が特定の欠陥を識別しないことに基づいて決定されてもよい。

【０１１１】

ブロック７２０に続いて、方法７００はブロック７１４に戻り得る。ブロック７２０からブロック７１４に戻った後、ブロック７１６、７１８、又は７２０に関して解析されていない、第１の静的解析から識別された何らかの他の欠陥があるかどうかが決定されてもよい。ブロック７１６、７１８、又は７２０に関して解析されていない、第１の静的解析から識別された他の欠陥があることに応じて、方法７００はブロック７１４から再びブロック７１６に進み得る。いくつかの実施形態において、ブロック７１４、７１６、７１８、及び７２０の動作は、第１の静的解析から識別された全ての欠陥がブロック７１６、７１８、及び７２０に関して解析されるまで繰り返されてもよい。ブロック７１４において、第１の静的解析から識別された全ての欠陥がブロック７１６、７１８、及び７２０に関して解析されたと決定されることに応じて、方法７００は終了し得る。

【０１１２】

本開示の範囲から逸脱することなく、方法７００に対して修正、追加、又は省略がなされてもよい。例えば、方法７００の動作は、異なる順序で実現されてもよい。例えば、いくつかの実施形態において、動作７１４、７１６、７１８、及び７２０は、いくつかの実施形態において動作７０６、７０８、７１０、及び７１２の前に実行されてもよい。別の例として、ブロック７０２及び７０４の動作は、記載されたものと異なる順序で実行されてもよい。さらに又は代わりに、２つ以上の動作が同時に実行されてもよい。例えば、いくつかの実施形態において、動作７０６、７０８、７１０、及び７１２は、いくつかの実施形態において動作７１４、７１６、７１８、及び７２０と同時に実行されてもよい。別の例として、ブロック７０２及び７０４の動作は、同時に実行されてもよい。さらに、概説された動作及びアクションは例として提供されているに過ぎず、動作及びアクションのいくつかは、開示される実施形態の本質を損なうことなく、任意であってもよく、より少ない動作及びアクションに組み合わせられてもよく、あるいはさらなる動作及びアクションへ拡張されてもよい。

【０１１３】

図８は、本開示に記載される少なくとも１つの実施形態による、ソフトウェアプログラムのソースコードから二次的な修正を除去する一例示的な方法８００のフローチャートである。方法８００は、テスト中コードに関して任意の適切なシステム、装置、又はデバイスにより実行されてもよい。例えば、図２のトリミングモジュール２０２及び図３の修復モジュール３０６のうち１つ以上、又は（例えば、１つ以上のモジュールにより指示される）図４のコンピューティングシステム４０２が、方法８００に関連づけられた動作のうち１つ以上を実行してもよい。別個のブロックで示されているが、方法８００のブロックのうち１つ以上に関連づけられたステップ及び動作は、特定の実装に依存してさらなるブロックに分割されてもよく、より少ないブロックに組み合わせられてもよく、あるいは消去されてもよい。

【０１１４】

方法８００はブロック８０２で開始してもよく、ブロック８０２において、抽象構文木（abstract syntax trees、ＡＳＴ）が、第１のソフトウェアプログラムの第１のソースコードの複数のイテレーションから生成されてもよい。いくつかの実施形態において、第１のソースコードの複数のイテレーションが取得されてもよく、第１のソースコードに関してイテレーション間で１つ以上の変更（各々が１つ以上の変更を含み得る）が生じた可能性がある。いくつかの実施形態において、第１のソースコードの第１のイテレーショ

ンが取得されてもよく、第１のイテレーションは、第１のソースコードの特定の部分における特定の変更を除外する（excludes）。さらに又は代わりに、第１のソースコードの第２のイテレーションが取得されてもよく、第２のイテレーションは、第１のソースコードの特定の部分に特定の変更を含む。いくつかの実施形態において、特定の変更は複数の修正を含んでもよく、該修正のうち１つ以上が二次的な修正であり得る。

【０１１５】

いくつかの実施形態において、ブロック８０２において、第１のＡＳＴが第１のイテレーションに関して生成されてもよい。これら又は他の実施形態において、ブロック８０２において、第２のＡＳＴが第２のイテレーションに関して生成されてもよい。

【０１１６】

例えば、図９Ａは、特定の部分９０４を含むソースコードの第１のイテレーションに関して生成され得る例示的な第１のＡＳＴ９０２を示す。さらに、図９Ａは、ソースコードの第２のイテレーションに関して生成され得る例示的な第２のＡＳＴ９０６を示し、特定の部分９０４に対して複数の修正を含む特定の変更９０８がなされている。

【０１１７】

図８に戻り、ブロック８０４において、ブロック８０２で生成されたＡＳＴは、１つ以上のステートメント部分木（sub-trees）の森（forests）に区分されてもよい。例えば、いくつかの実施形態において、第１のＡＳＴの１つ以上の第１の部分木が識別されてもよい。第１の部分木は、特定の変更に関連づけられた特定の部分に対応することができ、それに従って識別されてもよい。いくつかの実施形態において、第１の部分木は、第１のソースコードに含まれるステートメント呼び出しに従って識別されてもよい。これら又は他の実施形態において、第１のＡＳＴは、識別された第１の部分木に従って区分されてもよい。さらに又は代わりに、第２のＡＳＴの複数の第２の部分木が識別されてもよい。複数の第２の部分木もまた、特定の変更に関連づけられた特定の部分に対応することができ、それに従って識別されてもよい。これら又は他の実施形態において、第２の部分木は、識別された第２の部分木に従って区分されてもよい。いくつかの例において、特定の部分木はＡＳＴ全体であってもよく、それにより、ＡＳＴ全体が特定の部分木として識別されてもよい。

【０１１８】

例えば、図９Ｂは、図９Ａの第１のＡＳＴ９０２から識別され得る例示的な第１の部分木９１０を示す。特定の例において、第１の部分木９１０は、図９Ａの第１のＡＳＴ９０２のすべてであってもよい。図９Ｂはまた、図９Ａの第２のＡＳＴ９０６の第２の部分木であり得る第２の部分木９１２、９１４、及び９１６を示す。

【０１１９】

図８に戻り、ブロック８０６において、識別された森のテキスト表現が生成されてもよい。例えば、第１のテキスト表現が各第１の部分木に関して生成されてもよく、第２のテキスト表現が各第２の部分木に関して生成されてもよい。いくつかの実施形態において、テキスト表現は、それぞれの部分木に対応し得るコード行を含んでもよい。

【０１２０】

例えば、図９Ｃは、第１の部分木９１０に対応する第１のテキスト表現９１８を示す。図９Ａと図９Ｃとの比較により示されるように、第１のテキスト表現９１８は、第１の部分木９１０に対応する特定の部分９０４における第１のソースコードの第１のイテレーションのコード行を含み得る。

【０１２１】

さらに、図９Ｃは、第２の部分木９１２に対応し得る第２のテキスト表現９２０、第２の部分木９１４に対応し得る第２のテキスト表現９２２、及び第２の部分木９１６に対応し得る第２のテキスト表現９２４を示す。

【０１２２】

図８に戻る。ブロック８０８において、テキスト表現間のテキスト差異が決定されて、第２のテキスト表現から、第１のテキスト表現と異なる異なるテキスト表現を識別しても

よい。例えば、いくつかの実施形態において、各第１のテキスト表現と各第２のテキスト表現との間で差異決定が行われて、どの第２のテキスト表現が第１のテキスト表現のいずれとも同じでないかを識別してもよい。第１のテキスト表現のいずれとも同じでない第２のテキスト表現は、第１のイテレーションと第２のイテレーションとの間になされた修正に対応し得る。このような第２のテキスト表現は、異なるテキスト表現として識別されてもよい。

