(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7536722号
(P7536722)

(45)発行日　令和6年8月20日(2024.8.20)

(24)登録日　令和6年8月9日(2024.8.9)

(51)国際特許分類
Ｇ０６Ｆ　8/77 (2018.01)

ＦＩ
Ｇ０６Ｆ　8/77

請求項の数　12　（全20頁）

(21)出願番号　特願2021-137219(P2021-137219)
(22)出願日　令和3年8月25日(2021.8.25)
(65)公開番号　特開2023-31614(P2023-31614A)
(43)公開日　令和5年3月9日(2023.3.9)
審査請求日　令和5年7月27日(2023.7.27)

(73)特許権者　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(74)代理人　110001737
弁理士法人スズエ国際特許事務所
(72)発明者　森 奈実子
東京都港区芝浦一丁目１番１号 株式会社東芝内
(72)発明者　田村 文隆
東京都港区芝浦一丁目１番１号 株式会社東芝内
(72)発明者　村田 由香里
東京都港区芝浦一丁目１番１号 株式会社東芝内
審査官　児玉 崇晶

最終頁に続く

(54)【発明の名称】　変更度計測装置、方法及びプログラム

(57)【特許請求の範囲】
【請求項１】
第１関数を含む第１ソースコード及び第２関数を含む第２ソースコードを入力する入力手段と、
前記入力された第１ソースコードに含まれる第１関数を構文解析し、当該構文解析結果に基づいて当該第１関数の関数名を予測する第１予測手段と、
前記入力された第２ソースコードに含まれる第２関数を構文解析し、当該構文解析結果に基づいて当該第２関数の関数名を予測する第２予測手段と、
前記第１関数の関数名の予測結果及び前記第２関数の関数名の予測結果を比較することによって、前記第１及び第２関数間の第１変更度を計測する第１計測手段と、
前記計測された第１変更度を出力する出力手段と
を具備する変更度計測装置。
【請求項２】
前記第１予測手段は、関数を入力することによって当該関数の関数名の予測結果を出力するように機械学習により生成された学習済モデルを用いて、前記第１関数の関数名を予測し、
前記第２予測手段は、前記学習済モデルを用いて、前記第２関数の関数名を予測する
請求項１記載の変更度計測装置。
【請求項３】
前記学習済モデルは、前記関数の関数名の予測結果として当該関数名の複数の候補と当

該複数の候補の各々が当該関数の関数名である確率とを出力するように生成されており、
前記第１計測手段は、前記第１関数の関数名の予測結果として前記学習済モデルから出力された前記第１関数の関数名の複数の第１候補の各々が当該第１関数の関数名である確率の分布と、前記第２関数の関数名の予測結果として前記学習済モデルから出力された前記第２関数の関数名の複数の第２候補の各々が当該第２関数の関数名である確率の分布との差異に基づいて前記第１変更度を計測する
請求項２記載の変更度計測装置。
【請求項４】
前記複数の第１候補の各々が前記第１関数の関数名である確率の分布と、前記複数の第２候補の各々が前記第２関数の関数名である確率の分布との差異は、ＫＬダイバージェンスまたはＪＳダイバージェンスを用いて計算される請求項３記載の変更度計測装置。
【請求項５】
前記第１及び第２関数の関数名は、前記学習済モデル上においてベクトル表現として表され、
前記第１計測手段は、前記第１関数の関数名を表すベクトル表現と、前記第２関数の関数名を表すベクトル表現との差異に基づいて前記第１変更度を計測する
請求項２記載の変更度計測装置。
【請求項６】
前記第１関数の関数名を表すベクトル表現と、前記第２関数の関数名を表すベクトル表現との差異は、コサイン類似度またはユークリッド距離を用いて計算される請求項５記載の変更度計測装置。
【請求項７】
前記出力手段は、前記第１関数の関数名の予測結果及び前記第２関数の関数名の予測結果を更に出力する請求項１～６のいずれか一項に記載の変更度計測装置。
【請求項８】
前記入力された第１ソースコードに含まれる第１関数を表すテキストと前記入力された第２ソースコードに含まれる第２関数を表すテキストとを比較することによって前記第１及び第２関数間の第２変更度を計測する第２計測手段を更に具備し、
前記出力手段は、前記計測された第２変更度を更に出力する
請求項１～７のいずれか一項に記載の変更度計測装置。
【請求項９】
前記出力手段は、前記第１及び第２変更度をグラフ上に表示する表示処理手段含む請求項８記載の変更度計測装置。
【請求項１０】
第１及び第２ソースコードを入力する入力手段と、
前記入力された第１ソースコードを構文解析し、当該構文解析結果に基づいて当該第１ソースコードの要約を予測する第１予測手段と、
前記入力された第２ソースコードを構文解析し、当該構文解析結果に基づいて当該第２ソースコードの要約を予測する第２予測手段と、
前記第１ソースコードの要約の予測結果及び前記第２ソースコードの要約の予測結果を比較することによって、前記第１及び第２ソースコード間の第１変更度を計測する第１計測手段と、
前記計測された第１変更度を出力する出力手段と
を具備する変更度計測装置。
【請求項１１】
変更度計測装置が実行する方法であって、
第１関数を含む第１ソースコード及び第２関数を含む第２ソースコードを入力するステップと、
前記入力された第１ソースコードに含まれる第１関数を構文解析し、当該構文解析結果に基づいて当該第１関数の関数名を予測するステップと、

前記入力された第２ソースコードに含まれる第２関数を構文解析し、当該構文解析結果に基づいて当該第２関数の関数名を予測するステップと、
前記第１関数の関数名の予測結果及び前記第２関数の関数名の予測結果を比較することによって、前記第１及び第２関数間の第１変更度を計測するステップと、
前記計測された第１変更度を出力するステップと
を具備する方法。
【請求項１２】
変更度計測装置のコンピュータによって実行されるプログラムであって、
前記コンピュータに、
第１関数を含む第１ソースコード及び第２関数を含む第２ソースコードを入力するステップと、
前記入力された第１ソースコードに含まれる第１関数を構文解析し、当該構文解析結果に基づいて当該第１関数の関数名を予測するステップと、
前記入力された第２ソースコードに含まれる第２関数を構文解析し、当該構文解析結果に基づいて当該第２関数の関数名を予測するステップと、
前記第１関数の関数名の予測結果及び前記第２関数の関数名の予測結果を比較することによって、前記第１及び第２関数間の第１変更度を計測するステップと、
前記計測された第１変更度を出力するステップと
を実行させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施形態は、変更度計測装置、方法及びプログラムに関する。
【背景技術】
【０００２】
一般に、ソフトウェアプロダクトライン（ＳＰＬ：Software Product Line）開発のようなソフトウェア製品の開発においては、既存のソフトウェア製品を実現するためのソースコードを利用した派生開発が行われる。
【０００３】
この派生開発を効率的に行うためには、同種同系列のソフトウェア製品を実現するための２つのソースコード間で変更された箇所または変更されていない箇所を分析（把握）する必要があるが、当該分析には、当該ソースコードに含まれる関数が変更された度合い（以下、変更度と表記）を計測すること有用である。
【先行技術文献】
【特許文献】
【０００４】
【文献】国際公開第２０２０／０４９６２２号
【発明の概要】
【発明が解決しようとする課題】
【０００５】
そこで、本発明が解決しようとする課題は、２つのソースコードに含まれる関数間の変更度を計測することが可能な変更度計測装置、方法及びプログラムを提供することにある。
【課題を解決するための手段】
【０００６】
実施形態に係る変更度計測装置は、入力手段と、第１予測手段と、第２予測手段と、計測手段と、出力手段とを具備する。前記入力手段は、第１関数を含む第１ソースコード及び第２関数を含む第２ソースコードを入力する。前記第１予測手段は、前記入力された第１ソースコードに含まれる第１関数を構文解析し、当該構文解析結果に基づいて当該第１関数の関数名を予測する。前記第２予測手段は、前記入力された第２ソースコードに含まれる第２関数を構文解析し、当該構文解析結果に基づいて当該第２関数の関数名を予測す

