(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2020-4036
(P2020-4036A)
(43)公開日　令和2年1月9日(2020.1.9)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 8/35	(2018.01)	G06F	8/35		5B042
G06F 11/36	(2006.01)	G06F	11/36	184	5B376

審査請求 未請求　請求項の数 4　ＯＬ　（全 15 頁）

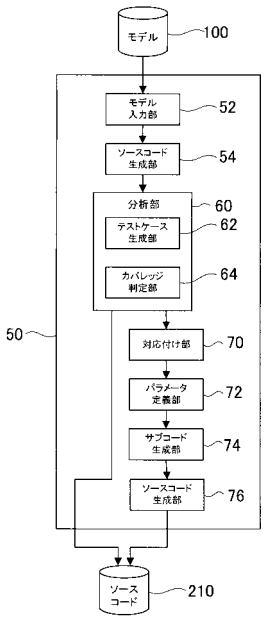
(21)出願番号　特願2018-122141 (P2018-122141)
(22)出願日　平成30年6月27日 (2018.6.27)

(71)出願人　000004260
株式会社デンソー
愛知県刈谷市昭和町１丁目１番地
(74)代理人　110000578
名古屋国際特許業務法人
(72)発明者　佐川　文崇
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内

Ｆターム(参考)　5B042 HH07　HH12　HH17　HH18
5B376 BC33　BC63　BC67

(54)【発明の名称】ソースコード生成装置

(57)【要約】
【課題】モデルを分割したサブシステム毎にサブソースコードを生成しても、モデルに対応してサブソースコードから生成されるソースコードの実行時間の増加を抑制しつつ、サブシステム毎のテストケースを実行可能にする技術を提供する。
【解決手段】本開示のソースコード生成装置５０によると、モデル１００のカバレッジが所定値未満であり、モデルを分割して生成された複数のサブシステムのそれぞれのカバレッジが所定値以上であるとカバレッジ判定部６４が判定すると、パラメータ定義部７２は、サブシステム毎に、対応付け部７０がモデルから生成された元のソースコードと処理ブロックの入出力パラメータとを対応付けた結果において、元のソースコードで定義されていない入出力パラメータを定義する。ソースコード生成部７６は、サブコード生成部７４がサブシステム毎に生成するサブソースコードから、モデルに対応するソースコード２１０を生成する。
【選択図】図２



【特許請求の範囲】
【請求項１】
入力データに対する処理を複数の処理ブロック（１１２、１２０～１２８）間におけるデータの入出力の仕様として記述されるモデル（１２、１００）から、前記モデルに対応するソースコード（１４、２１０）を生成するソースコード生成装置（５０）であって、
前記モデル（１００）から生成されるテストケースを前記モデルに実行させたカバレッジが所定値未満の場合、前記モデルを分割して生成された複数のサブシステム（１３０、１４０）のそれぞれに対し、前記サブシステムから生成されるテストケースを実行させた前記カバレッジが所定値以上か否かを判定するように構成されたカバレッジ判定部（６４、Ｓ４０６、Ｓ４１４）と、
前記複数のサブシステムの前記カバレッジがすべて前記所定値以上であると前記カバレッジ判定部が判定すると、前記サブシステム毎に、前記サブシステムに分割される前の前記モデルから生成された元のソースコード（２００）と、前記処理ブロックおよび前記処理ブロックの入出力パラメータとを対応付けるように構成された対応付け部（７０、Ｓ４３０、Ｓ４３２）と、
前記対応付け部が対応付けた結果において、前記元のソースコードで定義されていない前記入出力パラメータが存在すれば、前記サブシステム毎に、定義されていない前記入出力パラメータを定義するように構成されたパラメータ定義部（７２、Ｓ４３４）と、
前記サブシステム毎に、前記対応付け部が対応付けた結果と前記パラメータ定義部が定義する前記入出力パラメータとに基づいて、前記サブシステムに対応するサブソースコード（２１２、２１４）を生成するように構成されたサブコード生成部（７４、Ｓ４３６～Ｓ４４８）と、
前記サブコード生成部が生成する前記サブソースコードから、前記モデルに対応するソースコードを生成するように構成されたソースコード生成部（７６、Ｓ４５０）と、
を備えるソースコード生成装置。
【請求項２】
請求項１に記載のソースコード生成装置であって、
前記カバレッジ判定部は、前記サブシステムに前記テストケースを実行させた前記カバレッジが所定値未満の場合、前記カバレッジが所定値未満の前記サブシステムをさらに分割して生成された複数のサブシステムのそれぞれに対し、生成されたテストケースを実行させた前記カバレッジが所定値以上であるか否かを判定するように構成されている、
ソースコード生成装置。
【請求項３】
請求項１または２に記載のソースコード生成装置であって、
前記サブコード生成部は、前記サブシステムにおいて後段の処理を実行する前記処理ブロックから順番に、前記パラメータ定義部が定義する前記入出力パラメータを使用して、前記元のソースコードに基づいて、前記処理ブロックの処理に対応する前記サブソースコードを生成するように構成されている、
ソースコード生成装置。
【請求項４】
請求項３に記載のソースコード生成装置であって、
前記サブコード生成部は、前記処理ブロックが条件分岐の処理を実行し、いずれかの分岐先の処理に対応する記述が前記元のソースコードに存在しない場合、記述が存在しない分岐先の処理を表すサブコードを生成するように構成されている、
ソースコード生成装置。

