(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2012-181782
(P2012-181782A)
(43)公開日　平成24年9月20日(2012.9.20)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 11/28	(2006.01)	G06F 11/28	340A	5B042

審査請求 有　請求項の数 9　ＯＬ　（全 22 頁）

(21)出願番号　特願2011-45688(P2011-45688)
(22)出願日　平成23年3月2日(2011.3.2)

（特許庁注：以下のものは登録商標）
１．ＪＡＶＡ

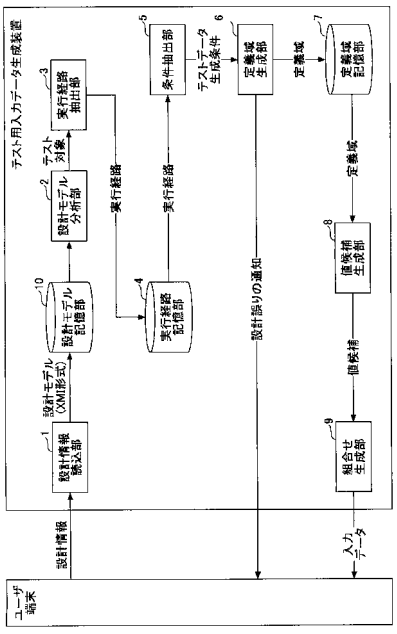
(71)出願人　000004226
日本電信電話株式会社
東京都千代田区大手町二丁目３番１号
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100124844
弁理士　石原　隆治
(72)発明者　張　暁晶
東京都千代田区大手町二丁目３番１号　日本電信電話株式会社内
(72)発明者　丹野　治門
東京都千代田区大手町二丁目３番１号　日本電信電話株式会社内

最終頁に続く

(54)【発明の名称】テスト用入力データ生成装置及び方法及びプログラム

(57)【要約】
【課題】　複数の条件を同時に満たした所望する文字列型の入力データの生成、および、設計誤りの検出を可能とする。
【解決手段】　本発明は、ユーザ端末より、設計書で記述した設計情報を読み込み、内部形式である設計モデルに変換し、設計モデルから画面と処理間のインタラクションをテスト対象として抽出し、テスト対象から実行経路を抽出し、実行経路記憶手段に格納し、実行経路記憶手段から各実行経路を読み込んで、該各実行経路の始点から終点までを走査することにより入力データが満たすべきテストデータ生成条件を抽出する。テストデータ生成条件を変数毎に整理し、入力データを構成する値候補を生成するための定義域を作成し、定義域記憶手段に格納し、定義域記憶手段から定義域を読み込み、該定義域が表す範囲の中から値候補を生成し、ユーザ端末に出力する。
【選択図】　図３

本発明の一実施の形態における装置構成図



【特許請求の範囲】
【請求項１】
特定のテストケースを実施する際の、確認したいシステムの振る舞いを引き起こせる、テスト用入力データを生成するテスト用入力データ生成装置であって、
ユーザ端末より、設計書で記述した設計情報を読み込み内部形式である設計モデルに変換する設計情報読込手段と、
前記設計モデルから画面と処理間のインタラクションをテスト対象として抽出する設計モデル分析手段と、
前記テスト対象から実行経路を抽出し、実行経路記憶手段に格納する実行経路抽出手段と、
前記実行経路記憶手段から各実行経路を読み込んで、該各実行経路の始点から終点までを走査することにより入力データが満たすべきテストデータ生成条件を抽出する条件抽出手段と、
前記テストデータ生成条件を変数毎に整理し、前記入力データを構成する値候補を生成するための定義域を作成し、定義域記憶手段に格納する定義域生成手段と、
前記定義域記憶手段から前記定義域を読み込み、該定義域が表す範囲の中から値候補を生成する値候補生成手段と、
複数の変数のそれぞれに前記値候補を割り当てて組み合わせを生成し、入力データとして前記ユーザ端末に出力する組み合わせ生成手段と、
を有することを特徴とするテスト用入力データ生成装置。
【請求項２】
前記定義域生成手段は、
前記定義域記憶手段の定義域に、前記テストデータ生成条件を順次格納していき、
前記条件抽出手段で抽出されたテストデータ生成条件に含まれる各条件と、既に該定義域記憶手段に格納されているテストデータ生成条件の各条件との相互矛盾を検出するチェック手段と、
前記チェック手段において、前記条件抽出手段で抽出されたテストデータ生成条件に含まれる各条件と矛盾があると判断された場合には、設計誤りとして前記ユーザ端末に通知し、矛盾がないと判断された場合には、前記定義域に前記条件を追加する通知手段と、
を含む請求項１記載のテスト用入力データ生成装置。
【請求項３】
前記定義域記憶手段は、
文字列の長さ、文字種、包含文字列の各属性をもつ文字列型の定義域と、数値型の定義域を有し、
前記定義域生成手段は、
前記テストデータ生成条件の変数のデータ型が文字列型である場合は、前記条件抽出手段で抽出されたテストデータ生成条件の変数毎に条件を振り分け、各条件の文字列の長さ、文字種、包含文字列の各属性を識別し、該属性毎に前記文字列型の定義域に該条件を格納し、該データ型が数値型の場合は、該数値型の定義域に該条件を格納する条件格納手段を含み、
前記条件格納手段は、
前記包含文字列について、文字列の長さの条件を新規に生成し、前記定義域に格納し、全ての条件の保存終了後に、前記文字列の長さに属する複数の条件をマージし、前記文字種に属する複数の条件をマージする手段を含む
請求項１または２記載のテスト用データ生成装置。
【請求項４】
前記チェック手段は、
前記定義域記憶手段の前記文字列型の定義域に格納されている、文字列型の入力データ生成における、異なる属性の条件間の相互矛盾、および、同じ属性の相反する条件の存在をチェックする手段を含み、