【０１２３】

例として、第１のテキスト表現９１８が第２のテキスト表現９２０、第２のテキスト表現９２２、及び第２のテキスト表現９２４の各々に対して比較されて、第２のテキスト表現９２０、第２のテキスト表現９２２、及び第２のテキスト表現９２４のうちいずれが第１のテキスト表現９１８と異なるかを決定してもよい。比較から行われる差異決定に基づいて、第２のテキスト表現９２０及び第２のテキスト表現９２２が第１のテキスト表現と異なることが決定され得る。さらに、第２のテキスト表現９２４は第１のテキスト表現９１８と同じであることが決定されてもよい。これら又は他の実施形態において、第２のテキスト表現９２０は異なるテキスト表現「Ｃ１」として識別されてもよく、第２のテキスト表現９２２は異なるテキスト表現「Ｃ２」として識別されてもよい。

【０１２４】

図８に戻り、ブロック８１０において、異なるテキスト表現の最小サイズのセットが識別されてもよい。いくつかの実施形態において、異なるテキスト表現の最小サイズのセットは、第１のソースコードに関して実現され得る最小数の異なるテキスト表現として識別されてもよく、それにより、第１のソースコードは、あたかも特定の変更全体が第１のソースコードに関して実現されたように同じイベントに対応する。同じイベントに対応する異なるテキスト表現の最小サイズのセットの識別は、異なるテキスト表現に対応するどの修正が二次的な修正であるかと、いずれが一次的な修正を含み得るかを示すことができる。

【０１２５】

例えば、特定の異なるテキスト表現又は異なるテキスト表現のセットが、特定の変更全体と同じイベントに対応しない場合、そのような異なるテキスト表現は、二次的な修正に対応する可能性が高い。対照的に、特定の異なるテキスト表現又は異なるテキスト表現のセットが、特定の変更全体と同じイベントに対応する場合、そのような異なるテキスト表現は、一次的な修正に対応する可能性が高い。そのようなものとして、どの異なるテキスト表現が同じイベントに対応する最小サイズのセット外であるかを識別することは、トリミングされ得る最大数の二次的な修正を識別し得る。

【０１２６】

いくつかの実施形態において、ブロック８１０の動作は、特定の変更に関してイベント対応決定を実行することを含んでもよい。いくつかの実施形態において、イベント対応決定は、特定のイベントを特定の変更に対応するものとして識別し得る。いくつかの実施形態において、方法５００、６００、及び／又は７００の１つ以上の動作が実行されて、特定の変更に関してイベント対応決定を実行してもよい。例えば、特定の変更の全てに対応する修正が、方法５００、６００、及び／又は７００に関して上記された選択された変更として使用されてもよい。これら又は他の実施形態において、第１のＡＳＴに対応する第１のソースコードの第１のイテレーションは、中に含まれた選択された変更を含まないソースコードのイテレーションとして使用されてもよい。

【０１２７】

これら又は他の実施形態において、ブロック８１０の動作は、異なるテキスト表現の各可能なセットに関してイベント対応決定を実行することを含んでもよく、異なるテキスト表現の各セットは、１つ以上の異なるテキスト表現を含む。いくつかの実施形態において、方法５００、６００、及び／又は７００の１つ以上の動作が実行されて、各可能なセットに関してイベント対応決定を実行してもよい。例えば、各セットに対応し、かつ第１のイテレーションに関し得る修正は、方法５００、６００、及び／又は７００に関して上記された選択された変更に対応し得る。これら又は他の実施形態において、第１のＡＳＴに

対応する第１のソースコードの第１のイテレーションは、方法５００、６００、及び／又は７００に関して記載される中に含まれた選択された変更を含まないソースコードのイテレーションとして使用されてもよい。

【０１２８】

さらに又は代わりに、ブロック８１０の動作は、マッチングセットとして、異なるテキスト表現のセットのうちいずれが特定のイベントに対応するかを識別することを含んでもよい。これら又は他の実施形態において、マッチングセット識別は、可能なセットに関してなされたイベント対応決定に基づいてもよい。さらに、上述のように、マッチングセット識別は、異なるテキスト表現のセットのうちいずれが一次的な修正を含み得るかを識別してもよい。反対に、マッチングセットでない異なるテキスト表現のセットは、二次的な修正のみを含むセットであり得る。

【０１２９】

これら又は他の実施形態において、ブロック８１０の動作は、最小サイズのセットとして、最少数の異なるテキスト表現を含む特定のマッチングセットを識別することを含んでもよい。換言すれば、他のマッチングセットは、一次的な修正に対応し得るが、最小サイズのセットより多くの二次的な修正にも対応し得る。そのようなものとして、最小サイズのセット外である異なるテキスト表現は、二次的な修正に対応するとみなされてもよい。

【０１３０】

例として図９Ｄに関して、上述のように、第１のテキスト表現９１８と第２のテキスト表現９２０、９２２、及び９２４の各々との間の差異決定は、第２のテキスト表現９２０及び９２２を、異なるテキスト表現Ｃ１及びＣ２としてそれぞれ識別し得る。異なるテキスト表現Ｃ１及びＣ２は、異なるテキスト表現の各可能なセットに編成されてもよく、それにより、第１のセットがＣ１のみを含んでもよく（［Ｃ１］）、第２のセットがＣ２のみを含んでもよく（［Ｃ２］）、第３のセットがＣ１及びＣ２を含んでもよい（［Ｃ１，Ｃ２］）。

【０１３１】

これら又は他の実施形態において、イベント「Ｅ１」が、特定の変更９０８に対応するものとして識別されてもよく、特定の変更９０８は、Ｃ１及びＣ２に対応する修正を含む。さらに、一例として、イベントＥ１は、第１のセット［Ｃ１］及び第３のセット［Ｃ１，Ｃ２］に対応するが第２のセット［Ｃ２］に対応しないものとして識別され得る。そのようなものとして、第１のセット［Ｃ１］及び第３のセット［Ｃ１，Ｃ２］は、マッチングセットとして識別されてもよい。さらに、第１のセット［Ｃ１］が第３のセット［Ｃ１，Ｃ２］より少数の異なるテキスト表現を含むため、第１のセット［Ｃ１］は最小サイズのセットであると決定され、最小サイズのセットとして選択されてもよい。

【０１３２】

図８に戻る。ブロック８１２において、最小サイズのセットの識別に基づいて第２のＡＳＴの１つ以上の部分木を除去することにより、修正された第２のＡＳＴが取得されてもよい。特に、上述のように、最小サイズのセット外である異なるテキスト表現は、二次的な修正に対応し得るものであり得る。そのようなものとして、いくつかの実施形態において、最小サイズのセット外である異なるテキスト表現は、二次的な修正に対応する二次的なテキスト表現として識別されてもよい。これら又は他の実施形態において、二次的なテキスト表現に対応する二次的な部分木は、二次的なＡＳＴから除去され得る二次的な部分木として識別されてもよい。

【０１３３】

例として、図９Ｅは、第２の部分木９１４が除去された、修正された第２のＡＳＴ９３０を示す。上述のように、第２の部分木９１４は、最小サイズのセットとして識別された第１のセット［Ｃ１］外である、異なるテキスト表現Ｃ２に対応する。そのようなものとして、第２の部分木９１４は除去されてもよい。注目すべきことに、図示された例において、第２の部分木９１６は除去されておらず、なぜならば、第２の部分木９１６は異なるテキスト表現でないテキスト表現に対応するためである。