る。前記計測手段は、前記第１関数の関数名の予測結果及び前記第２関数の関数名の予測結果を比較することによって、前記第１及び第２関数間の第１変更度を計測する。前記出力手段は、前記計測された第１変更度を出力する。

【図面の簡単な説明】

【０００７】

【図１】第１実施形態に係る関数間変更度計測装置の機能構成の一例を示すブロック図。

【図２】関数間変更度計測装置のシステム構成の一例を示す図。

【図３】関数間変更度計測装置の動作の概要について説明するための図。

【図４】関数間変更度計測装置の処理手順の一例を示すフローチャート。

【図５】ＡＳＴの一例を示す図。

【図６】ソースコードに含まれる関数の具体例を示す図。

【図７】変更度が表示された表示画面の一例を示す図。

【図８】変更度が表示された表示画面の一例を示す図。

【図９】ソースコード管理ツールを利用する構成について説明するための図。

【図１０】第２実施形態に係る関数間変更度計測装置の機能構成の一例を示すブロック図。

【図１１】関数間変更度計測装置の処理手順の一例を示すフローチャート。

【図１２】第１及び第２変更度が表示された表示画面の一例を示す図。

【図１３】第１及び第２変更度が表示された表示画面の別の例を示す図。

【発明を実施するための形態】

【０００８】

以下、図面を参照して、各実施形態について説明する。

（第１実施形態）

まず、第１実施形態について説明する。本実施形態に係る変更度計測装置は、ソフトウェア製品の開発における派生開発等を効率的に行うために、２つのソースコードに含まれる関数間の変更度を計測するために用いられる。以下の説明においては、本実施形態に係る変更度計測装置を、便宜的に、関数間変更度計測装置と称する。

【０００９】

ここで、関数間の変更度の計測するために当該関数間の差分を抽出することが考えられるが、当該関数間の差分を抽出するためのツールとしては、例えばＤｉｆｆと称されるテキスト比較ツールがある。

【００１０】

しかしながら、このようなテキスト比較ツールにおいては所定のプログラミング言語で記述されている関数を含む２つのソースコード（テキスト）を比較することによって当該２つのソースコードに含まれる関数間の差分が抽出されるため、例えば機能的には大きな変更がされていない場合であっても、リファクタリング等によりソースコード（関数）の記述が大幅に変更されている場合には、大きな差分が抽出され、高い変更度が計測されることになる。同様に、例えば機能的に大きな変更がされた場合であっても、ソースコード（関数）の記述の変更が小規模である場合には、抽出される差分が小さくなるため、低い変更度が計測されることになる。

【００１１】

このため、本実施形態においては、２つの関数が機能的な観点から異なっている程度を定量化するために、機械学習を用いた自然言語処理の応用の１つとして当該関数（本体）から当該関数の関数名を予測する技術を利用する。

【００１２】

具体的には、例えば２つの関数の関数名（予測名）が「ｓｏｒｔＢｙＮａｍｅ」及び「ｓｏｒｔＢｙＩｎｄｅｘ」（ソート機能を表す関数名）であれば、当該２つの関数の機能は類似している（つまり、変更度は小さい）と考えられる。一方、２つの関数の関数名（予測名）が「ｓｏｒｔＢｙＮａｍｅ」及び「ｓｈｏｗＤｉａｌｏｇ」であれば、当該２つの関数の機能は類似していない（つまり、変更度は大きい）と考えられる。

【００１３】

本実施形態に係る関数間変更度計測装置１０は、上記した関数名の予測結果を利用することによって、機能的な観点に基づく関数間の変更度（機能としての変更の大小を考慮した関数間の変更度）を計測する構成を有する。

【００１４】

図１は、本実施形態に係る関数間変更度計測装置の機能構成の一例を示すブロック図である。

【００１５】

図１に示すように、関数間変更度計測装置１０は、入力部１１、関数名予測部１２、変更度計測部１３及び出力部１４を含む。

【００１６】

入力部１１は、例えば関数間変更度計測装置１０を使用するユーザ（ソフトウェア製品の開発を行うユーザ）によって指定された２つのソースコード（以下、第１及び第２ソースコードと表記）を入力する。

【００１７】

本実施形態において入力部１１によって入力される第１及び第２ソースコードは、上記したように所定のプログラム言語で記述された関数を含む。また、第１及び第２ソースコードは、例えばソフトウェアプロダクトライン開発における同種同系列のソフトウェア製品（例えば、バージョンが異なるソフトウェア製品）を実現するためのソースコードであるものとする。

【００１８】

なお、入力部１１によって入力される第１及び第２ソースコードは、関数間変更度計測装置１０に含まれる格納部（図示せず）に予め格納されていてもよいし、当該関数間変更度計測装置１０の外部のサーバ装置等から受信されたものであってもよい。

【００１９】

関数名予測部１２は、入力部１１によって入力された第１ソースコードに含まれる関数（以下、第１関数と表記）を構文解析し、当該構文解析結果に基づいて当該第１関数の関数名を予測する。また、関数名予測部１２は、入力部１１によって入力された第２ソースコードに含まれる関数（以下、第２関数と表記）を構文解析し、当該構文解析結果に基づいて当該第２関数の関数名を予測する。

【００２０】

変更度計測部１３は、関数名予測部１２による第１関数の関数名の予測結果及び第２関数の関数名の予測結果を比較することによって、第１及び第２関数の変更度を計測（計算）する。

【００２１】

出力部１４は、変更度計測部１３によって計測された変更度を出力する。本実施形態において、出力部１４は、例えば変更度を表示する表示処理部である場合を想定しているが、当該変更度を外部のサーバ装置等に送信する送信部等であっても構わない。

【００２２】

図２は、図１に示す関数間変更度計測装置１０のシステム構成の一例を示す。ここでは、関数間変更度計測装置１０が例えばパーソナルコンピュータ（ＰＣ）等であるものとして説明するが、当該関数間変更度計測装置１０は他の電子機器（情報処理装置）であってもよい。

【００２３】

図２に示すように、関数間変更度計測装置１０は、ＣＰＵ１０１、不揮発性メモリ１０２、主メモリ１０３、ＢＩＯＳ－ＲＯＭ１０４、システムコントローラ１０５、入力デバイス１０６、表示デバイス１０７及びエンベデッドコントローラ（ＥＣ）１０８等を備える。

【００２４】

ＣＰＵ１０１は、関数間変更度計測装置１０内の各コンポーネントの動作を制御するプロセッサである。ＣＰＵ１０１は、ストレージデバイスである不揮発性メモリ１０２から

主メモリ１０３にロードされる様々なプログラムを実行する。このプログラムには、オペレーティングシステム（ＯＳ）及び関数間の変更度を計測するためのプログラム（以下、関数間変更度計測プログラムと表記）等が含まれる。