前記異なる属性の条件間の相互矛盾として、包含文字列と文字列の長さの間の相互矛盾及び包含文字列と文字種の間の相互矛盾を検証し、
前記同じ属性の相反する条件の存在として、複数の文字種の指定、及び、複数の文字列の長さの指定、及び、複数の包含文字列の指定を検証することにより、相互矛盾を検出する手段を含む
請求項２または３記載のテスト用データ生成装置。
【請求項５】
特定のテストケースを実施する際の、確認したいシステムの振る舞いを引き起こせる、テスト用入力データを生成するテスト用入力データ生成方法であって、
設計情報読込手段が、ユーザ端末より、設計書で記述した設計情報を読み込み内部形式である設計モデルに変換する設計情報読込ステップと、
設計モデル分析手段が、前記設計モデルから画面と処理間のインタラクションをテスト対象として抽出する設計モデル分析ステップと、
実行経路抽出手段が、前記テスト対象から実行経路を抽出し、実行経路記憶手段に格納する実行経路抽出ステップと、
条件抽出手段が、前記実行経路記憶手段から各実行経路を読み込んで、該各実行経路の始点から終点までを走査することにより入力データが満たすべきテストデータ生成条件を抽出する条件抽出ステップと、
定義域生成手段が、前記テストデータ生成条件を変数毎に整理し、前記入力データを構成する値候補を生成するための定義域を作成し、定義域記憶手段に格納する定義域生成ステップと、
値候補生成手段が、前記定義域記憶手段から前記定義域を読み込み、該定義域が表す範囲の中から値候補を生成する値候補生成ステップと、
組み合わせ生成手段が、複数の変数のそれぞれに前記値候補を割り当てて組み合わせを生成し、入力データとして前記ユーザ端末に出力する組み合わせ生成ステップと、
を行うことを特徴とするテスト用入力データ生成方法。
【請求項６】
前記定義域生成ステップにおいて、
前記定義域記憶手段の定義域に、前記テストデータ生成条件を順次格納していき、
前記条件抽出ステップで抽出されたテストデータ生成条件に含まれる各条件が、既に該定義域記憶手段に格納されているテストデータ生成条件の各条件との相互矛盾を検出するチェックステップと、
前記チェックステップにおいて、前記条件抽出ステップで抽出されたテストデータ生成条件に含まれる各条件と矛盾があると判断された場合には、設計誤りとして前記ユーザ端末に通知し、矛盾がないと判断された場合には、前記定義域に前記条件を追加する通知ステップと、
を行う請求項５記載のテスト用入力データ生成方法。
【請求項７】
前記定義域生成ステップにおいて、
前記テストデータ生成条件の変数のデータ型が文字列型である場合は、前記条件抽出ステップで抽出されたテストデータ生成条件の変数毎に条件を振り分け、各条件の文字列の長さ、文字種、包含文字列の各属性を識別し、該属性毎に、前記定義域記憶手段上の文字列型の定義域に該条件を格納し、該データ型が数値型の場合は、該数値型の定義域に該条件を格納する条件格納ステップを行い、
前記条件格納ステップにおいて、
前記包含文字列について、文字列の長さの条件を新規に生成し、前記定義域に格納し、全ての条件の保存終了後に、前記文字列の長さに属する複数の条件をマージし、前記文字種に属する複数の条件をマージする
請求項５または６記載のテスト用データ生成方法。
【請求項８】

前記チェックステップにおいて、
前記定義域記憶手段の前記文字列型の定義域に格納されている、文字列型の入力データ生成における、異なる属性の条件間の相互矛盾、および、同じ属性の相反する条件の存在をチェックする条件チェックステップを行い、
前記条件チェックステップでは、
前記異なる属性の条件間の相互矛盾として、包含文字列と文字列の長さの間の相互矛盾及び包含文字列と文字種の間の相互矛盾を検証し、
前記同じ属性の相反する条件の存在として、複数の文字種の指定、及び、複数の文字列の長さの指定、及び、複数の包含文字列の指定を検証することにより、相互矛盾を検出する
請求項６または７記載のテスト用データ生成方法。
【請求項９】
コンピュータを、
請求項１乃至４記載のテスト用データ生成装置の各手段として機能させるためのテスト用データ生成プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、テスト用入力データ生成装置及び方法及びプログラムに係り、特に、システムの振る舞いが設計情報に記述された通りになっているか否かを確認するためのテストで用いる、テストケースに含まれる入力データを人手により作成するのではなく、設計情報により自動的に作成するためのテスト用入力データ生成装置及び方法及びプログラムに関する。
【０００２】
詳しくは、文字列型の入力データの作成、及び、設計情報に存在する設計誤りを検出するためのテスト用入力データ生成装置及び方法及びプログラムに関する。
【背景技術】
【０００３】
プログラム開発工程におけるテストでは、テスト対象システムの振る舞いが設計された通りになっているか、振る舞いのバリエーションをそれぞれ確認する。この個々のバリエーションを「テストケース」と呼ぶ。
【０００４】
テスト対象システムの特定の振る舞いは、設計情報中に実行経路として記述される。特定のテストケースを実施するためには、該当の振る舞いを引き起こせる、つまり、該当の実行経路を通ることができる、テストデータが必要になる。テストデータは入力データと期待結果からなる。人手に代わり、自動でテストデータを準備できれば、開発のコスト改善につながる。
【０００５】
また、テストは「設計通りか否か」の確認であるため、設計情報に誤りがある場合、テストの結果も信頼できないものとなる。テストケースやテストデータを作成する際には、設計誤りが無いという前提で行う必要がある。
【０００６】
テストデータの自動生成技術としては、設計情報に基づくテストデータ生成手法がある（例えば、特許文献１参照）。特許文献１の技術は、統一モデル言語のＵＭＬクラス図とアクティビティ図で記述した設計モデルから、実行経路を抽出し、特定の実行経路に対して、許容する正常な入力データ、および、許容しない異常な入力データを生成する。
【０００７】
また、プログラム動作履歴を入力とするテストデータ生成手法がある。この技術は、テスト対象のプログラムと変数が満たすべき制約条件から、プログラムの未検証領域に到達するためのテストデータを生成する。

【０００８】
　また、設計情報に基づくテストデータ生成手法がある（例えば、特許文献２参照）。特許文献２の技術は、変数のデータ型を二種類に分類し、異なるポリシーでテストデータを生成することで、トータルのテストデータの生成件数を軽減する。
【先行技術文献】
【特許文献】
【０００９】
【特許文献１】特開２０１０－２６７０２３号公報
【特許文献２】特許３１０５２７９号公報
【発明の概要】
【発明が解決しようとする課題】
【００１０】
　しかしながら、上記のテストデータ自動生成の技術は、以下のような問題がある。
【００１１】
　プログラム動作履歴を入力とするテストデータ生成手法では、条件に従って数値型の変数に対する値を決定できるのみで、システムでよく使われる文字列型のテストデータ生成については言及していない。
【００１２】
　特許文献２では、数値に加え、扱えるデータ型として1文字であるCHAR型を挙げているが、文字の羅列である文字列型のデータ生成では、1文字ずつを単純に並べるだけでは所望する性質を有するテストデータは得られない。
【００１３】
　特許文献１では、文字列型の入力データ生成ができるが、単一の条件を満たした入力データしか生成できない。文字列の長さ、文字種、部分文字列などの、複数の相互に関連する条件がある場合、特許文献１の手法を用いて逐次的に1つずつの条件を満たしていっても、所望する入力データは得られない。つまり複数条件を「同時に」満たした生成ができないことが問題である。
【００１４】
　また、設計情報を入力とする特許文献１と特許文献２の技術では、設計情報が正しいと仮定してテストデータを生成するため、設計誤りの検出ができない。
【００１５】
　本発明は、上記の点に鑑みなされたもので、複数の条件を同時に満たした所望する文字列型の入力データの生成、および、設計誤りの検出を可能とするテスト用入力データ生成装置及び方法及びプログラムを提供することを目的とする。
【課題を解決するための手段】
【００１６】
　上記の課題を解決するため、本発明は、特定のテストケースを実施する際の、確認したいシステムの振る舞いを引き起こせる、テスト用入力データを生成するテスト用入力データ生成装置であって、
　ユーザ端末より、設計書で記述した設計情報を読み込み内部形式である設計モデルに変換する設計情報読込手段と、
　前記設計モデルから画面と処理間のインタラクションをテスト対象として抽出する設計モデル分析手段と、
　前記テスト対象から実行経路を抽出し、実行経路記憶手段に格納する実行経路抽出手段と、
　前記実行経路記憶手段から各実行経路を読み込んで、該各実行経路の始点から終点までを走査することにより入力データが満たすべきテストデータ生成条件を抽出する条件抽出手段と、
　前記テストデータ生成条件を変数毎に整理し、前記入力データを構成する値候補を生成するための定義域を作成し、定義域記憶手段に格納する定義域生成手段と、