【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、モデルからソースコードを生成する技術に関する。

【背景技術】
【０００２】
入力データに対する処理を複数の処理ブロック間におけるデータの入出力の仕様として記述されるモデルから、モデルに対応するソースコードを生成する技術として、例えば、特許文献１に記載されているものが知られている。
【０００３】
特許文献１に記載されているように、モデルまたはソースコードに対しては、モデルまたはソースコードの品質を検査するためにテストケースが生成される。テストケースは、テスト対象のモデルまたはソースコードに対する入力データである。通常、テストケースは、モデルの出力側から入力側に向けて逆算して生成される。
【０００４】
そして、モデルまたはソースコードがテストケースを実行した結果、モデルまたはソースコードにおける処理経路がどの程度テストされたかを表すカバレッジは、テストケースによる品質検査を保証するために必要な所定値以上であることが要求される。
【先行技術文献】
【特許文献】
【０００５】
【特許文献１】特開２００９－２９４８４６号公報
【発明の概要】
【発明が解決しようとする課題】
【０００６】
モデルが大型化し複雑になると、所定値以上のカバレッジを達成することが困難になる場合がある。そこで、モデルを分割し小型化して複数のサブシステムを生成し、サブシステム毎にテストケースを生成してカバレッジを向上させることが考えられる。
【０００７】
この場合、サブシステムに対応するサブソースコードを関数として定義すれば、サブソースコードからモデルに対応するソースコードを容易に生成でき、サブシステム毎のテストケースをサブシステムまたはサブソースコードで実行できる。
【０００８】
しかし、サブソースコードが関数として定義されると、定義される関数の数が上昇するのに従い関数の呼び出し回数が増加する。その結果、サブソースコードから生成されるソースコードの実行時間が増加するという課題が生じる。また、関数として定義しない場合には、サブシステムの入出力に相当するものがソースコードに無く、サブシステムのテストケースがソースコード上で実行できないという課題が生じる。
【０００９】
本開示は、モデルを分割したサブシステム毎にサブソースコードを生成しても、モデルに対応してサブソースコードから生成されるソースコードの実行時間の増加を抑制しつつ、サブシステム毎のテストケースを実行可能にする技術を提供する。
【課題を解決するための手段】
【００１０】
本開示のソースコード生成装置（５０）は、入力データに対する処理を複数の処理ブロック（１１２、１２０～１２８）間におけるデータの入出力の仕様として記述されるモデル（１２、１００）から、モデルに対応するソースコード（１４、２１０）を生成するソースコード生成装置であって、カバレッジ判定部（６４、Ｓ４０６、Ｓ４１４）と、対応付け部（７０、Ｓ４３０、Ｓ４３２）と、パラメータ定義部（７２、Ｓ４３４）と、サブコード生成部（７４、Ｓ４３６～Ｓ４４８）と、ソースコード生成部（７６、Ｓ４５０）と、を備える。
【００１１】
カバレッジ判定部は、モデルから生成されるテストケースをモデルに実行させたカバレッジが所定値未満の場合、モデルを分割して生成された複数のサブシステム（１３０、１

４０）のそれぞれに対し、サブシステムから生成されるテストケースを実行させたカバレッジが所定値以上か否かを判定する。

【００１２】

対応付け部は、複数のサブシステムのカバレッジがすべて所定値以上であるとカバレッジ判定部が判定すると、サブシステム毎に、サブシステムに分割される前のモデルから生成された元のソースコード（２００）と、処理ブロックおよび処理ブロックの入出力パラメータとを対応付ける。

【００１３】

パラメータ定義部は、対応付け部が対応付けた結果において、元のソースコードで定義されていない入出力パラメータが存在すれば、サブシステム毎に、定義されていない入出力パラメータを定義する。

【００１４】

サブコード生成部は、サブシステム毎に、対応付け部が対応付けた結果とパラメータ定義部が定義する入出力パラメータとを使用して、サブシステムに対応するサブソースコード（２１２、２１４）を生成する。ソースコード生成部は、サブコード生成部が生成するサブソースコードから、モデルに対応するソースコードを生成する。

【００１５】

このように、本開示のソースコード生成装置では、モデルを分割したサブシステム毎に、分割前のモデルから生成された元のソースコードにおいて定義されていない入出力パラメータを新たに定義する。そして、サブシステム毎に、対応付け部が対応付けた結果とパラメータ定義部が定義する入出力パラメータとに基づいて、サブシステムに対応するサブソースコードが生成される。これにより、サブシステムに対応して生成されるサブソースコードとサブソースコードとの間で、入出力パラメータによりデータの受け渡しができる。

【００１６】

したがって、サブシステムから関数として定義されるサブソースコードを生成する場合に比べ、関数の呼び出しにより生じる実行時間の増加を抑制しつつ、サブシステム毎のテストケースを実行できる。

【００１７】

尚、この欄および特許請求の範囲に記載した括弧内の符号は、一つの態様として後述する実施形態に記載の具体的手段との対応関係を示すものであって、本開示の技術的範囲を限定するものではない。

【図面の簡単な説明】

【００１８】

【図１】モデルベース開発システムを示すブロック図。
【図２】ソースコード生成装置を示すブロック図。
【図３】ソースコード生成のメイン処理を示すフローチャート。
【図４】ソースコード置換処理を示すフローチャート。
【図５】モデルの構成を示す説明図。
【図６】モデルから生成されたソースコードを示すプログラム。
【図７】モデルの分割を示す説明図。
【図８】サブシステムとソースコードとの対応を示す一覧図。
【図９】サブシステムを示す説明図。
【図１０】未定義の入出力パラメータを定義した一覧図。
【図１１】処理ブロックのソースコードの置換を説明する一覧図。
【図１２】処理ブロックの他のソースコードの置換を説明する一覧図。
【図１３】他の処理ブロックのソースコードの置換を説明する一覧図。
【図１４】他の処理ブロックのソースコードの置換を説明する一覧図。
【図１５】モデルに対応する置換終了後のソースコードを示すプログラム。