【００２５】

また、ＣＰＵ１０１は、ＢＩＯＳ－ＲＯＭ１０４に格納された基本入出力システム（ＢＩＯＳ）も実行する。ＢＩＯＳは、ハードウェア制御のためのプログラムである。

【００２６】

システムコントローラ１０５は、ＣＰＵ１０１のローカルバスと各種コンポーネントとの間を接続するデバイスである。

【００２７】

入力デバイス１０６は、例えばキーボード及びマウス等を含む。表示デバイス１０７は、例えば液晶表示装置のようなディスプレイ等を含む。ＥＣ１０８は、電力管理のためのワンチップマイクロコンピュータである。

【００２８】

図２においては、ＣＰＵ１０１、不揮発性メモリ１０２、主メモリ１０３、ＢＩＯＳ－ＲＯＭ１０４、システムコントローラ１０５、入力デバイス１０６、表示デバイス１０７及びＥＣ１０８のみが示されているが、関数間変更度計測装置１０は、例えばＨＤＤ（Hard Disk Drive）及びＳＳＤ（Solid State Drive）のような他の記憶装置を備えていてもよいし、外部装置との通信を実行するように構成された通信デバイス等を備えていてもよい。

【００２９】

なお、本実施形態において、図１に示す各部１１～１４の一部または全ては、例えば図２に示すＣＰＵ１０１（つまり、関数間変更度計測装置１０のコンピュータ）に上記した関数間変更度計測プログラムを実行させること、すなわち、ソフトウェアによって実現されるものとする。関数間変更度計測プログラムは、コンピュータ読み取り可能な記憶媒体に予め格納して頒布可能であるし、ネットワークを介して関数間変更度計測装置１０にダウンロードされても構わない。

【００３０】

ここでは、各部１１～１４の一部または全てがソフトウェアによって実現されるものとして説明したが、当該各部１１～１４の一部または全ては、例えばハードウェアによって実現されてもよいし、ソフトウェア及びハードウェアの組み合わせによって実現されてもよい。

【００３１】

以下、本実施形態に係る関数間変更度計測装置１０の動作について説明する。まず、図３を参照して、本実施形態に係る関数間変更度計測装置１０（変更度計測部１３）の動作の概要について説明する。

【００３２】

本実施形態において、関数間変更度計測装置１０（関数名予測部１２）は、関数を入力することによって当該関数の関数名の予測結果を出力するように機械学習により生成された学習済モデル１２ａを予め保持しているものとする。この学習済モデルから出力される関数名の予測結果には、例えば当該関数名の複数の候補（以下、関数名候補と表記）と、当該複数の関数名候補の各々が当該学習済モデルに入力された関数の関数名（正当な関数名）である確率（以下、関数名の確率と表記）とが含まれるものとする。

【００３３】

ここで、上記した学習済モデル１２ａに第１関数が入力された場合を想定する。この場合、学習済モデル１２ａは、例えば第１関数の３つの関数名候補及び当該３つの関数名候補の各々の確率を、第１関数の関数名の予測結果として出力する。

【００３４】

一方、学習済モデル１２ａに第２関数が入力された場合を想定する。この場合、学習済モデル１２ａは、例えば第２関数の３つの関数名候補及び当該３つの関数名候補の各々の

確率を、第２関数の関数名の予測結果として出力する。

【００３５】

本実施形態においては、上記した第１関数の関数名の予測結果及び第２関数の関数名の予測結果に基づいて第１及び第２関数間の変更度が計測される。この場合、第１及び第２関数間の変更度は、例えば第１関数の３つの関数名候補の確率分布と第２関数の３つの関数名候補の確率分布との差異に基づいて計測（計算）される。

【００３６】

次に、図４のフローチャートを参照して、関数間変更度計測装置１０の処理手順の一例について説明する。

【００３７】

まず、入力部１１は、例えばユーザによって指定された２つのソースコード（第１及び第２ソースコード）を入力する（ステップＳ１）。なお、第１及び第２ソースコードは、上記したようにバージョンが異なるソフトウェア製品を実現するためのソースコード（以下、バージョン違いのソースコードと表記）であるものとする。

【００３８】

次に、ステップＳ１において入力された第１及び第２ソースコードの各々から、変更度を計測する関数（第１及び第２関数）が切り出される（ステップＳ２）。

【００３９】

ここで、第１及び第２ソースコードにはそれぞれ複数の関数が含まれている（記述されている）ことが一般的であるが、当該第１及び第２ソースコードは上記したようにバージョン違いのソースコードであり、変更度を計測すべき第１ソースコードに含まれる第１関数と第２ソースコードに含まれる第２関数との対応関係を示す情報（以下、関数対応情報と表記）は、予め管理されているものとする。この関数対応情報は例えばソフトウェア製品の開発を行うユーザによって利用されるソースコード管理ツール（プラットフォーム）等において管理されているものとし、関数名予測部１２は、このような関数対応情報に基づいて、第１ソースコードから第１関数を切り出し、第２ソースコードから第２関数を切り出す。

【００４０】

ステップＳ２の処理が実行されると、関数名予測部１２は、上記した学習済モデルを用いて、当該ステップＳ２において切り出された第１関数の関数名及び第２関数の関数名を予測する。

【００４１】

ここで、本実施形態においては、関数名の予測に、例えばｃｏｄｅ２ｓｅｑ（Generating Sequences from Structured Representations of Code）に開示されている学習済モデル（以下、ｃｏｄｅ２ｓｅｑの学習済モデルと表記）を用いることができる。

【００４２】

以下、ｃｏｄｅ２ｓｅｑの学習済モデルの概要について簡単に説明する。ｃｏｄｅ２ｓｅｑの学習済モデルは、ニューラルネットワーク等の機械学習アルゴリズムを適用して生成されており、ソースコードの断片である関数の関数名を予測するために、当該関数を構文解析して当該関数を表すＡＳＴ（抽象構文木）を構築し、当該ＡＳＴの左端ノードから右端ノードへのパスを当該ニューラルネットワークへの入力に利用する。

【００４３】

ここで、図５は、ＡＳＴの一例を概略的に示している。ＡＳＴは、ソースコード（関数）において意味を成す最小単位（トークン）をノードとする木構造を有する。図５に示す例では、ｃｏｄｅ２ｓｅｑの学習済モデルに入力されるパス情報の例として、パス情報２０１～２０３が示されている。なお、パス情報２０１は、ＡＳＴの左端ノードであるノード２０１ａから右端ノードであるノード２０１ｂへのパスである。また、パス情報２０２は、ＡＳＴの左端ノードであるノード２０２ａから右端ノードであるノード２０２ｂへのパスである。更に、パス情報２０３は、ＡＳＴの左端ノードであるノード２０３ａから右端ノードであるノード２０３ｂへのパスである。