前記定義域記憶手段から前記定義域を読み込み、該定義域が表す範囲の中から値候補を生成する値候補生成手段と、
複数の変数のそれぞれに前記値候補を割り当てて組み合わせを生成し、入力データとして前記ユーザ端末に出力する組み合わせ生成手段と、
を有することを特徴とする。

【００１７】

また、本発明は、前記定義域生成手段において、
前記定義域記憶手段の定義域に、前記テストデータ生成条件を順次格納していき、
前記条件抽出手段で抽出されたテストデータ生成条件に含まれる各条件と、既に該定義域記憶手段に格納されているテストデータ生成条件の各条件との相互矛盾を検出するチェック手段と、
前記チェック手段において、前記条件抽出手段で抽出されたテストデータ生成条件に含まれる各条件と矛盾があると判断された場合には、設計誤りとして前記ユーザ端末に通知し、矛盾がないと判断された場合には、前記定義域に前記条件を追加する通知手段と、を含む。

【００１８】

また、本発明は、前記定義域記憶手段に、
文字列の長さ、文字種、包含文字列の各属性をもつ文字列型の定義域と、数値型の定義域を備え、
前記定義域生成手段において、
前記テストデータ生成条件の変数のデータ型が文字列型である場合は、前記条件抽出手段で抽出されたテストデータ生成条件の変数毎に条件を振り分け、各条件の文字列の長さ、文字種、包含文字列の各属性を識別し、該属性毎に前記文字列型の定義域に該条件を格納し、該データ型が数値型の場合は、該数値型の定義域に該条件を格納する条件格納手段を含み、
前記条件格納手段は、
前記包含文字列について、文字列の長さの条件を新規に生成し、前記定義域に格納し、全ての条件の保存終了後に、前記文字列の長さに属する複数の条件をマージし、前記文字種に属する複数の条件をマージする手段を含む。

【００１９】

また、本発明は、前記チェック手段において、
前記定義域記憶手段の前記文字列型の定義域に格納されている、文字列型の入力データ生成における、異なる属性の条件間の相互矛盾、および、同じ属性の相反する条件の存在をチェックする手段を含み、
前記異なる属性の条件間の相互矛盾として、包含文字列と文字列の長さの間の相互矛盾及び包含文字列と文字種の間の相互矛盾を検証し、
前記同じ属性の相反する条件の存在として、複数の文字種の指定、及び、複数の文字列の長さの指定、及び、複数の包含文字列の指定を検証することにより、相互矛盾を検出する手段を含む。

【発明の効果】

【００２０】

上記のように本発明によれば、相互に関連する複数の条件を同時に満たす、文字列型の入力データを生成する。すなわち文字列の長さ、文字種、包含文字列の３つの属性に分類できる複数の条件を同時に満たした、文字列型の入力データを生成することで、プログラム開発工程におけるテストでの入力データ作成をより効率的に行う。

【００２１】

また、本発明によれば、ソフトウェア設計情報に存在する設計誤りを検出する。すなわち、入力データ生成の過程での満たせない条件の検出により、通りえない実行経路が誤って設計されたことを検出できる。満たせない条件とは、文字列型の入力データ生成における、異なる属性の条件間の相互矛盾、および、同じ属性の相反する条件の存在を指し、以

下のタイプがある。異なる属性の条件間の相互矛盾としては、
・包含文字列と文字列の長さの間の相互矛盾；
・包含文字列と文字種の間の相互矛盾；
がある。
【００２２】
同じ属性の相反する条件の存在としては、
・複数の文字種の指定；
・複数の文字列の長さの指定
・複数の包含文字列の指定
を指す。包含文字列は、完全一致、前方部分一致、中間部分一致、後方部分一致を含む。文字種は、全角、半角、英数字、かな、を含む。
【００２３】
本発明は、文字列型の入力データを生成することでテストケースの作成を効率化し、設計誤りを検出することで設計の品質を向上させることが可能となる。
【図面の簡単な説明】
【００２４】
【図１】本発明の一実施の形態におけるテストデータ生成条件のデータ構造である。
【図２】本発明の一実施の形態における定義域記憶部の定義域のデータ構造である。
【図３】本発明の一実施の形態における装置構成図である。
【図４】本発明の一実施の形態におけるテスト用入力データ生成装置のフローチャートである。
【図５】本発明の一実施の形態における定義域生成部のフローチャートである。
【図６】本発明の一実施の形態における図５のＳ２３０の詳細なフローチャートである。
【図７】本発明の一実施の形態における図６のＳ２４０の詳細なフローチャートである。
【図８】本発明の一実施の形態における図７のＳ４２０の詳細なフローチャートである。
【図９】本発明の一実施の形態における図８のＳ５３０の詳細なフローチャートである。
【図１０】本発明の一実施の形態における図８のＳ５４０の詳細なフローチャートである。
【図１１】本発明の一実施の形態における図８のＳ５５０の詳細なフローチャートである。
【図１２】本発明の一実施の形態における図９のＳ６３０の詳細なフローチャートである。
【図１３】本発明の一実施の形態における図９のＳ６２０、図１０のＳ７２０の詳細なフローチャートである。
【図１４】本発明の一実施の形態における図９のＳ６１０の詳細なフローチャートである。
【図１５】本発明の一実施の形態における図９のＳ６４０、図１１のＳ８１０の詳細なフローチャートである。
【図１６】本発明の一実施の形態における図１０のＳ７１０の詳細なフローチャートである。
【図１７】本発明の一実施の形態における図５のＳ２５０の詳細なフローチャートである。
【図１８】本発明の一実施の形態における値候補生成部のフローチャートである。
【図１９】本発明の一実施の形態における図１８のＳ１５３０の詳細なフローチャートである。
【発明を実施するための形態】
【００２５】
以下図面と共に、本発明の実施の形態を説明する。
【００２６】
最初に、以下の実施の形態で用いる用語について説明する。