【発明を実施するための形態】

【００１９】
以下、本開示の実施形態を図に基づいて説明する。
［１．構成］
図１に示すモデルベース開発システム２は、ソースコード生成部１０と、モデル１２と、ソースコード１４と、テストケース生成部２０と、ソースコード実行部３０と、モデル実行部３２と、実行結果比較部４０と、結果表示部４２とを備えている。モデルベース開発システム２は、例えば、ディスプレイ、入出力装置、ＣＰＵ、ＲＡＭ、ＲＯＭ、ＨＤＤ等を備えるパーソナルコンピュータにより実現される。
【００２０】
以下、モデル１２から生成されたソースコード１４のカバレッジが所定値以上の場合のモデルベース開発システム２の作動について説明する。
ソースコード生成部１０は、ＨＤＤ等に記憶されているモデル１２から、例えばＣ言語で記述されたソースコード１４を生成する。
【００２１】
ソースコード生成部１０の例としては、Ｍａｔｈｗｏｒｋｓ（登録商標）社のＥｍｂｅｄｄｅｄ　Ｃｏｄｅｒ（登録商標）がある。なお、モデルは、モデルエディタ等のモデル作成用プログラムをパーソナルコンピュータで実行し、その実行時にユーザがモデル作成のための作業を入力装置を用いて行うことで作成されるようになっていてもよい。このようなモデルエディタがモデルベース開発システム２に含まれていてもよい。
【００２２】
ここで、モデル１２について説明する。モデル１２は、時系列の中でのデータ演算、データ入力およびデータ出力のうち少なくともいずれか１つの機能を表す処理ブロック、および処理ブロック間の入出力関係の仕様、すなわち、処理ブロックからどの処理ブロックにデータが渡されるかを示す結線の組み合わせとして記述される。
【００２３】
これら処理ブロックおよび結線の組み合わせによって、入力データに応じたモデル１２の処理が記述される。例えば、車載装置の制御処理であれば、エンジン制御、ブレーキ制御等の処理がある。モデルの例として、Ｍａｔｈｗｏｒｋｓ（登録商標）社のＳｉｍｕｌｉｎｋ（登録商標）モデルがある。
【００２４】
テストケース生成部２０は、モデル１２に基づいてテストケースを生成する。なお、テストケースは、モデル１２のみに基づいて生成してもよいし、あるいは、モデル１２のソースコード１４に基づいて生成してもよいし、またあるいは、モデル１２とソースコード１４との両方に基づいて生成してもよい。
【００２５】
テストケースとは、モデル１２の品質、および、ソースコード生成部１０によってモデル１２から生成されたソースコード１４の品質を検査するためのカバレッジテスト等の処理の実行において、その品質検査が十分となるよう、モデル１２やソースコード１４に入力するテスト用データである。例えば、後述するソースコード実行部３０およびモデル実行部３２の実行の際にも、このテストケースが使用される。
【００２６】
品質検査が十分となるようなテストケースとは、例えば、できる限りモデル１２の全ての処理ブロックまたはソースコード１４の全ての命令文が実行されるような入力データである。
ソースコード実行部３０は、テストケース生成部２０によって生成されたテストケースを入力データとして、ソースコード１４に記載された処理内容を実行する。その実行の際、ソースコード実行部３０は、ソースコード１４の実行単位による出力データ、および実行した命令文の時系列的なリストを生成する。
【００２７】
モデル実行部３２は、テストケース生成部２０によって生成されたテストケースを入力

データとして、モデル１２のシミュレーションを実行する。シミュレーションとは、モデル１２の表現するデータの入出力の時間変化をパーソナルコンピュータ上で仮想的に再現する処理をいう。

【００２８】

モデル実行部３２は、モデル１２のシミュレーションによる外部への出力データ、ならびに処理ブロック毎の実行時間および入出力データをＨＤＤ等に格納する。

実行結果比較部４０は、ソースコード実行部３０およびモデル実行部３２によって実行された実行結果データを比較し、その比較結果をＨＤＤ等に格納する。尚、比較とは、ソースコード実行部３０が実行したソースコード１４と、モデル実行部３２がシミュレーションを行ったモデル１２とのそれぞれの実行結果のデータの比較をいう。

【００２９】

比較項目としては、同じ入力データに対してモデル１２、ソースコード１４が出力したデータがある。そして実行結果比較部４０は、比較の結果、それぞれが合致しているか否かの合否判定を行い、その合否判定の情報もＨＤＤ等に格納する。結果表示部４２は、実行結果比較部４０による比較結果をディスプレイに出力する。

【００３０】

ここで、モデル１２の構造が大型化しており複雑な場合、テストケース生成部２０が生成するテストケースをソースコード実行部３０とモデル実行部３２とが実行しても、カバレッジが、モデル１２またはソースコード１４に対するテストケースによる品質検査を保証するために必要な所定値以上にならないことがある。

【００３１】

そこで、図２に、構造が大型化しており複雑な場合のモデル１００であっても、ソースコード２１０でテストケースを実行した結果のカバレッジを所定値以上にする本実施形態のソースコード生成装置５０を示す。

【００３２】

ソースコード生成装置５０は、モデル入力部５２と、ソースコード生成部５４と、分析部６０と、対応付け部７０と、パラメータ定義部７２と、サブコード生成部７４と、ソースコード生成部７６とを備えている。分析部６０は、テストケース生成部６２とカバレッジ判定部６４とを備えている。

【００３３】

ソースコード生成装置５０の各要素が実行するソースコード生成処理について以下に説明する。

［２．処理］

［２－１．メイン処理］

ソースコード生成装置５０が実行するソースコード生成のメイン処理について、図３のフローチャートに基づいて説明する。

【００３４】

Ｓ４００においてモデル入力部５２は、モデルと、モデルに付随する情報として、モデルを分割してサブシステムを生成する場合の分割箇所などを取得する。Ｓ４０２においてソースコード生成部５４は、モデルからソースコードを生成する。この場合、ソースコード生成部５４は、図１に示すソースコード生成部１０と実質的に同じ処理を実行する。

【００３５】

図５に、モデル１００の一例を示す。符号１０２～１０８は入力パラメータを示す。符号１１０は出力パラメータを示す。符号１１２、１２０～１２８は処理ブロックである。符号１１２は、１．０１の定数を出力する定数ブロックである。符号１２０は、定数ブロック１１２の値と、遅延処理ブロック１２８の出力との積を出力する乗算ブロックである。