【００４４】
ｃｏｄｅ２ｓｅｑの学習済モデルは、上記したパス情報２０１～２０３のような複数のパス情報（例えば、２００パス程度）を入力することにより、例えば一連の単語から構成される複数の関数名候補及び当該関数名候補の各々の確率を出力するように構築（生成）されている。
【００４５】
関数名予測部１２は、第１及び第２関数（を表すＡＳＴのパス情報）をｃｏｄｅ２ｓｅｑの学習済モデルに入力することによって、第１関数の複数の関数名候補及び当該関数名候補の各々の確率と、第２関数の複数の関数名候補及び当該関数名候補の各々の確率とを取得する（ステップＳ３）。
【００４６】
なお、関数名の予測にｃｏｄｅ２ｓｅｑの学習済モデルが用いられる場合には、ビームサーチによって複数の関数名候補及び当該関数名候補の各々予測確率分布を得ることができる。この関数名候補として得られる関数名の数は、ビームサーチのビーム幅によって定められる。
【００４７】
本実施形態においてはｃｏｄｅ２ｓｅｑの学習済モデルが用いられるものとして説明したが、入力された関数の関数名を予測することができるものであれば、他のアルゴリズムに基づいて生成された学習済モデルが用いられても構わない。
【００４８】
また、例えば学習済モデルから出力された複数の関数名候補の中に予め定められている値（例えば、数％）よりも確率が低い関数名候補が含まれている場合には、当該関数名候補を排除し、残りの関数名候補のみを取得するようにしてもよい。この場合、取得された関数名候補の確率を正規化するような処理を更に実行してもよい。すなわち、本実施形態においては、上記した第１関数の関数名及び第２関数の関数名の予測結果に対して、適宜、形式化するような処理を実行してもよい。
【００４９】
次に、変更度計測部１３は、ステップＳ３において取得された第１関数の複数の関数名候補の確率の分布（以下、第１確率分布と表記）と、第２関数の複数の関数名候補の確率の分布（以下、第２確率分布と表記）とに基づいて、第１及び第２関数間の変更度を計測（判別）する（ステップＳ４）。
【００５０】
ステップＳ４においては、第１及び第２確率分布の差異に基づいて変更度が計測される。この第１及び第２確率分布の差異は、例えばＫＬ（Kullback-Leibler）ダイバージェンスまたはＪＳ（Jensen-Shannon）ダイバージェンスのような計算手法（メトリクス）を用いて計算されるものとする。
【００５１】
出力部１４は、ステップＳ４において計測された変更度を出力する（ステップＳ５）。この場合、出力部１４は、変更度を表示デバイス１０７に出力し、当該変更度を当該表示デバイス１０７に表示してもよい。
【００５２】
以下、本実施形態に係る関数間変更度計測装置１０の動作の具体例について説明する。ここでは、図６に示すように、同種同系列のソフトウェア製品を実現するためのバージョン違いのソースコードとして、バージョン１のソースコード３０１、バージョン２．１のソースコード３０２及びバージョン２．２のソースコード３０３が存在するものとする。なお、ソースコード３０１～３０３には複数の関数が含まれているが、図６においては、当該ソースコード３０１～３０３に含まれている複数の関数のうち、上記した変更度を計測すべき関数（つまり、対応する関数）のみが示されている。
【００５３】
ここで、ソースコード３０１に含まれる関数及びソースコード３０２に含まれる関数間

の変更度を計測する場合を想定する。図７は、ソースコード３０１及び３０２が関数間変更度計測装置１０に入力されることによって計測された当該ソースコード３０１に含まれる関数及び当該ソースコード３０２に含まれる関数間の変更度が表示された表示デバイス１０７の表示画面１０７ａの一例を示す。

【００５４】

図７に示すように、表示画面１０７ａには、ソースコード３０１に含まれる関数（バージョン１の関数）の関数名の予測結果、ソースコード３０２に含まれる関数（バージョン２．１の関数）の関数名の予測結果、ＫＬダイバージェンス値及び当該関数間の変更度が表示されている。

【００５５】

ソースコード３０１に含まれる関数の関数名の予測結果は、当該関数の複数の関数名候補及び当該複数の関数名候補の各々の確率を含む。図７に示す例では、ソースコード３０１に含まれる関数の関数名の予測結果として、関数名候補「ｓｏｒｔ」及び当該関数名候補の確率「７２．６％」と、関数名候補「ｒｅｖｅｒｓｅ」及び当該関数名候補の確率「１５．６％」と、関数名候補「ｂｕｂｂｌｅｓｏｒｔ」及び当該関数名候補の確率「１１．８％」とが表示されている。

【００５６】

ソースコード３０２に含まれる関数の関数名の予測結果は、当該関数の複数の関数名候補及び当該複数の関数名候補の各々の確率を含む。図７に示す例では、ソースコード３０２に含まれる関数の関数名の予測結果として、関数名候補「ｐｒｉｎｔＡｒｒａｙ」及び当該関数名候補の確率「７２．６％」と、関数名候補「ｄｕｍｐＡｒｒａｙ」及び当該関数名候補の確率「３０．５％」と、関数名候補「ｓｏｒｔ」及び当該関数名候補の確率「１３．５％」とが表示されている。

【００５７】

ＫＬダイバージェンス値は、上記したＫＬダイバージェンスを用いて計算された、ソースコード３０１に含まれる関数の関数名の予測結果に基づく確率分布（ソースコード３０１に含まれる関数の複数の関数名候補の各々の確率の分布）及びソースコード３０２に含まれる関数の関数名の予測結果に基づく確率分布（ソースコード３０２に含まれる関数の複数の関数名候補の各々の確率の分布）の差異を表す数値である。図７に示す例では、ＫＬダイバージェンス値として「４．８９３」が表示されている。

【００５８】

ここで、本実施形態における変更度は、上記したＫＬダイバージェンス値に基づいて計測されるものとする。具体的には、変更度は例えば「無」、「小」及び「大」を含み、ＫＬダイバージェンス値＝０であれば、機能としての関数間の差異がないと判別され、変更度「無」が計測されるものとする。また、０＜ＫＬダイバージェンス値＜１．５であれば、機能としての関数間の差異が小さいと判別され、変更度「小」が計測されるものとする。更に、ＫＬダイバージェンス値≧１．５であれば、機能としての関数間の差異が大きいと判別され、変更度「大」が計測されるものとする。

【００５９】

この場合、図７に示す例ではＫＬダイバージェンス値は「４．８９３」であるため、表示画面１０７ａには、当該ＫＬダイバージェンス値に基づいて計測された変更度「大」が表示されている。

【００６０】

上記した図６に示すソースコード３０１に含まれる関数とソースコード３０２に含まれる関数とを比較すると、テキストの大部分が共通しているため、Ｄｉｆｆのようなテキスト比較ツールを用いた場合には変更度は小さいと判別される可能性が高い。しかしながら、本実施形態においては、ＫＬダイバージェンス値に基づいて機能としての関数間の差異が大きいと判別されることによって、変更度「大」を計測することができる。

【００６１】

次に、ソースコード３０１に含まれる関数及びソースコード３０３に含まれる関数間の

変更度を計測する場合を想定する。図８は、ソースコード３０１及び３０３が関数間変更度計測装置１０に入力されることによって計測された当該ソースコード３０１に含まれる関数及び当該ソースコード３０３に含まれる関数間の変更度が表示された表示デバイス１０７の表示画面１０７ｂの一例を示す。

【００６２】

図８に示すように、表示画面１０７ｂには、ソースコード３０１に含まれる関数（バージョン１の関数）の関数名の予測結果、ソースコード３０３に含まれる関数（バージョン２．２の関数）の関数名の予測結果、ＫＬダイバージェンス値及び当該関数間の変更度が表示されている。