【００２７】
「テストケース」とは、テスト対象システムの特定の振る舞いが正しいかを確認するためのものである。テストケースは、実行経路とテストデータを含む。
【００２８】
「テストデータ」とは、テスト対象システムへ与える刺激である。該当するテストケースで確認したい振る舞いを引き起こせる、言い換えれば、実行経路を通るものでなければならない。テストデータは、入力データと期待結果を含む。
【００２９】
「入力データ」とは、個々の入力変数がとる値の組み合わせである。変数には、文字列型、数値型、などのデータ型が定義されている。
【００３０】
「テストデータ生成条件」とは、設計モデルから抽出された実行経路に含まれ、図１（ａ）に示すように、『ａ＞３』のような変数（ここではａ）の取り得る値に関する条件の羅列を指す。条件は変数の名前"ａ"、条件種別"＞"、定数"３"から成る。図１（ｂ）に示すように、「条件種別」には、不等号や文字列の長さ等通常使われる記法を用いる。
【００３１】
「定義域」とは、変数がとりうる値の「許容される範囲」を示すものである。定義域は、図２に示すように、変数名、属性の配列、（各属性の中に）条件、設計誤り検出時のフラグをもつ。文字列型の定義域は、属性として、文字列の長さ、文字種、包含文字列の３つをもつ。
【００３２】
「包含文字列」とは、特定の文字列を含む文字列であり、完全一致、前方部分一致、中間部分一致、後方部分一致を含む。
【００３３】
「文字種」とは、使用可能な文字の集合を表すものである。選択肢として指定可能な文字種の一覧、およびそれらの包含関係は図１（ｃ）に示すように木構造で保持される。リーフにある候補は、包含する文字の一覧をもつ（例：[半角数字]であれば[0,1,2,・・・9]をもつ）。
【００３４】
「設計情報」とは、システムをどのような構成で実現するかを示す仕様である。本発明では、以下の設計書一式を設計情報として扱う。
【００３５】
・画面遷移図：機能内における各画面間の遷移を示すドキュメントを指す；
・画面要素一覧：画面内の各構成要素について入出力可否や属性（テキスト、プルダウンリスト等）に関して定義を記述したもの；
・入力制約定義書：入力となる画面要素単体についての制約事項を記述したもの；
・処理イベント定義書：画面上で発生したイベント（主にユーザ操作）に起因する処理の呼び出しを記述したもの；
・処理概要フロー定義書：処理の振る舞いを明確にするためのフロー：
・「処理詳細定義書」処理が終了した後の振る舞いを記述したもの：
図３は、本発明の一実施の形態におけるテスト用入力データ生成装置の構成を示す。
【００３６】
同図に示すテスト用入力データ生成装置は、設計情報読込部１、設計モデル分析部２、実行経路抽出部３、実行経路記憶部４、条件抽出部５、定義域生成部６、定義域記憶部７、値候補生成部８、組合せ生成部９、設計モデル記憶部１０から構成される。
【００３７】
このうち、設計情報読込部１、設計モデル分析部２、実行経路抽出部３、条件抽出部５、定義域生成部６、値候補生成部８、組合せ生成部９、設計モデル記憶部１０は、ＣＰＵ上の機能であり、実行経路記憶部４、定義域記憶部７は、ハードディスク装置等の記憶媒体である。

【００３８】
ユーザ端末は、設計情報をテスト用入力データ生成装置の設計情報読込部１に与える。また、テスト用入力データ生成装置から入力データを取得し、場合によっては設計誤りの通知を受け取る。
【００３９】
設計情報読込部１は、ユーザ端末から以下の設計書一式が与えられる。
【００４０】
・画面遷移図：機能内における各画面間の遷移を示す；
・画面要素定義書：画面内の各構成要素について入出力可否や属性に関する定義が記載されている；
・入力制約定義書：入力となる画面要素単体についての制約事項；
・処理イベント定義書：画面上で発生したイベントに起因する処理の呼び出しを記載；
・処理概要フロー定義書：処理の振る舞いを明確にするための処理フローを記載；
・処理詳細定義書：処理終了後の振る舞いを記述；
上記の設計書一式を読み込み、単一の中間形式の言語に変換したオブジェクトを、単一の設計モデルに統合し、設計モデル記憶部１０に保持する。ここでは、画面遷移図、処理概要フロー定義書はMS Visio（登録商標）形式、画面要素定義書、入力制約定義書、処理イベント定義書、処理詳細定義書はMS Excel（登録商標）形式で入力されるものとする。
【００４１】
設計モデル分析部２は、画面と処理のインタラクションを単位に、設計モデル記憶部１０に格納されている設計モデルから該当するアクティビティをテスト対象として抽出する。
実行経路抽出部３は、アクティビティから実行経路を抽出する。
【００４２】
実行経路記憶部４は、抽出された実行経路を保持する。
【００４３】
条件抽出部５は、実行経路記憶部４より実行経路を取得し、経路の始端から終端までを走査することで経路上の条件を順に集め、テストデータ生成条件として保持する。
【００４４】
定義域生成部６は、テストデータ生成条件を読み込み、変数毎に条件を分類した後、変数毎にその変数のデータ型に合った定義域を生成し、該当する変数に関する条件を1つずつ定義域に追加していき、矛盾を検出した時点で定義域に設計誤りフラグを立てて定義域記憶部７に保存し、残りの条件をスキップする。矛盾なくすべての条件の追加が終了したら定義域を定義域記憶部７に保存する。図２に示すようなフォーマットで定義域記憶部７に保存する。
【００４５】
定義域記憶部７は、図２に示すように、文字列型の変数の定義域、数値型の変数の定義域の２種類の定義域を保持する。
【００４６】
値候補生成部８は、定義域記憶部７より定義域を読み込み、定義域が表す範囲の中から境界値分析により値候補を複数生成する。
【００４７】
組合せ生成部９は、変数のそれぞれに、その変数に対し生成された値候補から1つ選び割り当てることで組合せを生成し、これを入力データとしてユーザ端末に出力する。ここでは、入力データはMS Excel形式で出力するものとする。
【００４８】
以下に、上記の構成における動作を説明する。
【００４９】
図４は、本発明の一実施の形態におけるテスト用入力データ生成装置の動作のフローチャートである。