【０１３４】
図８に戻り、いくつかの実施形態において、方法８００は、第２のＡＳＴをさらにトリミングするさらなる動作を含んでもよい。例えば、いくつかの実施形態において、方法８００は、前に解析された部分木に関して部分木であり得る部分木の別のセットに関して動作８０６、８０８、８１０、及び８１２を実行することを含んでもよい。いくつかの実施形態において、動作８０６、８０８、８１０、及び８１２は、すべての部分木が１つのレベル（「１」の高さとも呼ばれる）のみを有するまで、さらなる部分木セットに関して繰り返されてもよい。
【０１３５】
例えば、いくつかの実施形態において、方法８００はブロック８１４を含んでもよく、ブロック８１４において、第１のＡＳＴ及び第２のＡＳＴの任意の部分木が「１」より大きい高さを有するかどうかが決定されてもよい。少なくとも１つの部分木が「１」より大きい高さを有することに応じて、方法８００はブロック８１４からブロック８１６に進み得る。
【０１３６】
ブロック８１６において、最大の高さを有する部分木がさらなる部分木に区分されてもよい。例えば、いくつかの実施形態において、各部分木の高さ（すなわち、レベル数）が識別されてもよい。さらに又は代わりに、ＡＳＴの特定の部分木が、他の部分木より大きいレベル数を有するものとして識別されてもよい。これら又は他の実施形態において、特定の部分木はさらなる部分木に区分されてもよい。ブロック８１６に続いて、方法８００はブロック８０６に戻り得る。
【０１３７】
例として、図９Ｅに示すように、第１の部分木９１０は２つのレベル（「２」の高さとも呼ばれる）を有し得、第２の部分木９１２は５つのレベル（「５」の高さとも呼ばれる）を有し得、第２の部分木９１６もまた２つのレベル（「２」の高さとも呼ばれる）を有し得る。そのようなものとして、第２の部分木９１２が最も高いレベルを有することを所与として、第２の部分木はさらなる部分木に分割されてもよい。例えば、図９Ｆに示されるように、図９Ｅの第２の部分木９１２は、さらなる部分木９３２、９３４、９３６、及び９３８に分割されてもよい。第２の部分木９１２のさらなる部分木９３２、９３４、９３６への分割に続いて、最小サイズの変更セットが、上記のようなさらなる部分木９３２、９３４、９３６の対応するテキスト表現に関して決定されてもよい。いくつかの実施形態において、ブロック８０６、８０８、８１０、８１２、８１４、及び８１６の動作は、すべての部分木が１つのレベルのみを有するまで繰り返されてもよい。
【０１３８】
ブロック８１８において、修正された第２のＡＳＴから第１のソースコードの第３のイテレーションが取得されてもよい。例えば、第３のイテレーションは、修正された第２のＡＳＴを使用して第１のソースコードを再生成することにより取得されてもよい。上述のように、修正された第２のＡＳＴは、上記で詳述したように除去された１つ以上の部分を含むことがあり、１つ以上の部分は、第１のソースコードに対してなされた可能性がある二次的な修正に対応し得る。そのようなものとして、第１のソースコードの第３のイテレーションは、第１のソースコードから省略された１つ以上の二次的な修正を有し得る。そのようなものとして、第３のイテレーションの解析は、第３のイテレーションで削除された１つ以上の二次的な修正を含み得る第２のイテレーションの解析より効率的な可能性がある。
【０１３９】
いくつかの実施形態において、第１のソースコードの第３のイテレーションに基づいて、第１のソースコードに関して１つ以上の修復動作が実行されてもよい。例えば、いくつかの実施形態において、特定の変更は、第１のソースコードにエラーを導入し得る。さらに、第１のソースコードの第３のイテレーションに基づいて、特定の部分のうち特定のサブ部分が、エラーを導入する一次的な修正を含むものとして識別され得る。いくつかの実

施形態において、第１のイテレーションを第３のイテレーションに対して比較して差異を識別することにより、特定のサブ部分が識別されてもよい。これら又は他の実施形態において、特定のサブ部分が修正されて、エラーを修復してもよい。特定のサブ部分は、第３のイテレーションに基づいて特定のサブ部分がエラーを導入した一次的な修正に対応すると決定することに応じて、修正されてもよい。そのようなものとして、修復動作は、それらが第２のイテレーションに基づいて実行される場合より指示され、効率的であり得る。

【０１４０】

これら又は他の実施形態において、修復動作は、第２のソースコードに関してテストスイートを実行することに基づいて、第２のソースコードにおける１つ以上のエラーを識別することを含んでもよい。さらに又は代わりに、第２のソースコードのエラーを修復するための１つ以上の修復候補が、第１のソースコードの第３のイテレーションに基づいて識別され、あるいは優先順位づけされてもよい。

【０１４１】

例えば、いくつかの実施形態において、第３のイテレーションは、特定のエラーに対する修復を含むことがあり、第３のイテレーションから、１つ以上のコードパターンが識別されてもよく、１つ以上のコードパターンは、特定のエラーを修復するためになされ得る修正を示す。これら又は他の実施形態において、識別されたコードパターンが使用されて、特定のエラーに関連するか又は特定のエラーと同じ第２のソースコードのエラーに対する修復候補を選択し、あるいは優先順位づけしてもよい。コードパターンを識別するための第３のイテレーションの使用は、役立たないコードパターンと対照的にコードに役立つコードパターンを識別することを困難にし得る二次的な修正を除去することにより、第２のイテレーションより良い可能性がある。

【０１４２】

方法８００は、ソフトウェアプログラムのテスト及び修復の効率及び有効性を向上させる可能性がある。例えば、ＡＳＴ及びテキスト表現の組み合わせの使用は、ＡＳＴだけを使用するより効率的な方法で、しかしまたテキスト解析を使用するだけより効果的な方法で、二次的な修正を識別することを可能にし得る。そのようなものとして、方法８００の動作は、ソフトウェアプログラムに関して解析及びデバッグ動作を実行するように構成されたコンピューティングシステムの有効性及び効率を向上させる可能性がある。さらに、上記で詳述したように、二次的な修正の除去は、コンピューティングシステムによる問題及び解決策のより効率的な解析及び識別を提供するのに役立つ可能性がある。

【０１４３】

本開示の範囲から逸脱することなく、方法８００に対して修正、追加、又は省略がなされてもよい。例えば、方法８００の動作は、異なる順序で実現されてもよい。さらに又は代わりに、２つ以上の動作が同時に実行されてもよい。さらに、概説された動作及びアクションは例として提供されているに過ぎず、動作及びアクションのいくつかは、開示される実施形態の本質を損なうことなく、任意であってもよく、より少ない動作及びアクションに組み合わせられてもよく、あるいはさらなる動作及びアクションへ拡張されてもよい。

【０１４４】

上述のように、本開示に記載される実施形態は、以下でより詳細に論じられるように、種々のコンピュータハードウェア又はソフトウェアモジュールを含む専用又は汎用コンピュータ（例えば、図２のプロセッサ２５０）の使用を含んでもよい。さらに、上述のように、本開示に記載される実施形態は、記憶されたコンピュータ実行可能命令又はデータ構造を搬送し又は有するコンピュータ読取可能媒体（例えば、図２のメモリ２５２又はデータ記憶装置２５４）を使用して実現されてもよい。

【０１４５】

本開示において用いられるとき、用語「モジュール」又は「コンポーネント」は、モジュール又はコンポーネントのアクションを実行するように構成された特定のハードウェア実装、及び／又はコンピューティングシステムの汎用ハードウェア（例えば、コンピュータ読取可能媒体、処理デバイス等）に記憶され及び／又は汎用ハードウェアにより実行さ

れ得るソフトウェアオブジェクト又はソフトウェアルーチンを参照し得る。いくつかの実施形態において、本開示に記載される異なるコンポーネント、モジュール、エンジン、及びサービスが、コンピューティングシステム上で実行するオブジェクト又はプロセスとして（例えば、別個のスレッドとして）実現されてもよい。本開示に記載されるシステム及び方法のいくつかは、一般に、（汎用ハードウェアに記憶され、及び／又は汎用ハードウェアにより実行される）ソフトウェアで実現されるものとして記載されるが、特定のハードウェア実装、又はソフトウェアと特定のハードウェア実装との組み合わせもまた可能であり、企図される。本説明において、「コンピューティングエンティティ」は、本開示において前に定義された任意のコンピューティングシステム、又はコンピューティングシステム上で動作する任意のモジュール又はモジュレートの組み合わせであってもよい。

【０１４６】

本開示において、特に別記の特許請求の範囲（例えば、別記の特許請求の範囲の本文）において用いられる用語は、一般に「開放的」な用語として意図されている（例えば、用語「含んでいる」は、「含んでいるがこれに限定されない」と解釈されるべきであり、用語「有する」は、「少なくとも有する」と解釈されるべきであり、用語「含む」は、「含むがこれに限定されない」と解釈されるべきである、等）。