【００６３】

図８に示す例では、上記した図７と同様に、ソースコード３０１に含まれる関数の関数名の予測結果として、関数名候補「ｓｏｒｔ」及び当該関数名候補の確率「７２．６％」と、関数名候補「ｒｅｖｅｒｓｅ」及び当該関数名候補の確率「１５．６％」と、関数名候補「ｂｕｂｂｌｅｓｏｒｔ」及び当該関数名候補の確率「１１．８％」とが表示されている。

【００６４】

ソースコード３０３に含まれる関数の関数名の予測結果は、当該関数の複数の関数名候補及び当該複数の関数名候補の各々の確率を含む。図８に示す例では、ソースコード３０３に含まれる関数の関数名の予測結果として、関数名候補「ｓｏｒｔ」及び当該関数名候補の確率「７０．８％」と、関数名候補「ｂｕｂｂｌｅＳｏｒｔ」及び当該関数名候補の確率「２３．７％」と、関数名候補「ｒｅｖｅｒｓｅ」及び当該関数名候補の確率「５．５％」とが表示されている。

【００６５】

ＫＬダイバージェンス値は、ソースコード３０１に含まれる関数の関数名の予測結果に基づく確率分布（ソースコード３０１に含まれる関数の複数の関数名候補の各々の確率の分布）及びソースコード３０３に含まれる関数の関数名の予測結果に基づく確率分布（ソースコード３０３に含まれる関数の複数の関数名候補の各々の確率の分布）の差異を表す数値である。図８に示す例では、ＫＬダイバージェンス値として「１．２５４」が表示されている。

【００６６】

上記したようにＫＬダイバージェンス値＝０であれば変更度「無」が計測され、０＜ＫＬダイバージェンス値＜１．５であれば変更度「小」が計測され、ＫＬダイバージェンス値≧１．５であれば変更度「大」が計測されるものとすると、図８に示す例ではＫＬダイバージェンス値は「１．２５４」であるため、表示画面１０７ｂには、当該ＫＬダイバージェンス値に基づいて計測された変更度「小」が表示されている。

【００６７】

上記した図６に示すソースコード３０１に含まれる関数とソースコード３０３に含まれる関数とを比較すると、テキストは大幅に変更されているため、Ｄｉｆｆのようなテキスト比較ツールを用いた場合には変更度が大きいと判別される可能性が高い。しかしながら、ソースコード３０１に含まれる関数はバブルソートと称されるアルゴリズムに基づくソート関数であり、ソースコード３０３に含まれる関数はセレクションソートと称されるアルゴリズムに基づくソート関数であり、当該ソースコード３０１に含まれる関数及びソースコード３０３に含まれる関数は機能的には共通している。この場合、本実施形態においては、ＫＬダイバージェンス値に基づいて機能としての関数間の差異が小さいと判別されるため、変更度「小」を計測することができる。

【００６８】

なお、図７及び図８においては、ソースコードに含まれる関数の関数名の予測結果、ＫＬダイバージェンス値及び変更度が表示画面に表示されるものとして説明したが、当該表示画面には少なくとも変更度が表示されればよく、予測結果及びＫＬダイバージェンス値のうちの少なくとも一方は省略されても構わない。

【００６９】
　また、ここでは変更度「無」、「小」及び「大」の３つのうちの１つが計測されるものとして説明したが、例えば変更度「小」と変更度「大」」との間に相当する変更度「中」等が更に計測されてもよい。更に、本実施形態においては、上記したＫＬダイバージェンス値（つまり、関数名の予測結果に基づく確率分布の差異を表す数値）を変更度として用いてもよいし、当該ＫＬダイバージェンスに基づいて算出される数値を変更度として用いてもよい。すなわち、本実施形態における変更度は、任意の基準による解釈に基づくものであってもよい。
【００７０】
　上記したように本実施形態においては、第１関数を含む第１ソースコード及び第２関数を含む第２ソースコードを入力し、当該入力された第１ソースコードに含まれる第１関数を構文解析し、当該構文解析結果に基づいて当該第１関数の関数名を予測し、当該入力された第２ソースコードに含まれる第２関数を構文解析し、当該構文解析結果に基づいて当該第２関数の関数名を予測する。本実施形態においては、第１関数の関数名の予測結果及び第２関数の関数名の予測結果を比較することによって、第１及び第２関数間の変更度（第１変更度）が計測される。
【００７１】
　本実施形態においては、このような構成により、２つのソースコードに含まれる関数間の適切な変更度を計測することができる。
【００７２】
　具体的には、Ｄｉｆｆのようなテキスト比較ツールを用いて変更度を計測する場合には、変更度を計測する関数を構成する要素（空行及び空白等を含まない要素）が順番通りに並んで（記述されて）いなければ当該関数の共通部分として認識しないため、当該関数間で機能が共通していたとしても当該関数を表すテキスト（記述）が大幅に変更されている場合には、当該関数間の変更度が大きいと判別される可能性が高い。一方、関数間で機能が共通していない場合であっても、共通部分が多く含まれているような場合には、当該関数間の変更度が小さいと判別される可能性がある。
【００７３】
　これに対して、本実施形態においては、関数を表すテキスト（記述）が大幅に変更されている場合であっても、予測された関数名に基づく差異が小さいような場合には変更度が小さいと判別する（つまり、変更度「小」を計測する）ことができる。一方、関数を表現するテキストに共通部分が多く含まれているような場合であっても、予測された関数名に基づく差異が大きいような場合には、変更度が大きいと判別する（つまり、変更度「大」を計測する）ことができる。
【００７４】
　すなわち、本実施形態においては、機能的な変更に基づいて適切な関数間の変更度を計測することが可能となる。
【００７５】
　なお、本実施形態においては、関数を入力することによって当該関数名の予測結果を出力するように機械学習により生成された学習済モデルを用いて、第１ソースコードに含まれる第１関数及び第２ソースコードに含まれる第２関数の関数名が予測される構成により、精度の高い予測結果を得ることが可能となる。
【００７６】
　更に、上記した学習済モデルは関数名の予測結果として複数の関数名候補（関数名の複数の候補）と当該複数の関数名候補の各々が当該関数名である確率とを出力するように生成されており、本実施形態において、変更度は、第１関数の関数名の予測結果として学習済モデルから出力された第１関数の複数の関数名候補の各々が当該第１関数の関数名である確率の分布と、第２関数の関数名の予測結果として学習済モデルから出力された第２関数の複数の関数名候補の各々が当該第２関数の関数名である確率の分布との差異に基づいて計測される。このような構成によれば、学習済モデルを用いて適切な変更度を計測する

ことが可能となる。

【００７７】

なお、本実施形態においては、上記した第１関数の複数の関数名候補の各々が当該第１関数の関数名である確率の分布（第１関数の複数の関数名候補の確率分布）と第２関数の複数の関数名候補の各々が当該第２関数の関数名である確率の分布（第２関数の複数の関数名候補の確率分布）との差異がＫＬダイバージェンスまたはＪＳダイバージェンスを用いて計算されるものとして説明したが、当該差異は他の手法により計算されてもよい。

【００７８】

なお、ニューラルネットワーク（学習済モデル）上では関数名はベクトル表現として表されるため、本実施形態においては、当該ベクトル表現に基づく関数の差異を例えばコサイン類似度またはユークリッド距離を用いて計算し、当該差異に基づいて変更度を計測するように構成されていてもよい。