【００５０】
ステップ１００）　設計情報読込部１は、ユーザ端末から与えられる一式の設計書を読み込み、単一の中間言語（Java Object）に変換し、単一の設計モデル（XMI形式UML）に変換し、設計モデル記憶部１０に格納する。
【００５１】
ステップ１１０）　設計モデル分析部２は、設計モデル記憶部１０から取得した設計モデルを分解してアクティビティをテスト対象として抽出し、実行経路抽出部３に渡す。
【００５２】
ステップ１２０）　実行経路抽出部３は、取得した各テスト対象（アクティビティ）の開始ノードから終了ノードまで走査して実行経路を抽出し、全ての実行経路を実行経路記憶部４に渡す。実行経路は分岐条件、事後条件を含む。具体的な実行経路の抽出方法としては、前述の特許文献１の技術を適用することが可能である。
【００５３】
ステップ１３０）　条件生成部５は、実行経路記憶部４から実行経路を読み出し、各実行経路から該当実行経路を通るために満たすべき条件（分岐条件）を逐一集めてテストデータ生成条件として定義域生成部６に渡す。本ステップのテストデータの生成条件の集約方法としては、前述の特許文献１の技術を用いることができる。
【００５４】
ステップ１４０）　定義域生成部６は、テストデータ生成条件を構成する個々の条件を変数毎に振り分け、条件の属性（文字列の長さ、文字種、包含文字列）を識別し、変数毎の定義域にこれら条件を保存し、定義域記憶部７に保存する。条件を定義域に保存する際に条件間の矛盾が存在すれば、設計誤りを検出したとして定義域記憶部７の定義域（図２の「設計誤りがあるか」の欄）にフラグを立てて残りの条件をスキップする。
【００５５】
以下に、定義域生成部６の動作を説明する。図５は、本発明の一実施の形態における定義域生成部のフローチャートであり、上記のステップ１４０（変数毎に定義域を生成し記憶部に保存）の詳細な動作を示す。
【００５６】
ステップ２００）　定義域生成部６は、条件生成部５からテストデータ生成条件を取得し、メモリ（図示せず）に格納する。
【００５７】
ステップ２１０）　メモリ（図示せず）から条件を取得して、条件から変数名を抽出する。
【００５８】
ステップ２２０）　変数名毎にソートした条件を、メモリ（図示せず）に格納する。
【００５９】
ステップ２３０）　変数のデータ型を取得し、データ型に応じた定義域を生成し、メモリ（図示せず）に格納する。当該処理の詳細については、図６に沿って説明する。図６は、本発明の一実施の形態におけるステップ２３０の詳細なフローチャートである。
【００６０】
ステップ３００）　条件から対象とする変数を特定する。変数のデータ型が数値型、または、文字列型のいずれであるかを判定し、数値型の場合はステップ３１０に、文字列型の場合はステップ３３０に移行する。
【００６１】
ステップ３１０）　変数のデータ型が数値型である場合、空白の数値型の定義域を生成しメモリ（図示せず）に格納する。
【００６２】
ステップ３２０）　数値型の定義域に該当変数の変数名を設定する。
【００６３】
ステップ３３０）　変数のデータ型が文字列型である場合、空白の文字列型の定義域を

生成しメモリ（図示せず）に格納する。

【００６４】

ステップ３４０） 文字列型の定義域に該当変数の変数名を設定する。

【００６５】

ステップ２４０） 定義域生成部６は、メモリ（図示せず）からステップ２３０で格納された数値型、文字列型の条件を取得して、条件の属性に応じて定義域記憶部７の定義域の該当位置に保存する。当該処理の詳細については、図７に沿って説明する。図７は、本発明の一実施の形態におけるステップ２４０（定義域に保存）の詳細なフローチャートである。

【００６６】

ステップ４００） メモリ（図示せず）から条件と定義域を取得し、定義域のデータ型を文字列型、数値型のいずれであるかを判定する。

【００６７】

ステップ４１０） 定義域に変数名があれば、数値型として条件を定義域記憶部７の数値型の定義域（図２（ｂ））へ保存する。

【００６８】

ステップ４２０） 定義域が空白であれば、文字列型として条件を定義域記憶部７の文字列型の定義域（図２（ａ））へ保存する。当該処理の詳細については、図８に沿って説明する。図８は、本発明の一実施の形態におけるステップ４２０の詳細なフローチャートである。