【０１４７】

さらに、特定数の導入された請求項記載が意図されている場合、そのような意図は請求項に明示的に記載され、そのような記載がない場合、そのような意図は存在しない。例えば、理解の助けとして、以下の別記の特許請求の範囲は、請求項記載を導入するために、導入フレーズ「少なくとも１つの」及び「１つ以上の」の使用を含むことがある。しかしながら、そのようなフレーズの使用は、不定冠詞「一の」（“ａ”又は“ａｎ”）による請求項記載の導入が、同じ請求項が導入フレーズ「１つ以上の」又は「少なくとも１つの」と「一の」などの不定冠詞とを含むときでも、そのような導入された請求項記載を含む任意の特定の請求項を１つのそのような記載のみ含む実施形態に限定することを暗に示すように見なされるべきではない（例えば、「一の」（“ａ”及び／又は“ａｎ”）は「少なくとも１つの」又は「１つ以上の」を意味するよう解釈されるべきである）。請求項記載を導入するために用いられる定冠詞の使用についても同様である。

【０１４８】

さらに、特定数の導入された請求項記載が明示的に記載されている場合であっても、当業者は、そのような記載は少なくとも記載された数を意味するよう解釈されるべきであることを認識するであろう（例えば、他の修飾語を伴わない「２つの記載」というただそれだけの記載は、少なくとも２つの記載、又は２つ以上の記載を意味する）。さらに、「Ａ、Ｂ、及びＣ等のうち少なくとも１つ」又は「Ａ、Ｂ、及びＣ等のうち１つ以上」と類似の規定が用いられている例において、一般に、そのような構造は、Ａ単独、Ｂ単独、Ｃ単独、Ａ及びＢ共に、Ａ及びＣ共に、Ｂ及びＣ共に、又はＡ、Ｂ、及びＣ共に等を含むことが意図される。

【０１４９】

さらに、明細書においてか、特許請求の範囲においてか、又は図面においてかにかかわらず、２つ以上の代替的な用語を提示するいかなる分離的なワード又はフレーズも、用語のうち１つ、用語のうちいずれか、又は双方の用語を含む可能性を考慮するよう理解されるべきである。例えば、フレーズ「Ａ又はＢ」は、「Ａ」又は「Ｂ」又は「Ａ及びＢ」の可能性を含むよう理解されるべきである。

【０１５０】

本開示に記載された全ての例及び条件付き言語は、本開示と発明者が当該分野を促進するために寄与した概念とを理解する際に読者を助けるための教育的目的を意図されており、そのような具体的に記載された例及び条件への限定を伴わないとみなされるべきである。本開示の実施形態が詳細に記載されたが、本開示の主旨及び範囲から逸脱することなく、種々の変更、置換、及び改変をこれに行うことができる。

上記の実施形態につき以下の付記を残しておく。

（付記１）
複数のイベント対応を決定し、各イベント対応は、第１のソフトウェアプログラムの第１のソースコードに対してなされた複数の変更のうちそれぞれの変更と前記第１のソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示すステップであり、前記複数のイベント対応は、複数のイベントタイプ推論動作のうち２つ以上のイベントタイプ推論動作を実行することにより決定され、前記複数のイベントタイプ推論動作は、
不良導入イベントが前記複数の変更のうち第１の変更に対応することを、前記第１の変更が前記第１のソースコードに含まれないで合格し、かつ前記第１の変更が前記第１のソースコードに含まれて不合格になった前記第１のソースコードの第１のソフトウェアテストを識別することに基づいて、決定することと、
不良訂正イベントが前記複数の変更のうち第２の変更に対応することを、前記第２の変更が前記第１のソースコードに含まれないで不合格になり、かつ前記第２の変更が前記第１のソースコードに含まれて合格した前記第１のソースコードの第２のソフトウェアテストを識別することに基づいて、決定することと、
欠陥導入イベントが前記複数の変更のうち第３の変更に対応することを、前記第３の変更が前記第１のソースコードに含まれないで前記第１のソースコードに対して実行された第１の静的解析から第１の欠陥が識別されないことに基づいて、及び前記第３の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第２の静的解析から前記第１の欠陥が識別されることに基づいて、決定することと、
欠陥訂正イベントが前記複数の変更のうち第４の変更に対応することを、前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第３の静的解析から第２の欠陥が識別されることに基づいて、及び前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第４の静的解析から前記第２の欠陥が識別されないことに基づいて、決定することと、
特定のプラットフォームの第１のバージョンから前記特定のプラットフォームの第２のバージョンへのプラットフォーム移行イベントが前記複数の変更のうち第５の変更に対応することを、前記第５の変更が中に含まれた前記第１のソースコードの第１のビルドが、前記第５の変更が中に含まれた前記第１のソースコードの第２のビルドに関して省略されるエラーを有することに基づいて、決定することであり、前記第１のビルドは前記特定のプラットフォームの前記第１のバージョンを使用して実行され、前記第２のビルドは前記特定のプラットフォームの前記第２のバージョンを使用して実行される、ことと、
を含む、ステップと、
第２のソフトウェアプログラムの第２のソースコードにおける１つ以上のエラーを、前記第２のソースコードに関してテストスイートを実行することに基づいて識別するステップと、
前記第１のソースコードから決定された前記複数のイベント対応を使用して前記１つ以上のエラーに関して前記第２のソースコードに対して修復動作を実行するステップと、
を含む方法。
（付記２）
前記複数の変更のうち特定の変更についてのコミットメッセージを生成し、前記コミットメッセージは前記特定の変更に関して決定されたそれぞれのイベント対応を示すステップ、をさらに含む付記１に記載の方法。
（付記３）
前記不良導入イベントが前記第１の変更に対応すると決定することは、
前記第１の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第１のソフトウェアテストを含む、ことと、
前記第１の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、

前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別することと、
前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することと、
前記不良導入イベントが前記第１の変更に対応することを、前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別すること及び前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することに応じて決定することと、
を含む、付記１に記載の方法。
（付記４）
前記不良訂正イベントが前記第２の変更に対応すると決定することは、
前記第２の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第２のソフトウェアテストを含む、ことと、
前記第２の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、
前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別することと、
前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別することと、
前記不良訂正イベントが前記第２の変更に対応することを、前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別すること及び前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別することに応じて決定することと、
を含む、付記１に記載の方法。
（付記５）
前記欠陥導入イベントが前記第３の変更に対応すると決定することは、
前記第３の変更を含む前記第１のソースコードの部分に関して前記第２の静的解析を実行することであり、前記第２の静的解析は前記第１の欠陥を識別する、ことと、
前記第３の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第１の静的解析を実行することであり、前記第１の静的解析は前記第１の欠陥を識別しない、ことと、
前記欠陥導入イベントが前記第３の変更に対応することを、前記第２の静的解析が前記第１の欠陥を識別すること及び前記第１の静的解析が前記第１の欠陥を識別しないことに応じて決定することと、
を含む、付記１に記載の方法。
（付記６）
前記欠陥訂正イベントが前記第４の変更に対応すると決定することは、
前記第４の変更を含む前記第１のソースコードの部分に関して前記第３の静的解析を実行することであり、前記第３の静的解析は前記第２の欠陥を識別しない、ことと、
前記第４の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第４の静的解析を実行することであり、前記第４の静的解析は前記第２の欠陥を識別する、ことと、
前記欠陥訂正イベントが前記第４の変更に対応することを、前記第４の静的解析が前記第２の欠陥を識別すること及び前記第３の静的解析が前記第２の欠陥を識別しないことに応じて決定することと、
を含む、付記１に記載の方法。
（付記７）
前記プラットフォーム移行イベントが前記第５の変更に対応すると決定することは、
前記第５の変更を含む前記第１のソースコードの部分に関して前記第１のビルドを実行