【００７９】

ここでは第１関数の関数名の予測結果（確率分布またはベクトル表現）と第２関数の関数名の予測結果（確率分布またはベクトル表現）との差異に基づいて変更度が計測されるものとして説明したが、当該変更度は、当該予測結果の類似度を用いて計測されても構わない。

【００８０】

なお、本実施形態においては、上記したように計測された変更度が表示（出力）されることによって、ユーザは第１及び第２関数間における変更の度合いを容易に把握することが可能であるが、本実施形態に係る関数間変更度計測装置１０は、例えば当該変更度に加えて、第１及び第２関数の関数名の予測結果（関数名候補及び確率）を更に表示（出力）する構成であってもよい。このような構成によれば、ユーザは、第１及び第２関数の関数名の予測結果に基づいて、当該第１及び第２関数間の変更の内容（概要）を把握することも可能となる。

【００８１】

また、本実施形態においては単に第１及び第２ソースコードが関数間変更度計測装置１０（入力部１１）によって入力されるものとして説明したが、当該第１及び第２ソースコードは、図９に示すソースコード管理ツール（構成管理ツール）１５において管理されていてもよい。ソースコード管理ツール１５は例えばソフトウェア製品を実現するためのソースコード（のバージョン等）を管理する機能を有しており、当該ソースコード管理ツール１５を利用することによって、本実施形態において説明した同種同系列のソフトウェア製品を実現するためのソースコードを効率的に取り出して入力するようなことが可能となる。すなわち、本実施形態においては、関数間変更度計測装置１０にソースコード管理ツール１５を埋め込むことによって、バージョン違いの関数をユーザが任意で指定し、当該指定された関数を含むソースコードを当該ソースコード管理ツール１５から取り出して関数名の予測及び変更度の計測を行うような構成を容易に実現することができる。

【００８２】

なお、本実施形態に係る関数間変更度計測装置１０は例えば派生開発を効率的に行うために用いられるが、当該関数間変更度計測装置１０によって計測される変更度は、ソースコードに含まれる関数を変更した際に、当該関数の変更内容が適切であるか否かを確認する際の指標として用いることも可能である。具体的には、リファクタリングにより関数が変更されたにもかかわらず、機能としての変更度が大きいと判別された（つまり、変更度「大」が計測された）ような場合には、当該関数に対して意図しない変更がされている可能性があるため、当該関数の確認を促すようなことが可能となる。一方、機能を変更するように関数を変更したにもかかわらず、機能としての変更度が小さいと判別された（つまり、変更度「小」が計測された）ような場合には、当該関数の確認を促すようなことが可能となる。また、同じ関数のバージョン違いのもので機能としての変更度が大きいと判別された場合には、元の関数から機能の内容が大幅に変更された可能性があるため、関数名自体を変更する必要性について確認を促すようにしてもよい。

【００８３】
　また、本実施形態においては、学習済モデルを用いてソースコードに含まれる関数から当該関数の関数名を予測し、当該予測された関数名の差異に基づいて関数間の変更度を計測するものとして説明したが、本実施形態は、例えばソースコード（に含まれる関数）が入力されることによって当該ソースコード（に含まれる関数）の要約を出力する（つまり、ソースコードからソースコードの要約を予測する）ように構築された学習済モデルを用いて、当該要約の差異に基づいてソースコード（に含まれる関数）間の変更度を計測するような構成としてもよい。すなわち、本実施形態に係る変更度計測装置は、第１及び第２ソースコードを入力し、当該入力された第１ソースコードの構文解析結果に基づいて当該第１ソースコードの要約を予測し、当該入力された第２ソースコードの構文解析結果に基づいて当該第２ソースコードの要約を予測し、当該要約を比較することによって、第１及び第２ソースコード間の変更度を計測するソースコード間変更度計測装置として実現されてもよい。この場合における学習済モデルから出力される予測結果は、例えばコメントのような形式のものであってもよいし、人手で任意に付される要約データのような形式のものであってもよい。
【００８４】
　また、本実施形態においては、ソースコードに記述されているクラスから予測されたクラス名の差異に基づいて当該ソースコード（クラス）間の変更度を計測する構成等とすることも可能である。
【００８５】
　すなわち、本実施形態は、入力されるソースコード（または当該ソースコードに含まれる関数）によって実現される機能に関する予測結果の差異に基づいて変更度を計測するように構成されていてもよい。
【００８６】
　（第２実施形態）
　次に、第２実施形態について説明する。本実施形態は、前述した第１実施形態において計測される変更度と、前述したＤｉｆｆのようなテキスト比較ツールを用いて計測される変更度とを出力する点で、当該第１実施形態とは異なる。
【００８７】
　図１０は、本実施形態に係る関数間変更度計測装置の機能構成の一例を示すブロック図である。図１０においては、前述した図１と同一の部分には同一参照符号を付して、その詳しい説明を省略する。なお、図１０に示す第１変更度計測部１３は、前述した第１実施形態において説明した変更度計測部１３と同一の機能部である。
【００８８】
　図１０に示すように、関数間変更度計測装置１０は、第２変更度計測部１６を含む。第２変更度計測部１６は、前述したＤｉｆｆのようなテキスト比較ツールを用いて、入力部１１によって入力された第１ソースコードに含まれる第１関数（を表すテキスト）及び第２ソースコードに含まれる第２関数（を表すテキスト）の差異に基づく変更度を計測する。
【００８９】
　なお、関数間変更度計測装置１０のシステム構成は前述した第１実施形態と同様であるが、上記した第２変更度計測部１６の一部または全ては、前述した図２に示すＣＰＵ１０１に関数間変更度計測プログラムを実行させること、すなわち、ソフトウェアによって実現されるものとする。なお、第２変更度計測部１６の一部または全ては、ハードウェアによって実現されてもよいし、ソフトウェア及びハードウェアの組み合わせによって実現されてもよい。
【００９０】
　次に、図１１のフローチャートを参照して、関数間変更度計測装置１０の処理手順の一例について説明する。
【００９１】
　まず、前述した図４に示すステップＳ１～Ｓ３の処理に相当するステップＳ１１～Ｓ１

３の処理が実行される。

【００９２】

次に、第１変更度計測部１３は、ステップＳ１３において取得された第１関数の複数の関数名候補の各々の分布（第１確率分布）と、第２関数の複数の関数名候補の各々の分布（第２確率分布）とに基づいて、第１及び第２関数間の第１変更度を計測する（ステップＳ１４）。

【００９３】

なお、ステップＳ１４の処理（つまり、当該ステップＳ１４において計測される第１変更度）は前述した図４に示すステップＳ４の処理（つまり、当該ステップＳ４において計測される変更度）と同様であるため、ここではその詳しい説明を省略する。本実施形態においては、例えばＫＬダイバージェンス値（関数名の予測結果に基づく確率分布の差異を表す数値）が第１変更度として計測されるものとする。また、本実施形態における第１変更度は、機能としての第１及び第２関数間の変更度に相当する。

【００９４】

次に、第２変更度計測部１６は、ステップＳ１２において切り出された第１及び第２関数間の第２変更度を、テキスト比較ツールを用いて計測する（ステップＳ１５）。本実施形態において、第２変更度は、例えば第１及び第２関数間のテキストの差異を表す数値として計測されるものとする。本実施形態における第２変更度は、テキストとしての第１及び第２関数間の変更度に相当する。