【００６９】

ステップ５００） メモリ（図示せず）から条件を取り込む。

【００７０】

ステップ５０５） 条件をパースし条件種別を抽出。

【００７１】

ステップ５１０） 属性毎の許容される条件種別の一覧を読み込む。

【００７２】

ステップ５１５） 属性毎の許容される条件種別の一覧に載っていない、解読できない条件種別があれば、誤って記述されたと判断し、定義域に設計誤りのフラグを付与する。

【００７３】

ステップ５２０） 条件種別が包含文字列（例えば、［X Contains"あいうえお"］）の条件ならば、定義域の包含文字列属性に条件を追加する。

【００７４】

ステップ５３５） 条件種別が文字種の条件（例えば、［X UseSet"半角数字"］）ならば、定義域の文字種属性に条件を追加する。

【００７５】

ステップ５４５） 条件種別が文字列の長さの条件（例えば、［X Length>=10］）であるような条件ならば、定義域の文字列の長さ属性に条件を追加する。

【００７６】

ステップ５３０） 包含文字列の条件追加による矛盾検出を行う。当該処理の詳細については、図９に沿って後述する。

【００７７】

ステップ５４０） 文字種の条件追加による矛盾検出を行う。当該処理の詳細については、図１０に沿って後述する。

【００７８】

ステップ５５０） 文字列の長さの条件追加による矛盾検出を行う。当該処理の詳細については、図１１に沿って後述する。

【００７９】

以降、条件追加時の矛盾検出について図９、図１０、図１１で説明する。

【００８０】

図９は、本発明の一実施の形態におけるステップ５３０の詳細なフローチャートである。

【００８１】

ステップ６００）　包含文字列の条件と定義域を取得する。

【００８２】

ステップ６１０）　包含文字列の条件同士の矛盾検出を行う。当該処理の詳細については、図１４に沿って後述する。

【００８３】

ステップ６２０）　包含文字列と文字種の矛盾検出を行う。当該処理の詳細については、図１３に沿って後述する。

【００８４】

ステップ６３０）　包含文字列の条件から暗黙的に読み取れる文字列の長さ属性の条件（最小値）を生成し追加する。当該処理の詳細については、図１３に沿って後述する。

【００８５】

ステップ６４０）　文字列の長さの条件同士の矛盾検出を行う。当該処理の詳細については、図１２に沿って後述する。

【００８６】

図１０は、本発明の一実施の形態におけるステップ５４０の詳細なフローチャートである。

【００８７】

ステップ７００）　文字種の条件と定義域を取得する。

【００８８】

ステップ７１０）　文字種の条件同士の矛盾検出を行う。当該処理の詳細については、図１３に沿って後述する。

【００８９】

ステップ７２０）　包含文字列と文字種の矛盾検出を行う。当該処理の詳細については、図１０に沿って後述する。

【００９０】

図１１は、本発明の一実施の形態におけるステップ５５０の詳細なフローチャートである。

【００９１】

ステップ８００）　文字列の長さの条件と定義域を取得する。

【００９２】

ステップ８１０）　文字列の長さの条件同士の矛盾検出を行う。当該処理の詳細については、図１２に沿って後述する。

【００９３】

以降、矛盾検出の仕方について図１２、図１３、図１４で説明する。

【００９４】

図１２は、本発明の一実施の形態におけるステップ６３０の詳細なフローチャートである。

【００９５】

ステップ９００）　包含文字列属性の条件をすべて取得する。

【００９６】

ステップ９１０）　Containsで指定された文字列に重なりがなければ、文字列の文字数をカウントし、すべて足し合わせた数をmaxとする。

【００９７】

ステップ９２０）　Containsで指定された文字列に重なりがあれば、他と重ならない部分文字列の文字数をカウントし、足し合わせた数をmaxとする。

【００９８】

ステップ９３０）　条件 [変数 Length>= max] を生成し、定義域記憶部７の定義域の

文字列の長さ属性に追加する。

【００９９】

図１３は、本発明の一実施の形態におけるステップ６２０、ステップ７２０に共通する（包含文字列と文字種の矛盾検出）詳細なフローチャートである。

【０１００】

ステップ１０００）　包含文字列属性の条件、文字種属性の条件をすべて定義域から取得する。

【０１０１】

ステップ１０１０）　文字種属性の条件はあるかを判定する。なければ終了する。

【０１０２】

ステップ１０２０）　もし文字種属性の条件があれば、包含文字列Containsで指定された文字列群Cを取得する。

【０１０３】

ステップ１０３０）　Cに含まれる各文字列をそれぞれ、1文字ずつに分割し、メモリ（図示せず）内の文字種指定の文字一覧に含まれるか照合する。なお、文字種指定の文字一覧とは、数字なら"０，１，…，９"、ひらがなであれば、"あ～を"、アルファベットであれば"Ａ～Ｚ"などである。

【０１０４】

ステップ１０４０）　含まれない文字があるかを判定し、もしなければ終了する。

【０１０５】

ステップ１０５０）　含まれない文字があれば、矛盾を検出したとして、定義域に設計誤りのフラグを付与する。検出される矛盾の例を示す。条件[X Contains "123あ45"]と条件[X UseSet "半角数字"]が共に定義域に存在すれば、文字"あ"は半角数字に含まれないため設計誤りとなる。

【０１０６】

図１４は、本発明の一実施の形態におけるステップ６１０（包含文字列の条件同士の矛盾検出）の詳細なフローチャートである。

【０１０７】

ステップ１１００）定義域記憶部７の定義域から包含文字列属性の条件をすべて取得する。

【０１０８】

ステップ１１１０）　条件種別がNotContains（包含文字列ではない）の条件はあるか判定する。なければ終了する。

【０１０９】

ステップ１１２０）　NotContainsで指定された文字列群Nを取得する。

【０１１０】

ステップ１１３０）　条件種別がContains（包含文字列）である条件はあるかを判断し、なければ終了する。

【０１１１】

ステップ１１４０）　Containsで指定された文字列群Cを取得する。

【０１１２】

ステップ１１５０）　文字列群Cの要素CiがNの要素Niを部分文字列として包含するか、つまり、Ni⊂Ciが成立するようなNiとCiのペアが存在するかを逐次判定する。存在しなければ終了する。

【０１１３】

ステップ１１６０）　存在すれば、矛盾を検出したとして、定義域記憶部７の定義域に設計誤りのフラグを付与する。検出される矛盾の例を示す。条件[X NotContains "あい"]と条件[X Contains"あいうえお"]が共に定義域に存在すれば、"あいうえお"は"あい"を包含するため設計誤りとなる。

【０１１４】

図１５は、本発明の一実施の形態におけるステップ６４０、ステップ８１０（文字列の長さの条件同士の矛盾検出）に共通する詳細なフローチャートである。

【０１１５】

ステップ１２００）　文字列の長さの条件をすべて取得する。

【０１１６】

ステップ１２１０）　条件を一次連立不等式として見なし、解の存在を判定する。

【０１１７】

ステップ１２２０）　解が存在するか。存在すれば終了する。

【０１１８】

ステップ１２３０）　解が存在しなければ、矛盾を検出したとして、定義域に設計誤りのフラグを付与する。検出される矛盾の例を示す。条件[X Length<= 6]と条件[X Length>= 4]と条件[X Length!= 5]が共に定義域に存在すれば、連立不等式の解がない（重なる範囲が無い）ため設計誤りとなる。

【０１１９】

図１６は、本発明の一実施の形態におけるステップ７１０（文字種の条件同士の矛盾検出）の詳細なフローチャートである。

【０１２０】

ステップ１３００）　文字種属性の条件をすべて取得する。

【０１２１】

ステップ１３１０）　条件種別がUseSetである条件で指定された文字種候補をすべて集め、図１（ｃ）に示す文字種候補の一覧を照らし合わせて包含関係にあるかを判定する。包含関係であれば終了する。判定は木構造における親子関係の走査で行う。例えば、全角と全角かなは包含関係にあるが、全角と半角数字は包含関係にない。

【０１２２】

ステップ１３２０）　包含関係にない文字種候補のペアがあれば、矛盾を検出したとして、定義域に設計誤りのフラグを付与する。検出される矛盾の例を示す。条件[X UseSet "全角"]と条件[X UseSet "半角英数字"]が共に定義域に存在すれば、重なりが無いため設計誤りとなる。

【０１２３】

以上の処理により、定義域への条件の保存が完了した。引き続き図５の説明に戻る。

【０１２４】

ステップ２５０）　定義域に設計誤りのフラグがついているかを判定し、フラグがついていなければ定義域を単純化し、フラグがついていれば終了する。当該処理の詳細については、図１７に沿って説明する。図１７は、本発明の一実施の形態におけるステップ２５０（定義域を単純化）の詳細なフローチャートである。

【０１２５】

ステップ１４００）　文字列の長さの条件をすべて取得する。

【０１２６】

ステップ１４１０）　文字列の長さの条件を、２つの条件（最大値と最小値）にまで単純化する。例えば、

・条件[X Length>= 5]；
・条件[X Length< 20]；
・条件[X Length> 3]；
・条件[X Length<= 10]；

が存在する場合、最大値を示す条件[X Length<= 10]と最小値を示す条件[X Length>= 5]だけを残して他の条件を定義域から削除する。

【０１２７】

ステップ１４２０）　文字種の条件をすべて取得する。

【０１２８】

ステップ１４３０）　文字種の条件を、1つの条件（包含関係における最下位の文字種

候補）にまで単純化する。例えば、
・条件[X UseSet "半角数字"]；
・条件[X UseSet "半角英数字"]；
・条件[X UseSet "半角"]；
が存在する場合、半角⊃半角英数字⊃半角数字であるため、条件[X UseSet"半角数字"]だけを残して他の条件を定義域から削除する。

【０１２９】
当該図１７に示す処理を行なうことで複数の条件を構造化、一元化し、単一の定義域が生成されるため、変数のとりうる値の範囲が明確になり、後述する値候補生成部８において全ての条件を同時に満たした値を生成が可能となる。