することであり、前記第１のビルドは前記エラーを識別する、ことと、
前記第５の変更を中に含ませた前記第１のソースコードの前記部分に関して前記第２のビルドを実行することであり、前記第２のビルドは前記エラーを識別しない、ことと、
前記プラットフォーム移行イベントが前記第５の変更に対応することを、前記第１のビルドが前記エラーを識別すること及び前記第２のビルドが前記エラーを識別しないことに応じて決定することと、
を含む、付記１に記載の方法。
（付記８）
命令を記憶するように構成された１つ以上の非一時的コンピュータ読取可能記憶媒体であって、前記命令は、実行されたことに応じてシステムに動作を実行させ、前記動作は、
複数のイベント対応を決定し、各イベント対応は、第１のソフトウェアプログラムの第１のソースコードに対してなされた複数の変更のうちそれぞれの変更と前記第１のソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示すことであり、前記複数のイベント対応は、複数のイベントタイプ推論動作のうち２つ以上のイベントタイプ推論動作を実行することにより決定され、前記複数のイベントタイプ推論動作は、
不良導入イベントが前記複数の変更のうち第１の変更に対応することを、前記第１の変更が前記第１のソースコードに含まれないで合格し、かつ前記第１の変更が前記第１のソースコードに含まれて不合格になった前記第１のソースコードの第１のソフトウェアテストを識別することに基づいて、決定することと、
不良訂正イベントが前記複数の変更のうち第２の変更に対応することを、前記第２の変更が前記第１のソースコードに含まれないで不合格になり、かつ前記第２の変更が前記第１のソースコードに含まれて合格した前記第１のソースコードの第２のソフトウェアテストを識別することに基づいて、決定することと、
欠陥導入イベントが前記複数の変更のうち第３の変更に対応することを、前記第３の変更が前記第１のソースコードに含まれないで前記第１のソースコードに対して実行された第１の静的解析から第１の欠陥が識別されないことに基づいて、及び前記第３の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第２の静的解析から前記第１の欠陥が識別されることに基づいて、決定することと、
欠陥訂正イベントが前記複数の変更のうち第４の変更に対応することを、前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第３の静的解析から第２の欠陥が識別されることに基づいて、及び前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第４の静的解析から前記第２の欠陥が識別されないことに基づいて、決定することと、
特定のプラットフォームの第１のバージョンから前記特定のプラットフォームの第２のバージョンへのプラットフォーム移行イベントが前記複数の変更のうち第５の変更に対応することを、前記第５の変更が中に含まれた前記第１のソースコードの第１のビルドが、前記第５の変更が中に含まれた前記第１のソースコードの第２のビルドに関して省略されるエラーを有することに基づいて、決定することであり、前記第１のビルドは前記特定のプラットフォームの前記第１のバージョンを使用して実行され、前記第２のビルドは前記特定のプラットフォームの前記第２のバージョンを使用して実行される、ことと、
を含む、ことと、
第２のソフトウェアプログラムの第２のソースコードにおける１つ以上のエラーを、前記第２のソースコードに関してテストスイートを実行することに基づいて識別することと、
前記第１のソースコードから決定された前記複数のイベント対応を使用して前記１つ以上のエラーに関して前記第２のソースコードに対して修復動作を実行することと、
を含む、１つ以上の非一時的コンピュータ読取可能記憶媒体。
（付記９）
前記動作は、前記複数の変更のうち特定の変更についてのコミットメッセージを生成し、前記コミットメッセージは前記特定の変更に関して決定されたそれぞれのイベント対応を示すことをさらに含む、付記８に記載の１つ以上のコンピュータ読取可能記憶媒体。

（付記１０）
前記不良導入イベントが前記第１の変更に対応すると決定することは、
前記第１の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第１のソフトウェアテストを含む、ことと、
前記第１の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、
前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別することと、
前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することと、
前記不良導入イベントが前記第１の変更に対応することを、前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別すること及び前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することに応じて決定することと、
を含む、付記８に記載の１つ以上のコンピュータ読取可能記憶媒体。
（付記１１）
前記不良訂正イベントが前記第２の変更に対応すると決定することは、
前記第２の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第２のソフトウェアテストを含む、ことと、
前記第２の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、
前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別することと、
前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別することと、
前記不良訂正イベントが前記第２の変更に対応することを、前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別すること及び前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別することに応じて決定することと、
を含む、付記８に記載の１つ以上のコンピュータ読取可能記憶媒体。
（付記１２）
前記欠陥導入イベントが前記第３の変更に対応すると決定することは、
前記第３の変更を含む前記第１のソースコードの部分に関して前記第２の静的解析を実行することであり、前記第２の静的解析は前記第１の欠陥を識別する、ことと、
前記第３の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第１の静的解析を実行することであり、前記第１の静的解析は前記第１の欠陥を識別しない、ことと、
前記欠陥導入イベントが前記第３の変更に対応することを、前記第２の静的解析が前記第１の欠陥を識別すること及び前記第１の静的解析が前記第１の欠陥を識別しないことに応じて決定することと、
を含む、付記８に記載の１つ以上のコンピュータ読取可能記憶媒体。
（付記１３）
前記欠陥訂正イベントが前記第４の変更に対応すると決定することは、
前記第４の変更を含む前記第１のソースコードの部分に関して前記第３の静的解析を実行することであり、前記第３の静的解析は前記第２の欠陥を識別しない、ことと、
前記第４の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第４の静的解析を実行することであり、前記第４の静的解析は前記第２の欠陥を識別する、ことと、

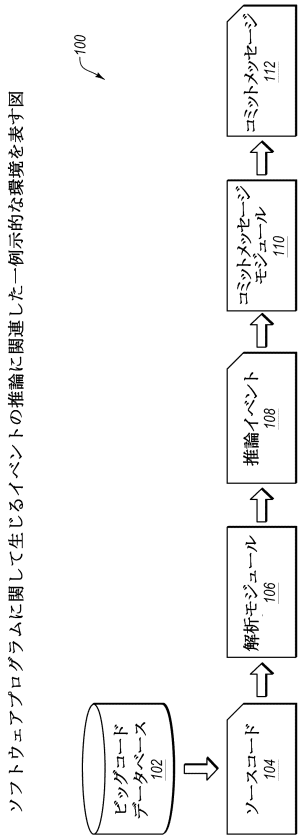
前記欠陥訂正イベントが前記第４の変更に対応することを、前記第４の静的解析が前記第２の欠陥を識別すること及び前記第３の静的解析が前記第２の欠陥を識別しないことに応じて決定することと、
を含む、付記８に記載の１つ以上のコンピュータ読取可能記憶媒体。
（付記１４）
前記プラットフォーム移行イベントが前記第５の変更に対応すると決定することは、
前記第５の変更を含む前記第１のソースコードの部分に関して前記第１のビルドを実行することであり、前記第１のビルドは前記エラーを識別する、ことと、
前記第５の変更を中に含ませた前記第１のソースコードの前記部分に関して前記第２のビルドを実行することであり、前記第２のビルドは前記エラーを識別しない、ことと、
前記プラットフォーム移行イベントが前記第５の変更に対応することを、前記第１のビルドが前記エラーを識別すること及び前記第２のビルドが前記エラーを識別しないことに応じて決定することと、
を含む、付記８に記載の１つ以上のコンピュータ読取可能記憶媒体。
（付記１５）
システムであって、
命令を記憶するように構成された１つ以上のコンピュータ読取可能記憶媒体と、
前記１つ以上のコンピュータ読取可能記憶媒体に通信上結合され、前記命令の実行に応じて当該システムに動作を実行させるように構成された１つ以上のプロセッサと、
を含み、前記動作は、
複数のイベント対応を決定し、各イベント対応は、第１のソフトウェアプログラムの第１のソースコードに対してなされた複数の変更のうちそれぞれの変更と前記第１のソフトウェアプログラムに関して生じるそれぞれのイベントタイプとの間の対応を示すことであり、前記複数のイベント対応は、複数のイベントタイプ推論動作のうち２つ以上のイベントタイプ推論動作を実行することにより決定され、前記複数のイベントタイプ推論動作は、
不良導入イベントが前記複数の変更のうち第１の変更に対応することを、前記第１の変更が前記第１のソースコードに含まれないで合格し、かつ前記第１の変更が前記第１のソースコードに含まれて不合格になった前記第１のソースコードの第１のソフトウェアテストを識別することに基づいて、決定することと、
不良訂正イベントが前記複数の変更のうち第２の変更に対応することを、前記第２の変更が前記第１のソースコードに含まれないで不合格になり、かつ前記第２の変更が前記第１のソースコードに含まれて合格した前記第１のソースコードの第２のソフトウェアテストを識別することに基づいて、決定することと、
欠陥導入イベントが前記複数の変更のうち第３の変更に対応することを、前記第３の変更が前記第１のソースコードに含まれないで前記第１のソースコードに対して実行された第１の静的解析から第１の欠陥が識別されないことに基づいて、及び前記第３の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第２の静的解析から前記第１の欠陥が識別されることに基づいて、決定することと、
欠陥訂正イベントが前記複数の変更のうち第４の変更に対応することを、前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第３の静的解析から第２の欠陥が識別されることに基づいて、及び前記第４の変更が前記第１のソースコードに含まれて前記第１のソースコードに対して実行された第４の静的解析から前記第２の欠陥が識別されないことに基づいて、決定することと、
特定のプラットフォームの第１のバージョンから前記特定のプラットフォームの第２のバージョンへのプラットフォーム移行イベントが前記複数の変更のうち第５の変更に対応することを、前記第５の変更が中に含まれた前記第１のソースコードの第１のビルドが、前記第５の変更が中に含まれた前記第１のソースコードの第２のビルドに関して省略されるエラーを有することに基づいて、決定することであり、前記第１のビルドは前記特定のプラットフォームの前記第１のバージョンを使用して実行され、前記第２のビルドは