【００９５】

なお、詳細な説明については省略するが、ステップＳ１５においては、前述した第１実施形態における変更度と同様に、第１及び第２関数間でのテキストの差が大きい場合に変更度「大」が計測され、当該第１及び第２関数間でのテキストの差異が小さい場合に変更度「小」が計測されるような処理が実行されてもよい。

【００９６】

なお、ここでは便宜的にステップＳ１１～ステップＳ１５の処理が順に実行されるものとして説明したが、ステップＳ１３及びＳ１４の処理と、ステップＳ１５の処理とは、順番が入れ替えられてもよいし、並列に実行されてもよい。

【００９７】

ステップＳ１５の処理が実行されると、出力部１４は、ステップＳ１４において計測された第１変更度と、ステップＳ１５において計測された第２変更度とを出力する（ステップＳ１６）。

【００９８】

ここで、ステップＳ１６においては例えば第１及び第２変更度を表示デバイス１０７に表示することができるが、図１２は、第１及び第２変更度が表示された表示デバイス１０７の表示画面の一例を示している。

【００９９】

なお、図１２においては、上記したように第１変更度としてＫＬダイバージェンス値（つまり、第１及び第２関数の関数名の予測結果に基づく確率分布の差異を表す数値）が計測され、第２変更度として第１及び第２関数間のテキストの差異を表す数値が計測された場合を想定している。

【０１００】

図１２に示す表示画面１０７ｃにおいては、例えば横軸を第１変更度、縦軸を第２変更度とするグラフ上に、計測された第１及び第２変更度の対応関係をプロットすることにより、当該第１及び第２変更度が表示されている。これによれば、第１及び第２変更度の比較結果を容易に把握することが可能となる。

【０１０１】

なお、図１２に示す領域４０１に第１及び第２変更度の対応関係がプロットされる場合には、機能としての第１及び第２関数間の変更度は小さいが、テキストとしての第１及び第２関数間の変更度は大きいことを意味している。また、図１２に示す領域４０２に第１

及び第２変更度の対応関係がプロットされる場合には、機能としての第１及び第２関数間の変更度は大きいが、テキストとしての第１及び第２関数間の変更度は小さいことを意味している。

【０１０２】

一方、例えば領域４０３に第１及び第２変更度の対応関係がプロットされる場合には、機能としての第１及び第２関数間の変更度及びテキストとしての第１及び第２関数間の変更度が概ね同様であることを意味している。

【０１０３】

表示画面１０７ｃには１つの第１及び第２変更度の対応関係のみがプロットされた例が示されているが、例えば第１及び第２ソースコードの各々に含まれる複数の関数のうちの変更度が計測される２つの関数の組毎に第１及び第２変更度が計測された場合には、図１３に示すような表示画面１０７ｄを表示することも可能である。この表示画面１０７ｄによれば、第１及び第２ソースコード全体の変更度を把握することも可能である。

【０１０４】

上記したように本実施形態においては、第１ソースコードに含まれる第１関数を表すテキストと第２ソースコードに含まれる第２関数を表すテキストとを比較することによって第１及び第２関数間の第２変更度を計測し、当該計測された第２変更度を第１変更度とともに出力する構成により、ユーザは、第２変更度をも考慮して、第１及び第２関数間における変更の度合いを把握することができる。

【０１０５】

更に、本実施形態においては、上記した図１２及び図１３に示すようなグラフ上に第１及び第２変更度（の対応関係）を表示することにより、ユーザは、当該第１及び第２変更度の比較結果を容易に把握することができる。

【０１０６】

なお、本実施形態においては第１及び第２変更度をグラフ上に表示するものとして説明したが、当該第１及び第２変更度は、表形式等の他の形式で表示されてもよい。更に、第１及び第２変更度は前述した第１実施形態において説明したように変更度「無」、「小」及び「大」のように表されてもよく、このような第１及び第２変更度を比較可能な態様で表示してもよい。

【０１０７】

すなわち、本実施形態においては、第１及び第２変更度を出力し、当該出力された第１及び第２変更度をユーザが把握することができる態様であれば、当該第１及び第２変更度の内容及び表示形式等は、適宜、変更されても構わない。

【０１０８】

以上述べた少なくとも１つの実施形態によれば、ソースコードに含まれる関数間の変更度を計測することが可能な変更度計測装置、方法及びプログラムを提供することができる。

【０１０９】

本発明のいくつかの実施形態を説明したが、これらの実施形態は、例として提示したものであり、発明の範囲を限定することは意図していない。これら実施形態は、その他の様々な形態で実施されることが可能であり、発明の要旨を逸脱しない範囲で、種々の省略、置き換え、変更を行うことができる。これら実施形態やその変形は、発明の範囲や要旨に含まれると同様に、特許請求の範囲に記載された発明とその均等の範囲に含まれるものである。

【符号の説明】

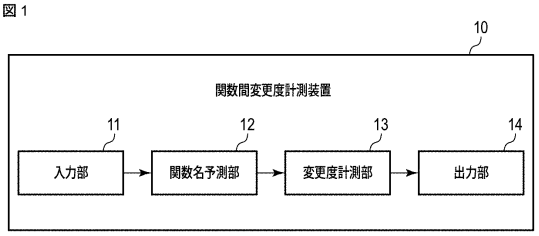
【０１１０】

１０…関数間変更度計測装置、１１…入力部、１２…関数名予測部、１３…変更度計測部，第１変更度計測部、１４…出力部、１５…ソースコード管理ツール、１６…第２変更度計測部、１０１…ＣＰＵ、１０２…不揮発性メモリ、１０３…主メモリ、１０４…ＢＩＯＳ－ＲＯＭ、１０５…システムコントローラ、１０６…入力デバイス、１０７…表示デバイス、１０８…ＥＣ。