【０１３０】
以上の処理により、これにより冗長な情報が削除され、定義域が完全に生成された。

【０１３１】
引き続き、図４の説明に戻る。

【０１３２】
ステップ１５０）　値候補生成部８は、定義域を取得し設計誤りフラグがついているか判定し、ついていなければ値候補を生成する。

【０１３３】
ステップ１６０）　値候補生成部８は、定義域に設計誤りフラグがついていれば該当定義域を破棄し、ユーザ端末へ設計誤りの存在を通知する。

【０１３４】
以下に、値候補生成部８の動作を説明する。図１８は、本発明の一実施の形態における値候補生成部のフローチャートであり、図４のＳ１５０（値候補生成）の詳細なフローを示す。

【０１３５】
ステップ１５００）　値候補生成部８は、定義域記憶部７の定義域を取得する。

【０１３６】
ステップ１５１０）　定義域のデータ型を文字列型、数値型のいずれであるかを判定する。

【０１３７】
ステップ１５２０）　定義域のデータ型が数値型である場合は数値型の値候補を生成する。

【０１３８】
ステップ１５３０）　定義域のデータ型が文字列型である場合は、文字列型の値候補を生成する。当該処理の詳細については、図１９に沿って後述する。

【０１３９】
図１９は、本発明の一実施の形態におけるステップ１５３０（文字列型の値候補の生成）の詳細なフローチャートである。

【０１４０】
ステップ１６００）　値候補生成部８は、定義域記憶部７から定義域を取得する。

【０１４１】
ステップ１６１０）　文字列の長さの条件をもとに、長さの最大値と最小値を求める。
ステップ１６２０）　長さの最大値と最小値に基づき、最大値、最大値－１、最小値＋１、最小値の4パターンの長さをもつ空白の文字列Tを作成する。

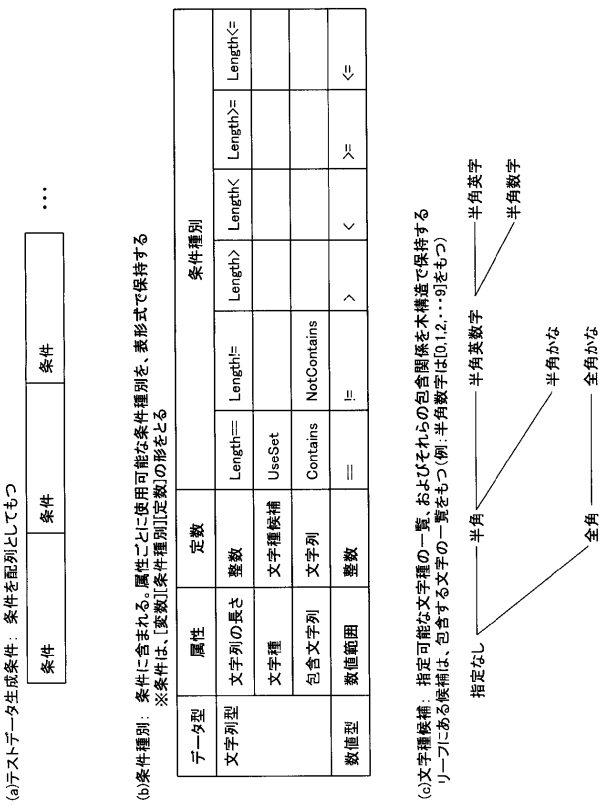
【０１４２】
ステップ１６３０）　包含文字列の条件で指定された包含文字列をそれぞれ、Tのランダムな位置に埋め込む。

【０１４３】
ステップ１６４０）　文字種の条件で指定された文字種に属する文字をランダムに生成し、Tの残りの位置を補完する。

【０１４４】
上記のように、値候補生成部８は、定義域記憶部７から取得した矛盾がないと保証された定義域の条件に基づいて、文字列の長さ、文字種、包含文字列の３つの属性をそれぞれ満足するように文字列を構成することで、複数条件を同時に満たした文字列型の入力データを生成することができる。
【０１４５】
以上の処理により、入力データを構成する変数のとる値候補が生成された。引き続き、図４の説明に戻る。
【０１４６】
ステップ１７０）　組合せ生成部９は、変数のそれぞれに、その変数に対し値候補生成部８で生成された値候補から1つ選び割り当てることで組合せを生成し、これを入力データとしてユーザ端末に出力する。
【０１４７】
上記の処理により、テストデータ生成条件を満たす文字列型の入力データを生成できる。
【０１４８】
本発明では、定義域記憶部７に定義域というデータ構造を設け、入力データが満たすべき条件を一元化することで、すべての条件を同時に満たした文字列型の入力データを生成し、テストケースの作成の効率が向上する。また、定義域の生成時に条件間の矛盾を検出することにより、設計誤りを検出し、設計の品質が向上する。
【０１４９】
なお、上記の図３に示すテスト用入力データ生成装置の構成要素の動作をプログラムとして構築し、テスト用入力データ生成装置として利用されるコンピュータにインストールして実行させる、または、ネットワークを介して流通させることが可能である。
また、構築されたプログラムをハードディスクや、フレキシブルディスク・ＣＤ－ＲＯＭ等の可搬記憶媒体に格納し、コンピュータにインストールする、または、配布することが可能である。
【０１５０】
なお、本発明は、上記の実施の形態に限定されることなく、特許請求の範囲内において種々変更・応用が可能である。
【符号の説明】
【０１５１】
１　設計情報読込部
２　設計モデル分析部
３　実行経路抽出部
４　実行経路記憶部
５　条件抽出部
６　定義域生成部
７　定義域記憶部
８　値候補生成部
９　組合せ生成部
１０　設計モデル記憶部