前記特定のプラットフォームの前記第２のバージョンを使用して実行される、ことと、
を含む、ことと、
前記複数の変更のうち特定の変更についてのコミットメッセージを生成し、前記コミットメッセージは前記特定の変更に関して決定されたそれぞれのイベント対応を示すことと、
を含む、システム。
（付記１６）
前記不良導入イベントが前記第１の変更に対応すると決定することは、
前記第１の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第１のソフトウェアテストを含む、ことと、
前記第１の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、
前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別することと、
前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することと、
前記不良導入イベントが前記第１の変更に対応することを、前記第１の実行に関して前記第１のソフトウェアテストを不合格テストとして識別すること及び前記第２の実行に関して前記第１のソフトウェアテストを合格テストとして識別することに応じて決定することと、
を含む、付記１５に記載のシステム。
（付記１７）
前記不良訂正イベントが前記第２の変更に対応すると決定することは、
前記第２の変更を含む前記第１のソースコードの部分に関してテストスイートの第１の実行を実行することであり、前記テストスイートは前記第２のソフトウェアテストを含む、ことと、
前記第２の変更が中に含まれない前記第１のソースコードの前記部分に関して前記テストスイートの第２の実行を実行することと、
前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別することと、
前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別することと、
前記不良訂正イベントが前記第２の変更に対応することを、前記第１の実行に関して前記第２のソフトウェアテストを合格テストとして識別すること及び前記第２の実行に関して前記第２のソフトウェアテストを不合格テストとして識別することに応じて決定することと、
を含む、付記１５に記載のシステム。
（付記１８）
前記欠陥導入イベントが前記第３の変更に対応すると決定することは、
前記第３の変更を含む前記第１のソースコードの部分に関して前記第２の静的解析を実行することであり、前記第２の静的解析は前記第１の欠陥を識別する、ことと、
前記第３の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第１の静的解析を実行することであり、前記第１の静的解析は前記第１の欠陥を識別しない、ことと、
前記欠陥導入イベントが前記第３の変更に対応することを、前記第２の静的解析が前記第１の欠陥を識別すること及び前記第１の静的解析が前記第１の欠陥を識別しないことに応じて決定することと、
を含む、付記１５に記載のシステム。
（付記１９）

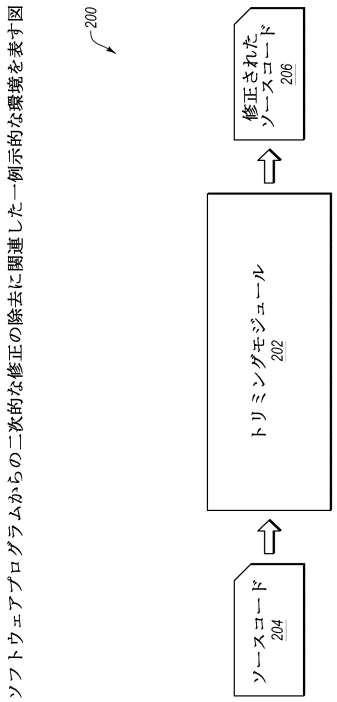
前記欠陥訂正イベントが前記第４の変更に対応すると決定することは、
前記第４の変更を含む前記第１のソースコードの部分に関して前記第３の静的解析を実行することであり、前記第３の静的解析は前記第２の欠陥を識別しない、ことと、
前記第４の変更が中に含まれない前記第１のソースコードの前記部分に関して前記第４の静的解析を実行することであり、前記第４の静的解析は前記第２の欠陥を識別する、ことと、
前記欠陥訂正イベントが前記第４の変更に対応することを、前記第４の静的解析が前記第２の欠陥を識別すること及び前記第３の静的解析が前記第２の欠陥を識別しないことに応じて決定することと、
を含む、付記１５に記載のシステム。

（付記２０）
前記プラットフォーム移行イベントが前記第５の変更に対応すると決定することは、
前記第５の変更を含む前記第１のソースコードの部分に関して前記第１のビルドを実行することであり、前記第１のビルドは前記エラーを識別する、ことと、
前記第５の変更を中に含ませた前記第１のソースコードの前記部分に関して前記第２のビルドを実行することであり、前記第２のビルドは前記エラーを識別しない、ことと、
前記プラットフォーム移行イベントが前記第５の変更に対応することを、前記第１のビルドが前記エラーを識別すること及び前記第２のビルドが前記エラーを識別しないことに応じて決定することと、
を含む、付記１５に記載のシステム。