【００３６】

符号１２２～１２６が示す条件分岐ブロックは、上中下の入力パラメータのうち、中央の信号が０でないときに上の信号を出力し、中央の信号が０の場合は下の信号を出力する

。
符号１２８が示す遅延処理ブロックは、１周期前の値を出力する。
【００３７】
図５に示されるモデル１００から、ソースコード生成部５４が生成したＣ言語によるソースコード２００を図６に示す。
Ｓ４０４において分析部６０のテストケース生成部６２は、モデル１００に実行させるテストケースを生成する。この場合、テストケース生成部６２は、図１に示すテストケース生成部２０と実質的に同じ処理を実行する。Ｓ４０６において分析部６０のカバレッジ判定部６４は、モデル１００にテストケースを実行させた結果のカバレッジが所定値以上であるか否かを判定する。
【００３８】
Ｓ４０６の判定がＹｅｓである、つまりモデル１００にテストケースを実行させた結果のカバレッジが所定値以上の場合、Ｓ４０８において、図１に示すモデルベース開発システム２により、モデル１００と、Ｓ４０２で生成されたソースコード２００とに対してテストが実行される。
【００３９】
これに対し、Ｓ４０６の判定がＮｏである、つまりモデル１００にテストケースを実行させた結果のカバレッジが所定値未満の場合、Ｓ４１０においてカバレッジ判定部６４は、Ｓ４００でモデル入力部５２が取得したモデル１００の分割箇所でモデル１００を分割し複数のサブシステムを生成する。
【００４０】
図７に、Ｓ４００でモデル入力部５２が取得した分割情報に基づいてモデル１００を２つのサブシステム１３０、１４０に分割した例を示す。
Ｓ４１２においてテストケース生成部６２は、サブシステム１３０、１４０のそれぞれについてテストケースを生成する。
【００４１】
Ｓ４１４においてカバレッジ判定部６４は、サブシステム１３０、１４０のそれぞれについてテストケースを実行した結果のカバレッジが所定値以上であるか否かを判定する。
Ｓ４１４の判定がＮｏである、つまりカバレッジが所定値未満のサブシステムが存在する場合、処理はＳ４１０に戻り、カバレッジが所定値未満のサブシステムがさらに複数のサブシステムに分割される。
【００４２】
Ｓ４１４の判定がＹｅｓである、つまりすべてのサブシステム、図７の例ではサブシステム１３０、１４０のカバレッジが所定値以上の場合、Ｓ４１６のソースコード置換処置において、対応付け部７０とパラメータ定義部７２とサブコード生成部７４とソースコード生成部７６とは、Ｓ４０２で生成されたソースコードを置換して、複数のサブシステム１３０、１４０のそれぞれに対応するサブソースコードを生成する。
【００４３】
そして、サブソースコードをまとめて、モデル１００に対応するソースコードを生成する。それぞれのサブシステム１３０、１４０のカバレッジが所定値以上であれば、それぞれのサブシステム１３０、１４０から生成されたサブソースコードをまとめて生成されたモデル１００に対応するソースコードのカバレッジも所定値以上である。
【００４４】
尚、Ｓ４１６のソースコード置換処理については、図４のフローチャートに基づいて詳細に説明する。
そして、Ｓ４１８において、Ｓ４１６で生成されたサブシステム１３０、１４０に対応するサブソースコードと、サブシステム１３０、１４０とに対して、図１に示すモデルベース開発システム２においてソースコード実行部３０とモデル実行部３２とが、Ｓ４１２で生成されたテストケースを使用してテストを実行する。
【００４５】

［２－２．ソースコード置換処理］
図３のＳ４１６においてソースコード生成装置５０が実行するソースコード置換処理について、図４のフローチャートに基づいて説明する。
【００４６】
Ｓ４３０において対応付け部７０は、モデル１００を分割して生成されたサブシステム１３０、１４０に対し置換処理を実行する順番を決定する。モデル１００において早く実行されるサブシステム１３０から置換処理が実行される。
【００４７】
Ｓ４３２において対応付け部７０は、サブシステム１３０、１４０のそれぞれについて、処理ブロックと入出力パラメータと一時パラメータとに対し、ソースコード２００を対応付けた一覧を生成する。図８に、対応付け部７０がサブシステム１３０についてソースコード２００を対応付けて生成した一覧を示す。
【００４８】
図８の一覧と図９に示すサブシステム１３０とにおいて、Ａ、Ｂ、Ｃが処理ブロックであり、ａ、ｂ、ｃ、ｄが入力パラメータであり、ｅが出力パラメータであり、ｆ、ｇが一時パラメータである。ｆ、ｇが示す一時パラメータも、いずれかの処理ブロックにおける入力パラメータまたは出力パラメータである。
【００４９】
処理ブロックは、処理の種別として、反復、分岐、順接に分類される。分岐、順接が反復される場合は、反復／分岐、反復／順接として分類される。
各処理ブロックに対応付けるソースコードは、反復および順接については、処理ブロックに関連するソースコードの記述がすべて抽出される。分岐については、複数の分岐の条件がソースコードに記述されている場合、該当する分岐の条件だけが抽出される。
【００５０】
図８の一覧において、入出力パラメータとして、「Ａ」で表される条件分岐ブロック１２４の出力パラメータの「ｅ」と、「Ｂ」で表される条件分岐ブロック１２２の出力パラメータであり条件分岐ブロック１２４の入力パラメータである一時パラメータの「ｆ」と、「Ｃ」で表される乗算ブロック１２０の出力パラメータであり条件分岐ブロック１２２の入力パラメータである一時パラメータの「ｇ」とに対応する記述が、図６に示すソースコード２００において未定義であることが分かる。
【００５１】
そこで、Ｓ４３４においてパラメータ定義部７２は、一覧において未定義の入出力パラメータとして、出力パラメータの「ｅ」と一時パラメータの「ｆ」、「ｇ」とを、図１０に示すように、double sub1_out1[2]、double sub1_tmp1[2]、double sub1_tmp2[2]として、図１０の下線で示すように処理後のサブソースコードで定義する。
【００５２】
Ｓ４３６においてサブコード生成部７４は、サブシステム１３０において、ソースコードを置換してサブソースコードを生成する処理ブロックの順番を決定する。本実施形態では、サブコード生成部７４は、乗算ブロック１２０と条件分岐ブロック１２２、１２４とのうち、条件分岐ブロック１２２、１２４を先に置換する。さらに、サブコード生成部７４は、条件分岐ブロック１２２、１２４のうち、後段で分岐処理が実行される条件分岐ブロック１２４を条件分岐ブロック１２２よりも先に置換する。
【００５３】
これは、サブシステム１３０からサブソースコードが生成された場合、後段の条件分岐ブロックに対応するサブソースコードから実行されるので、分岐条件のいずれかが満たされない場合、分岐条件が満たされない場合に実行されるサブソースコードが実行されない。これにより、生成されたサブソースコードの実行時間が短縮される。
【００５４】
Ｓ４３６の処理により、サブシステム１３０において、条件分岐ブロック１２４、条件分岐ブロック１２２、乗算ブロック１２０の順番で、対応するソースコードが置換される