【図面】

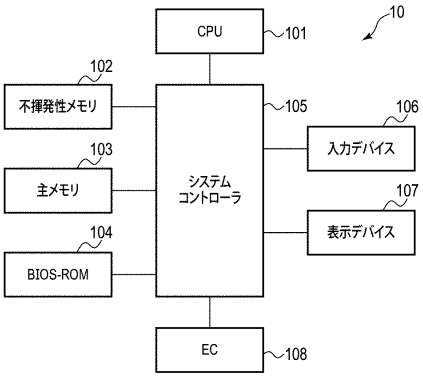
【図１】

図１



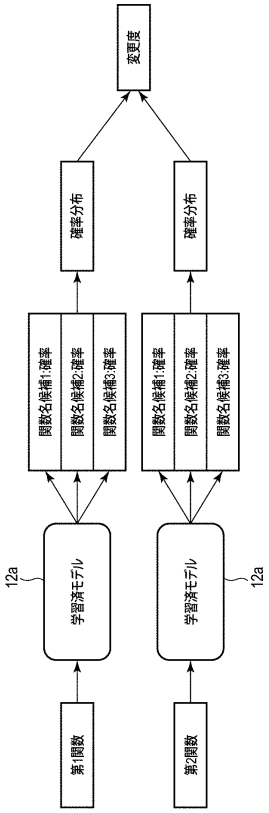
【図２】

図２



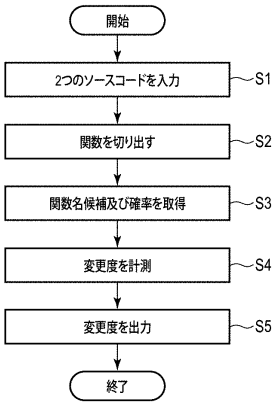
【図３】

図３



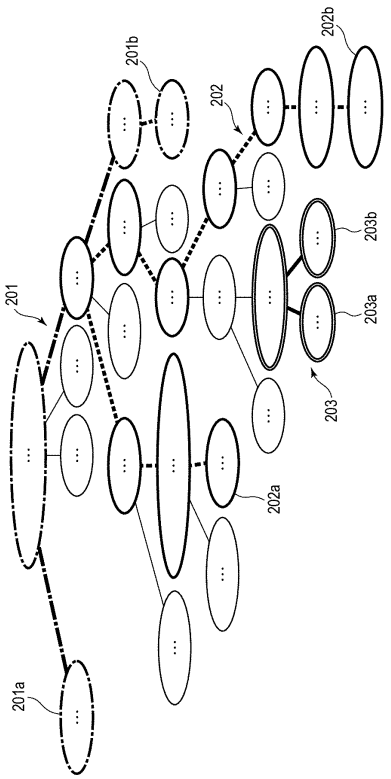
【図４】

図４



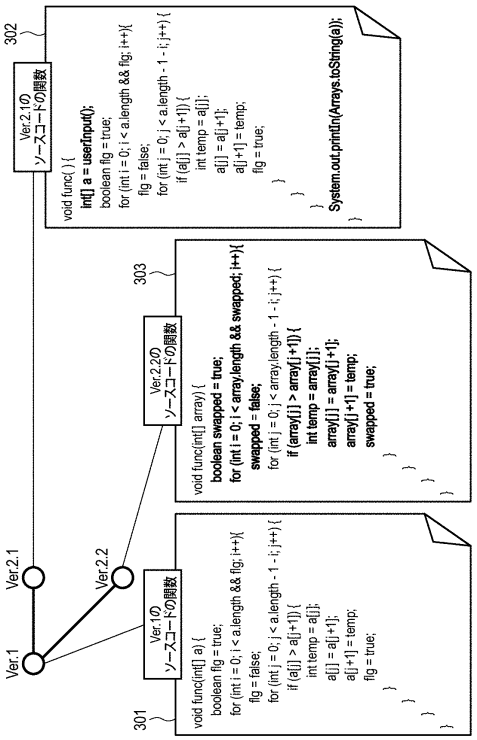
【図５】

図５



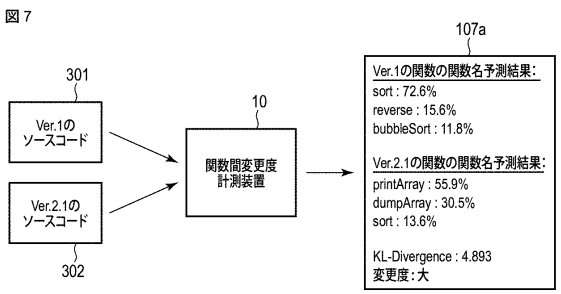
【図６】

図６



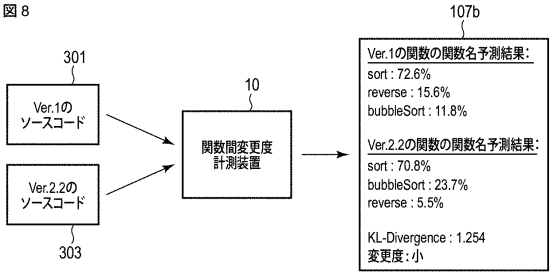
【図７】

図７



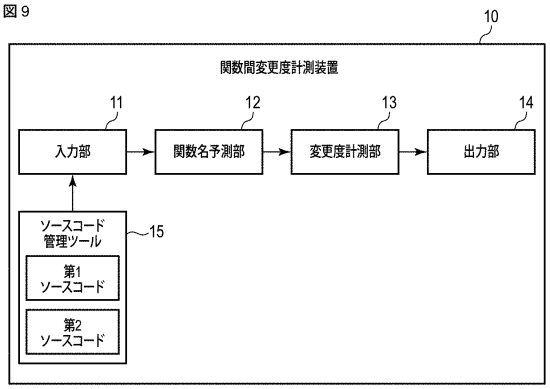
【図８】

図８



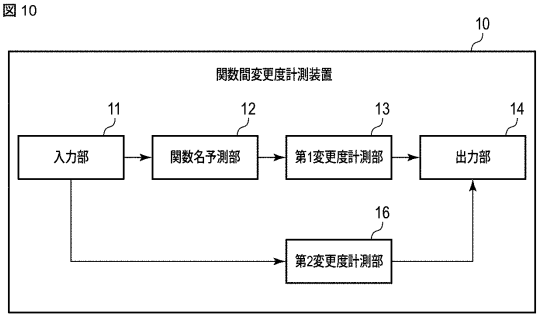
【図９】

図９



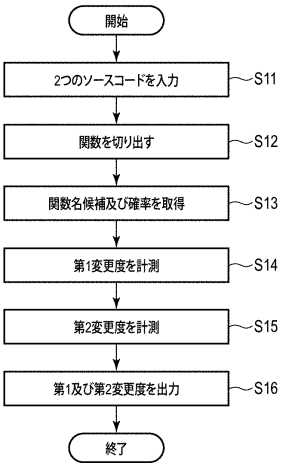
【図１０】

図１０



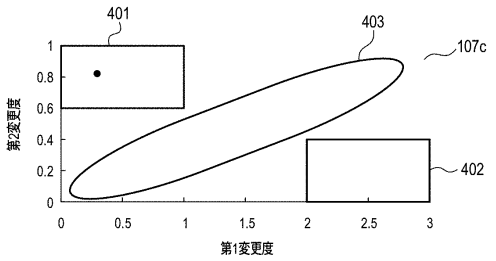
【図１１】

図１１



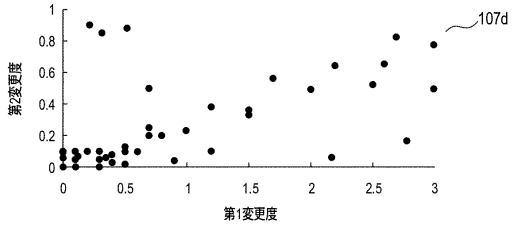
【図１２】

図１２



【図１３】

図13



フロントページの続き

(56)参考文献 特開２０２０－１６０８５４（ＪＰ，Ａ）
Uri Alon, Meital Zilberstein, Omer Levy, Eran Yahav，code2vec: Learning Distributed Representations of Code，arXiv [online]，2018年10月30日，[retrieved on 2024.06.27], retrieved from the Internet: 〈URL: https://arxiv.org/abs/1803.09473〉
Uri Alon, Shaked Brody, Omer Levy, Eran Yahav，code2seq: Generating Sequences from Structured Representations of Code，arXiv [online]，2019年02月21日，[retrieved on 2024.06.27], retrieved from the Internet: 〈URL: https://arxiv.org/abs/1808.01400〉
金 明哲，テキストアナリティクス１ テキストアナリティクスの基礎と実践 ，2021年03月23日，pp.137-141
土居通夫、阿萬裕久、山田宏之，クラスの再利用頻度と保守性の関係に関する一考察，電子情報通信学会技術研究報告，Ｖｏｌ．１０５ Ｎｏ．２０７，2005年07月18日，pp.25-30

(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ８／７７