【図面】
【図１】

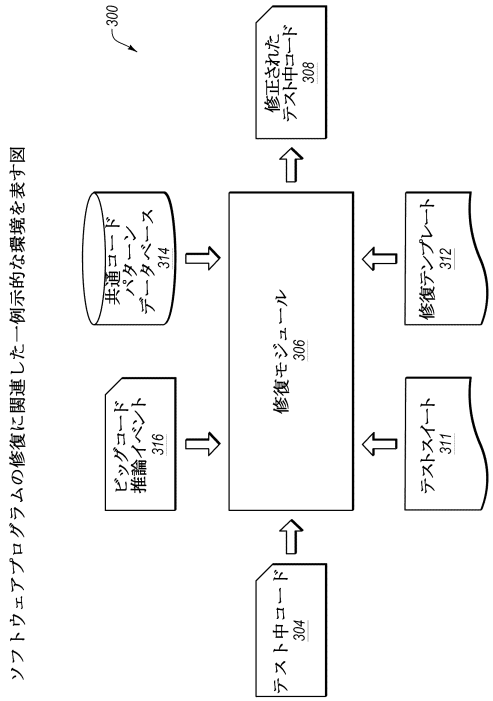


【図２】



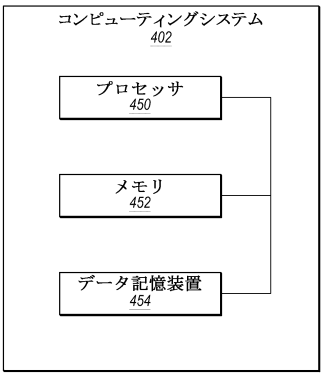
【図３】

ソフトウェアプログラムの修復に関連した一例示的な環境を表す図



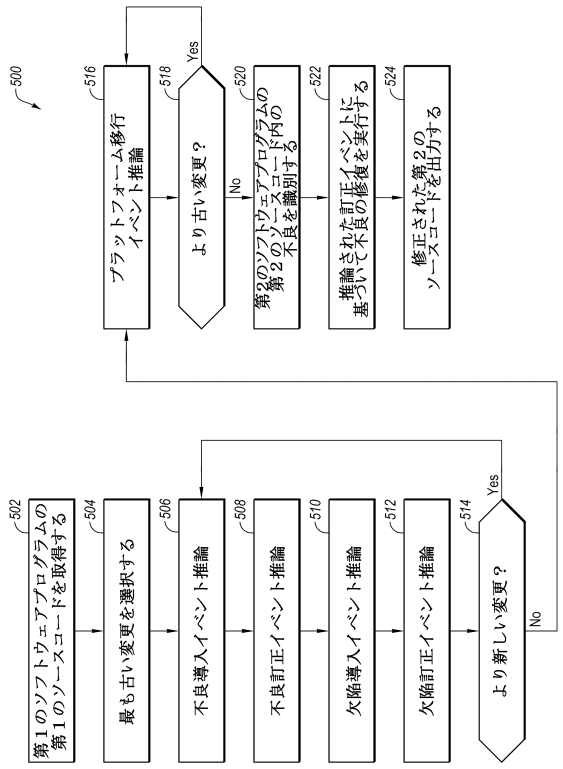
【図４】

一例示的なコンピューティングシステムのブロック図



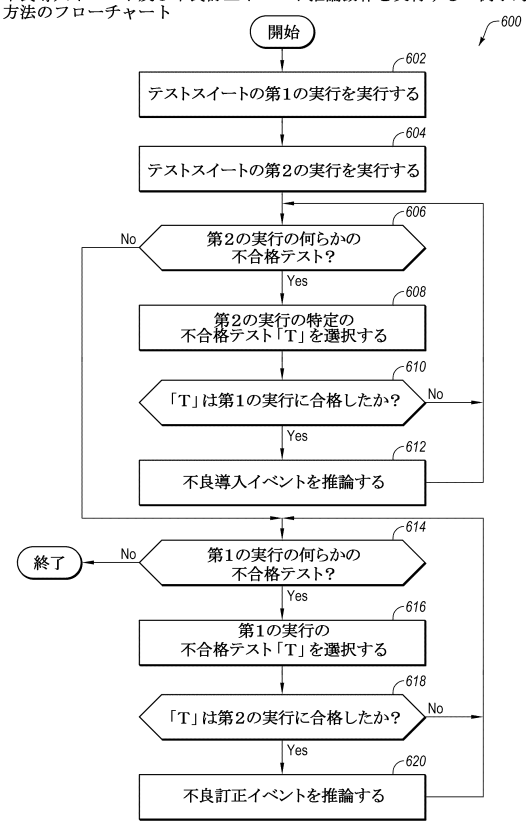
【図５】

ソフトウェアプログラムに関して生じるイベントを推論する一例示的な方法のフローチャート



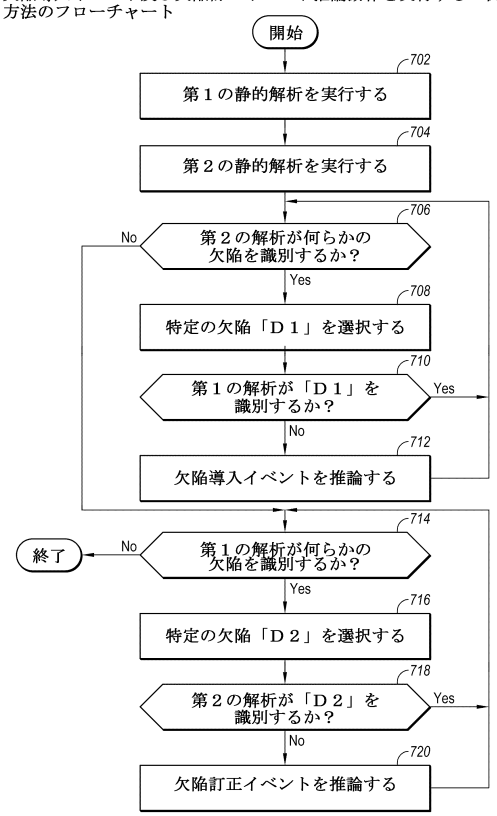
【図６】

不良導入イベント及び不良訂正イベント推論動作を実行する一例示的な方法のフローチャート



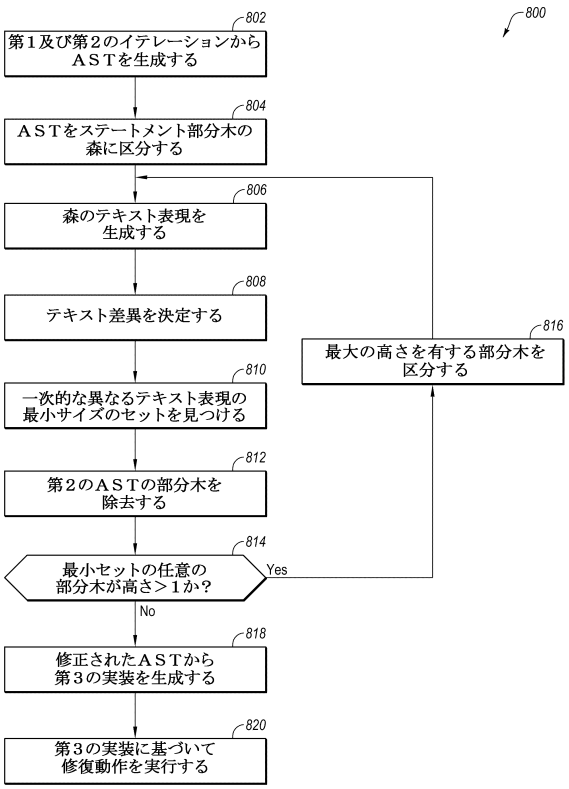
【図７】

欠陥導入イベント及び欠陥訂正イベント推論動作を実行する一例示的な方法のフローチャート



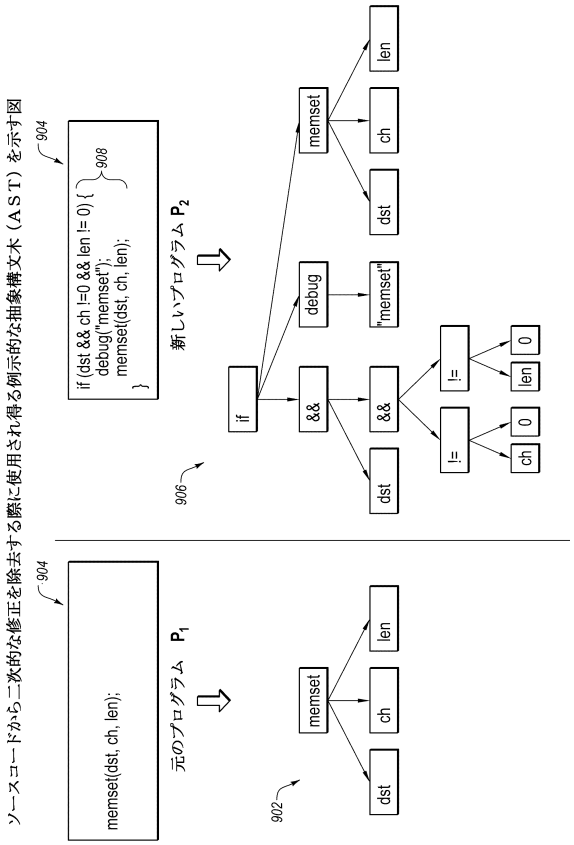
【図８】

ソフトウェアプログラムのソースコードから二次的な修正を除去する一例示的な方法のフローチャート