。
Ｓ４３８においてサブコード生成部７４は、今回の置換処理の対象である処理ブロックに対応するソースコードの記述において、分岐先の処理が省略されているか否かを判定する。Ｓ４３８の判定がＹｅｓである、つまりソースコードにおいて分岐先の処理が省略されている場合、処理はＳ４４０に移行する。Ｓ４３８の判定がＮｏである、つまり分岐先の処理が省略されていない場合、処理はＳ４４２に移行する。
【００５５】
例えば、条件分岐ブロック１２４に対応する「if (!(in3[s4t_tmp] != 0.0F))」の判定に対し、ソースコード２００には判定結果が偽の場合の分岐先の記述が省略されているので、Ｓ４３８の判定はＹｅｓである。
【００５６】
この場合、Ｓ４４０においてサブコード生成部７４は、図１１の下線で示すように、条件分岐ブロック１２４に対応するサブソースコードにおいて、「else」が追加される。
Ｓ４４２において、サブコード生成部７４は、処理ブロックに対応する処理で、出力パラメータに入力パラメータを代入する記述を追加する。
【００５７】
例えば、図１２の下線で示すように、条件分岐ブロック１２４に対応する「if (!(in3[s4t_tmp] != 0.0F))」の判定に対し、判定結果が真であれば、sub1_out1[s4t_tmp]=out1[s4t_tmp]の記述が追加され、判定結果が偽であれば、sub1_out1[s4t_tmp]=sub1_tmp1[s4t_tmp]の記述が追加される。
【００５８】
Ｓ４４４においてサブコード生成部７４は、置換処理が終了した処理ブロックを対象から除く。Ｓ４４６においてサブコード生成部７４は、すべての処理ブロックに対する置換処理が終了したか否かを判定する。Ｓ４４６の判定がＮｏである、つまり置換処理が未処理の処理ブロックが存在する場合、処理はＳ４３６に移行する。
【００５９】
例えば、条件分岐ブロック１２４に対する置換処理が終了しても、条件分岐ブロック１２２対する置換処理は終了していないので、処理はＳ４３６に移行し、条件分岐ブロック１２４と同様に条件分岐ブロック１２２に対する置換処理が実行される。
【００６０】
条件分岐ブロック１２２の処理は条件分岐ブロック１２４において判定結果が偽である「else」側で実行されるので、図１３の下線で示すように、サブソースコードにおいて条件分岐ブロック１２２の処理に対応する記述が条件分岐ブロック１２４のサブソースコードにおける「else」側に記述される。
【００６１】
この場合、条件分岐ブロック１２４の判定結果が偽のときに実行されるsub1_out1[s4t_tmp]=sub1_tmp1[s4t_tmp]において、条件分岐ブロック１２４の判定結果が真であればsub1_tmp1[s4t_tmp]=in1[s4t_tmp]が実行され、条件分岐ブロック１２４の判定結果が偽であればsub1_tmp1[s4t_tmp]=sub1_tmp2[s4t_tmp]が実行される。
【００６２】
そして、条件分岐ブロック１２２に対する置換処理が終了すると、乗算ブロック１２０に対する置換処理が実行される。
この場合、図１４の下線で示すように、条件分岐ブロック１２２の判定結果が偽のときに実行されるsub1_tmp1[s4t_tmp]=sub1_tmp2[s4t_tmp]の前に、乗算ブロック１２０の処理結果であるout1[s4t_tmp]＊0.01がsub1_tmp2[s4t_tmp]に代入される。
【００６３】
Ｓ４４６の判定がＹｅｓである、つまり、すべての処理ブロックの置換処理が終了した場合、Ｓ４４８においてサブコード生成部７４は、分割された複数のサブシステムについて、置換処理が終了したか否かを判定する。Ｓ４４８の判定がＮｏである、つまり置換処理が未処理のサブシステムが存在する場合、処理はＳ４３２に移行する。

【００６４】
例えば、図７においてモデル１００を分割したサブシステム１３０、４０のうち、サブシステム１３０の置換処理が終了すると、サブシステム１３０と同様にサブシステム１４０の置換処理が実行される。サブシステム１４０の置換処理では、サブシステム１３０の置換処理で定義された入出力パラメータが使用される。
【００６５】
そして、Ｓ４４８の判定がＹｅｓである、つまり、すべてのサブシステムの置換処理が終了すると、Ｓ４５０においてソースコード生成部７６は、図１５に示すように、生成されたサブソースコード２１２、２１４をまとめて、モデル１００に対応するソースコード２１０を生成する。
【００６６】
図１５において、符号２１２がサブシステム１３０から生成されたサブソースコードであり、符号２１４がサブシステム１４０から生成されたサブソースコードである。
サブシステム１４０に対応するソースコード２１４の部分において、条件分岐ブロック１２６の判定であるif (in4s4t_tmp] != 0.0F)の判定結果が偽の場合、out1[s4t_tmp]には同じ値が代入されるので、分岐先の記述は追加されていない。
【００６７】
図１５に示すように、モデル１００から分割されたサブシステム１３０、１４０において、元のソースコード２００に対して、置換処理により生成されたサブソースコード２１２、２１４には関数定義は存在していない。そして、サブソースコード２１２、２１４は、反復と分岐と順接との処理だけで記述されている。
【００６８】
図３のＳ４１８で実行されるテストは、Ｓ４１２においてサブシステム１３０、１４０のそれぞれに対して生成されたテストケースで実行される。
［３．効果］
以上説明した上記実施形態では、以下の効果を得ることができる。
【００６９】
（１）元のソースコード２００に対して、置換処理により生成されたサブソースコード２１２、２１４には関数定義は存在しておらず、サブソースコード２１２、２１４は、反復と分岐と順接との処理だけで記述されている。したがって、サブソースコード２１２、２１４から生成されるソースコード２１０を実行するときに関数が呼び出されないので、ソースコード２１０の実行時間が増加することを抑制しつつ、サブシステム毎のテストケースを実行できる。
【００７０】
（２）モデル１００から分割されて、簡単化かつ小型化されたサブシステム１３０、１４０と、サブシステム１３０、１４０に対応して生成されたサブソースコード２１２、２１４との組み合わせに対して、それぞれのサブシステム１３０、１４０から生成されたテストケースによりテストを実行する。
【００７１】
サブシステム１３０、１４０はモデル１２よりも簡単化かつ小型化されているので、生成されたテストケースにより実行されたテストのカバレッジは向上する。
上記実施形態において、Ｓ４０６、Ｓ４１４がカバレッジ判定部の処理に対応し、Ｓ４３０、Ｓ４３２が対応付け部の処理に対応し、Ｓ４３４がパラメータ定義部の処理に対応し、Ｓ４３６～Ｓ４４８がサブコード生成部の処理に対応し、Ｓ４５０がソースコード生成部の処理に対応する。
［４．他の実施形態］
（１）上記実施形態では、置換処理前のモデルであるモデル１００を１回分割して生成されたサブシステム１３０、１４０のカバレッジが所定値以上になる例を説明した。
【００７２】
これに対し、分割して生成されたすべてのサブシステムのカバレッジが所定値以上にな

るまで、カバレッジが所定値未満のサブシステムをさらに分割して複数のサブシステムを生成してもよい。

【００７３】

（２）上記実施形態では、ソースコード生成装置５０のカバレッジ判定部６４が、モデル１００から取得する分割情報に基づいて、モデル１００またはサブシステム１３０、１４０を分割して複数のサブシステムを生成した。

【００７４】

これに対し、例えばオペレータが、ソースコード生成装置５０のマウスまたはキーボード等の入力装置を操作してモデル１００を分割してもよい。

（３）上記実施形態では、ソースコード生成装置５０のカバレッジ判定部６４が、モデル１００から取得する分割情報に基づいて、モデル１００またはサブシステム１３０、１４０を分割して複数のサブシステムを生成した。

【００７５】

これに対し、ソースコード生成装置５０が、モデル１００の構造を解析し、モデル１００の分割箇所を決定する分割部を備えてもよい。

（４）上記実施形態における一つの構成要素が有する複数の機能を複数の構成要素によって実現したり、一つの構成要素が有する一つの機能を複数の構成要素によって実現したりしてもよい。また、複数の構成要素が有する複数の機能を一つの構成要素によって実現したり、複数の構成要素によって実現される一つの機能を一つの構成要素によって実現したりしてもよい。また、上記実施形態の構成の一部を省略してもよい。また、上記実施形態の構成の少なくとも一部を、他の上記実施形態の構成に対して付加または置換してもよい。尚、特許請求の範囲に記載した文言のみによって特定される技術思想に含まれるあらゆる態様が本開示の実施形態である。

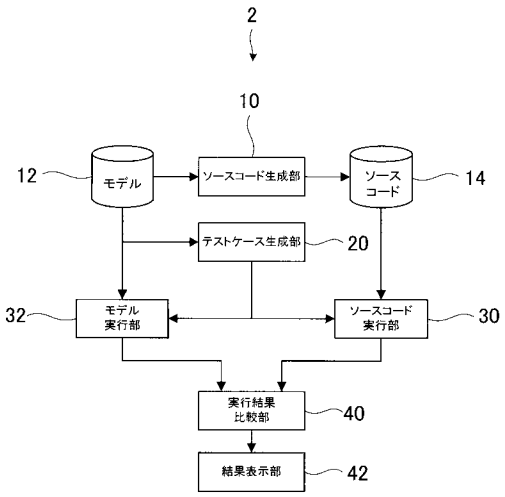
（５）上述したソースコード生成装置の他、当該ソースコード生成装置を構成要素とするシステム、当該ソースコード生成装置としてコンピュータを機能させるためのソースコード生成プログラム、このソースコード生成プログラムを記録した記録媒体、ソースコード生成方法など、種々の形態で本開示を実現することもできる。

【符号の説明】

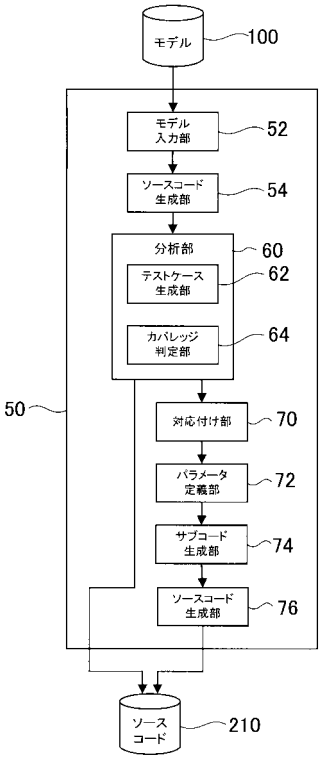
【００７６】

５０：ソースコード生成装置、６４カバレッジ判定部、７０：対応付け部、７２：パラメータ定義部、７４：サブコード生成部、７６：ソースコード生成部、１００：モデル、１１２：定数ブロック（処理ブロック）、１２０：乗算ブロック（処理ブロック）、１２２～１２６：条件分岐ブロック（処理ブロック）、１２８：遅延ブロック（処理ブロック）、１３０、１４０：サブシステム、２００：元のソースコード、２１０：ソースコード、２１２、２１４：サブソースコード

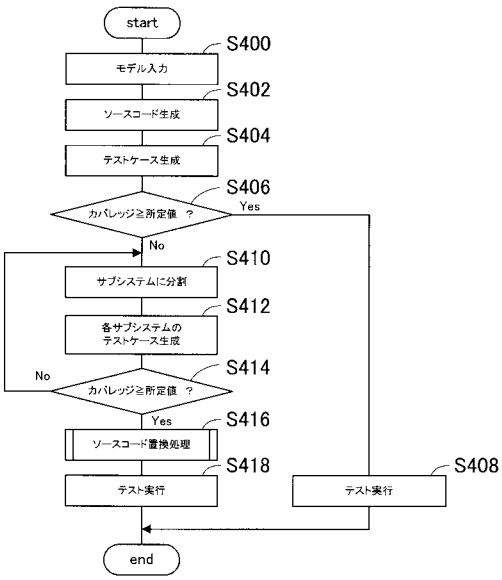
【図１】


2
10
12
モデル
ソースコード生成部
ソースコード
14
テストケース生成部
20
32
モデル実行部
ソースコード実行部
30
実行結果比較部
40
結果表示部
42