【図９Ａ】

ソースコードから二次的な修正を除去する際に使用され得る例示的な抽象構文木（ＡＳＴ）を示す図



【図９Ｂ】

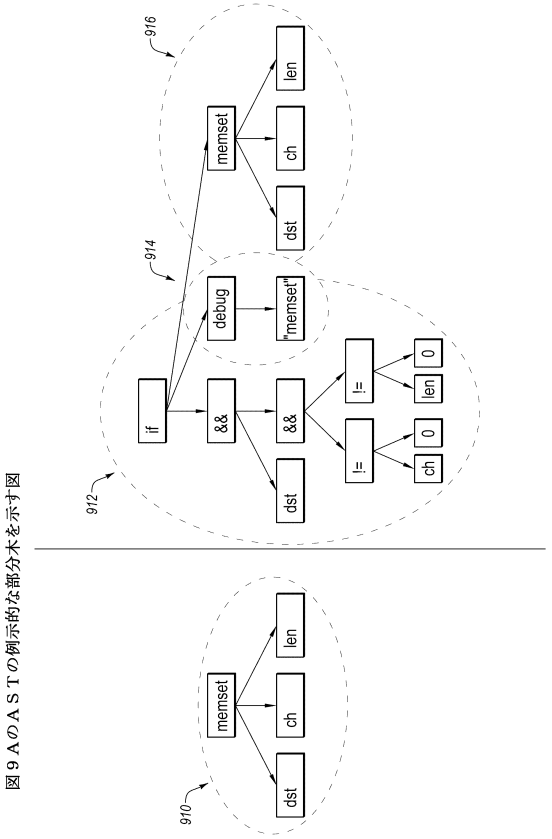
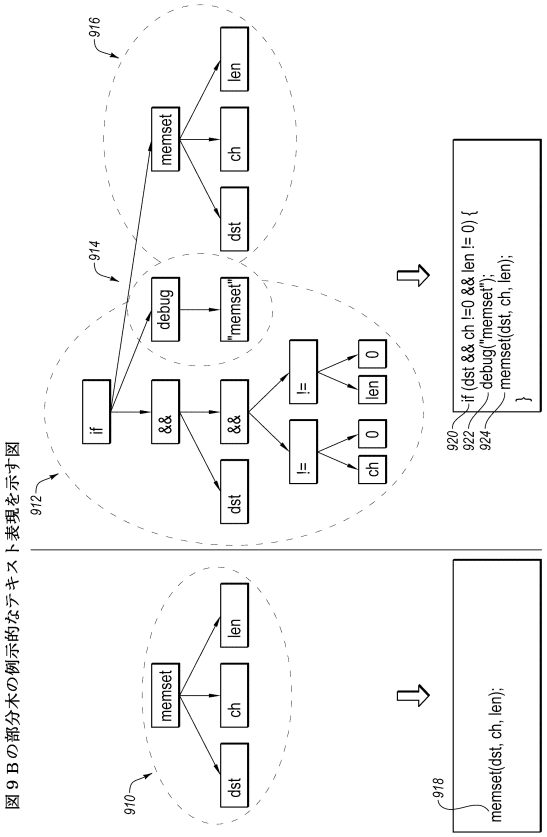
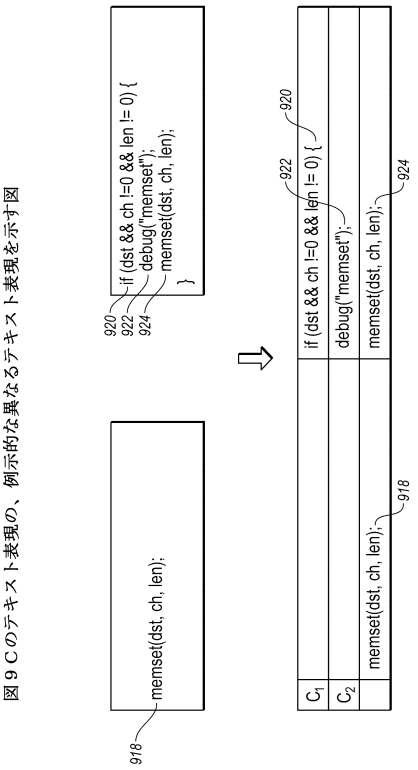


図９ＡのＡＳＴの例示的な部分木を示す図

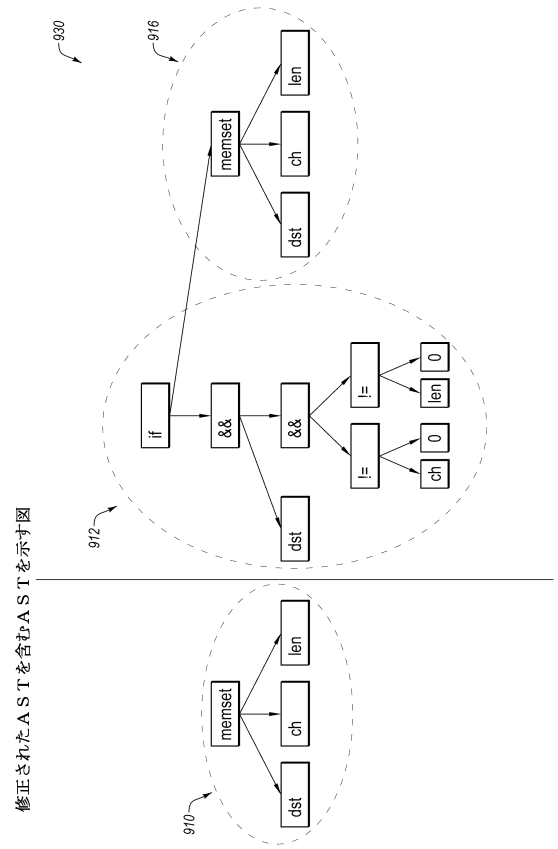
【図９Ｃ】



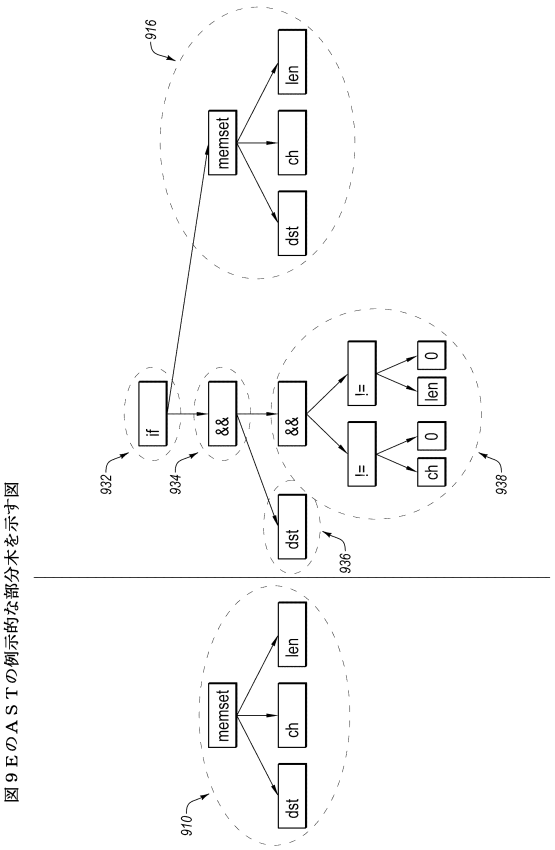
【図９Ｄ】



【図９Ｅ】



【図９Ｆ】



フロントページの続き

４０８５，サニーヴェイル，イースト アークス アヴェニュ １２４０番 フジツウ ラボラトリーズ アメリカ内

審査官　山本 俊介

(56)参考文献　特開昭６３－５６７２８（ＪＰ，Ａ）
特開２０１８－１８１９７（ＪＰ，Ａ）
特開２０１７－１５１５９４（ＪＰ，Ａ）
特表２００７－５３５７２３（ＪＰ，Ａ）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１１／３６
Ｇ０６Ｆ　８／７０