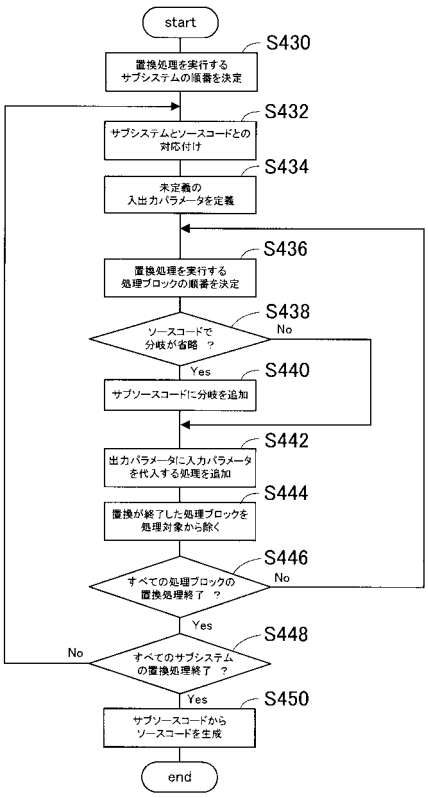
【図２】


モデル
100
モデル入力部
52
ソースコード生成部
54
分析部
60
テストケース生成部
62
カバレッジ判定部
64
50
対応付け部
70
パラメータ定義部
72
サブコード生成部
74
ソースコード生成部
76
ソースコード
210

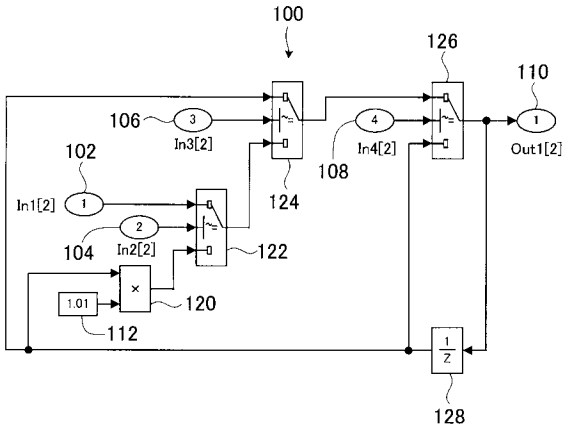
【図３】


start
S400
モデル入力
S402
ソースコード生成
S404
テストケース生成
S406
カバレッジ≧所定値 ？
Yes
No
S410
サブシステムに分割
S412
各サブシステムのテストケース生成
S414
No
カバレッジ≧所定値 ？
Yes
S416
ソースコード置換処理
S418
テスト実行
S408
テスト実行
end

【図４】


start
S430
置換処理を実行するサブシステムの順番を決定
S432
サブシステムとソースコードとの対応付け
S434
未定義の入出力パラメータを定義
S436
置換処理を実行する処理ブロックの順番を決定
S438
ソースコードで分岐が省略 ？
No
Yes
S440
サブソースコードに分岐を追加
S442
出力パラメータに入力パラメータを代入する処理を追加
S444
置換が終了した処理ブロックを処理対象から除く
S446
すべての処理ブロックの置換処理終了 ？
No
Yes
S448
No
すべてのサブシステムの置換処理終了 ？
Yes
S450
サブソースコードからソースコードを生成
end

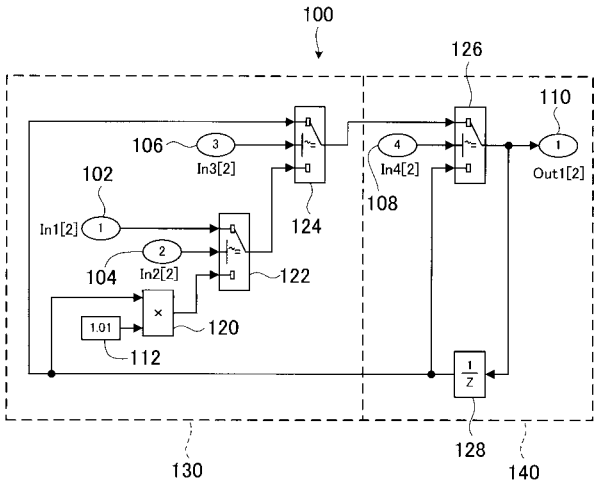
【図５】



【図６】

```
200
Void main (void)
{
    signed long s4t_tmp;
    for (s4t_tmp = 0; s4t_tmp < 2; s4t_tmp++)
    {
        if ((in4[s4t_tmp] != 0.0F) && (!(in3[s4t_tmp] != 0.0F)))
        {
          if (in2[s4t_tmp] != 0.0F)
          {
            out1[s4t_tmp] = in1[s4t_tmp];
          }
          else
          {
             out1[s4t_tmp] *= 1.01;
          }
        }
    }
}
```

【図７】



【図８】

サブシステムとソースコードの対応			
種別	名称	対応するソースコード	サブソースコード

入力パラメータ	a	in1[(s4t_tmp)]	signed long s4t_tmp;
			for (s4t_tmp = 0; s4t_tmp < 2; s4t_tmp++)
			{
			if(in3[s4t_tmp] != 0.0F)
			{
			…
			}
			}

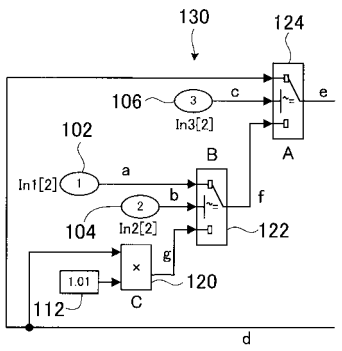
入力パラメータ	b	in2[(s4t_tmp)]	
入力パラメータ	c	in3[(s4t_tmp)]	
入力パラメータ	d	out1[(s4t_tmp)]	
出力パラメータ	e	未定義	
一時パラメータ	f	未定義	
一時パラメータ	g	未定義	

反復／分岐	A	if (!(in3[s4t_tmp] != 0.0F))
		{
		…
		}	

反復／分岐	B	if (in2[s4t_tmp] != 0.0F)
		{
		out1[s4t_tmp] =in1[s4t_tmp];
		}
		else
		{
		out1[s4t_tmp] *= 1.01;
		}	

反復／順接	C	out1[s4t_tmp] *= 1.01;	

【図９】



【図１０】

サブシステムとソースコードの対応

種別	名称	対応するソースコード	処理後のサブソースコード

入力パラメータ	a	in1[(s4t_tmp)]	double sub1_out1[2];
			double sub1_tmp1[2];
			double sub1_tmp2[2];
			signed long s4t_tmp;
			for (s4t_tmp = 0; s4t_tmp < 2; s4t_tmp++){
			if(in3[s4t_tmp] != 0.0F)
			{
			・・・
			}
			}

入力パラメータ	b	in2[(s4t_tmp)]	
入力パラメータ	c	in3[(s4t_tmp)]	
入力パラメータ	d	out1[(s4t_tmp)]	
出力パラメータ	e	double sub1_out1[2]	
一時パラメータ	f	double sub1_tmp1[2]	
一時パラメータ	g	double sub1_tmp2[2]	

反復／分岐	A	if (!(in3[s4t_tmp] != 0.0F))
		{
		・・・
		}	

反復／分岐	B	if (in2[s4t_tmp] != 0.0F)
		{
		out1[s4t_tmp] =in1[s4t_tmp];
		}
		else
		{
		out1[s4t_tmp] *= 1.01;
		}	

反復／順接	C	out1[s4t_tmp] *= 1.01;	

【図１１】

サブシステムとソースコードの対応

種別	名称	対応するソースコード	処理後のサブソースコード

入力パラメータ	a	in1[(s4t_tmp)]	double sub1_out1[2];
			double sub1_tmp1[2];
			double sub1_tmp2[2];
			signed long s4t_tmp;
			for (s4t_tmp = 0; s4t_tmp < 2; s4t_tmp++){
			if(in3[s4t_tmp] != 0.0F){
			・・・
			}else{
			・・・
			}
			}

入力パラメータ	b	in2[(s4t_tmp)]	
入力パラメータ	c	in3[(s4t_tmp)]	
入力パラメータ	d	out1[(s4t_tmp)]	
出力パラメータ	e	double sub1_out1[2]	
一時パラメータ	f	double sub1_tmp1[2]	
一時パラメータ	g	double sub1_tmp2[2]	

反復／分岐	A	if (!(in3[s4t_tmp] != 0.0F))
		{
		・・・
		}	

反復／分岐	B	if (in2[s4t_tmp] != 0.0F)
		{
		out1[s4t_tmp] =in1[s4t_tmp];
		}
		else
		{
		out1[s4t_tmp] *= 1.01;
		}	

反復／順接	C	out1[s4t_tmp] *= 1.01;	

【図１２】

サブシステムとソースコードの対応

種別	名称	対応するソースコード	処理後のサブソースコード

入力パラメータ	a	in1[(s4t_tmp)]	double sub1_out1[2];
			double sub1_tmp1[2];
			double sub1_tmp2[2];
			signed long s4t_tmp;
			for (s4t_tmp = 0; s4t_tmp < 2; s4t_tmp++){
			if(in3[s4t_tmp] != 0.0F){
			sub1_out1[s4t_tmp]=out1[s4t_tmp];
			}else{
			sub1_out1[s4t_tmp]=sub1_tmp1[s4t_tmp];
			}
			}

入力パラメータ	b	in2[(s4t_tmp)]	
入力パラメータ	c	in3[(s4t_tmp)]	
入力パラメータ	d	out1[(s4t_tmp)]	
出力パラメータ	e	double sub1_out1[2]	
一時パラメータ	f	double sub1_tmp1[2]	
一時パラメータ	g	double sub1_tmp2[2]	
反復／分岐	A	処理完了	

反復／分岐	B	if (in2[s4t_tmp] != 0.0F)
		{
		out1[s4t_tmp] =in1[s4t_tmp];
		}
		else
		{
		out1[s4t_tmp] *= 1.01;
		}	

反復／順接	C	out1[s4t_tmp] *= 1.01;	

【図１３】

サブシステムとソースコードの対応

種別	名称	対応するソースコード	処理後のサブソースコード

入力パラメータ	a	in1[(s4t_tmp)]	double sub1_out1[2];
			double sub1_tmp1[2];
			double sub1_tmp2[2];
			signed long s4t_tmp;
			for (s4t_tmp = 0; s4t_tmp < 2; s4t_tmp++){
			if(in3[s4t_tmp] != 0.0F){
			sub1_out1[s4t_tmp]=out1[s4t_tmp];
			}else{
			if (in2[s4t_tmp] != 0.0F){
			sub1_tmp1[s4t_tmp]=in1[s4t_tmp];
			}else{
			sub1_tmp1[s4t_tmp]=sub1_tmp2[s4t_tmp];
			}
			sub1_out1[s4t_tmp]=sub1_tmp1[s4t_tmp];
			}
			}

入力パラメータ	b	in2[(s4t_tmp)]	
入力パラメータ	c	in3[(s4t_tmp)]	
入力パラメータ	d	out1[(s4t_tmp)]	
出力パラメータ	e	double sub1_out1[2]	
一時パラメータ	f	double sub1_tmp1[2]	
一時パラメータ	g	double sub1_tmp2[2]	
反復／分岐	A	処理完了	

反復／分岐	B	if (in2[s4t_tmp] != 0.0F)
		{
		out1[s4t_tmp] =in1[s4t_tmp];
		}
		else
		{
		out1[s4t_tmp] *= 1.01;
		}	

反復／順接	C	out1[s4t_tmp] *= 1.01;	

【図１４】

サブシステムとソースコードの対応

種別	名称	対応するソースコード	処理後のサブソースコード

入力パラメータ	a	in1[(s4t_tmp)]	double sub1_out1[2];
			double sub1_tmp1[2];
			double sub1_tmp2[2];
			signed long s4t_tmp;
			for (s4t_tmp = 0; s4t_tmp < 2; s4t_tmp++){
			if(in3[s4t_tmp] != 0.0F){
			sub1_out1[s4t_tmp]=out1[s4t_tmp];
			}else{
			if (in2[s4t_tmp] != 0.0F){
			sub1_tmp1[s4t_tmp]=in1[s4t_tmp];
			}else{
			sub1_tmp2[s4t_tmp]=out1[s4t_tmp]*0.01;
			sub1_tmp1[s4t_tmp]=sub1_tmp2[s4t_tmp];
			}
			sub1_out1[s4t_tmp]=sub1_tmp1[s4t_tmp];
			}
			}

入力パラメータ	b	in2[(s4t_tmp)]	
入力パラメータ	c	in3[(s4t_tmp)]	
入力パラメータ	d	out1[(s4t_tmp)]	
出力パラメータ	e	double sub1_out1[2]	
一時パラメータ	f	double sub1_tmp1[2]	
一時パラメータ	g	double sub1_tmp2[2]	
反復／分岐	A	処理完了	
反復／分岐	B	処理完了	
反復／順接	C	処理完了	

【図１５】