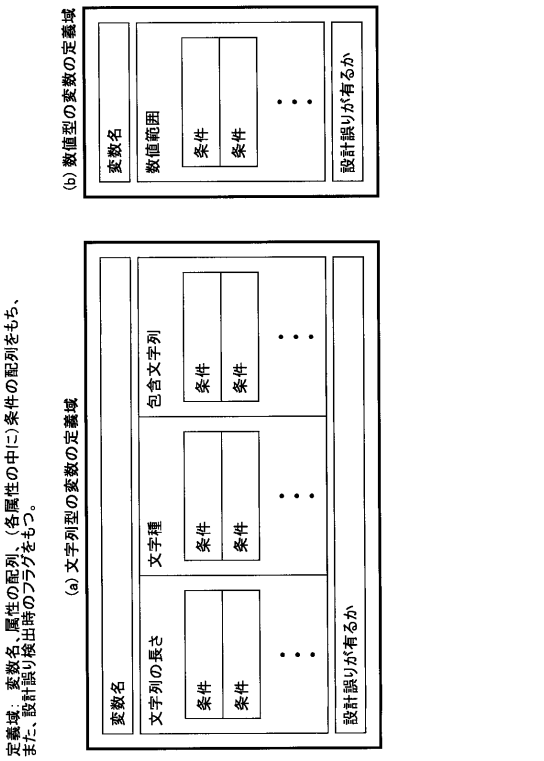
【図１】

本発明の一実施の形態におけるテストデータ生成条件のデータ構造



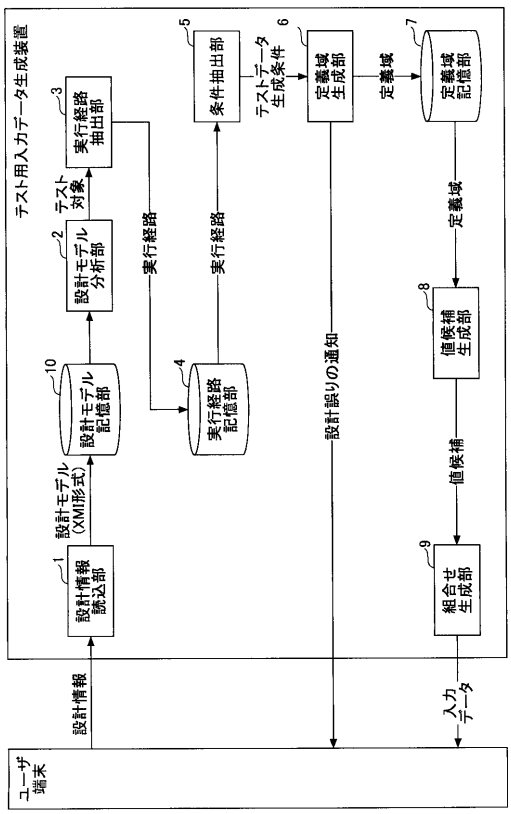
【図２】

本発明の一実施の形態における定義域記憶部の定義域のデータ構造



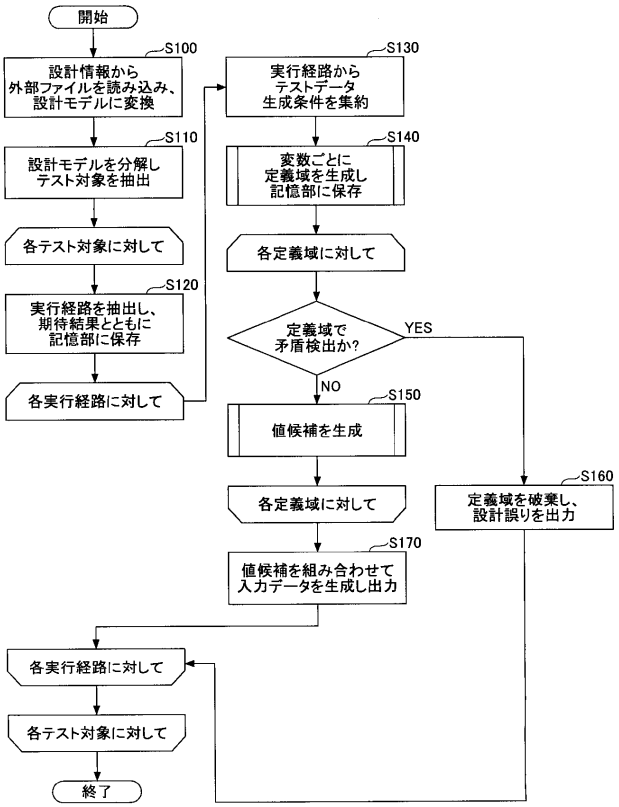
【図３】

本発明の一実施の形態における装置構成図



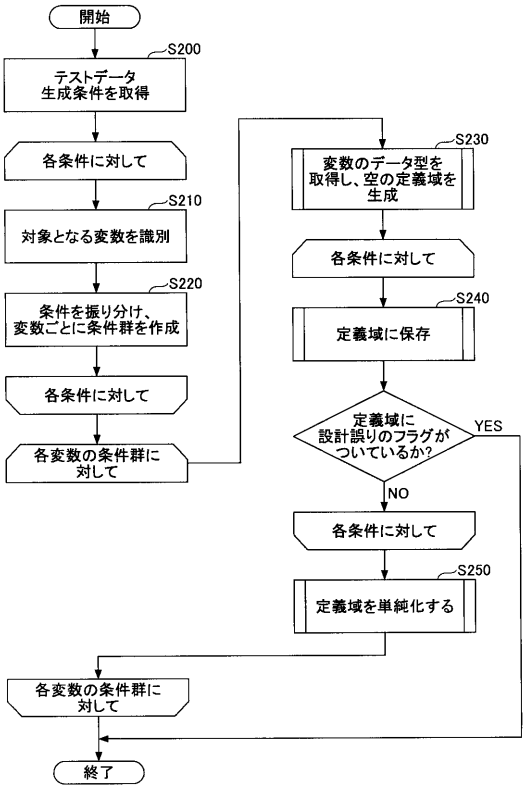
【図４】

本発明の一実施の形態におけるテスト用入力データ生成装置のフローチャート



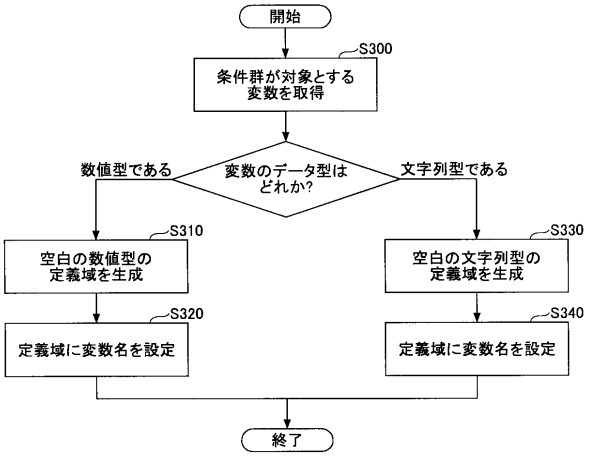
【図５】

本発明の一実施の形態における定義域生成部のフローチャート



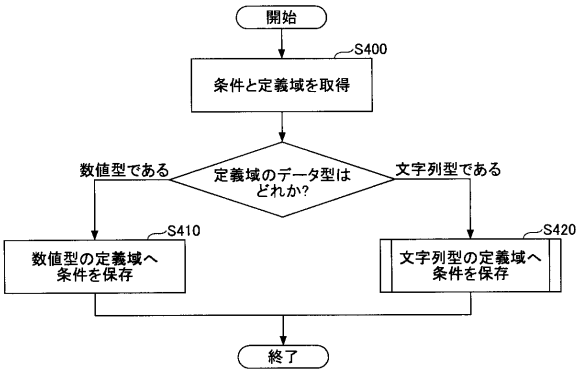
【図６】

本発明の一実施の形態における図5のS230の詳細なフローチャート



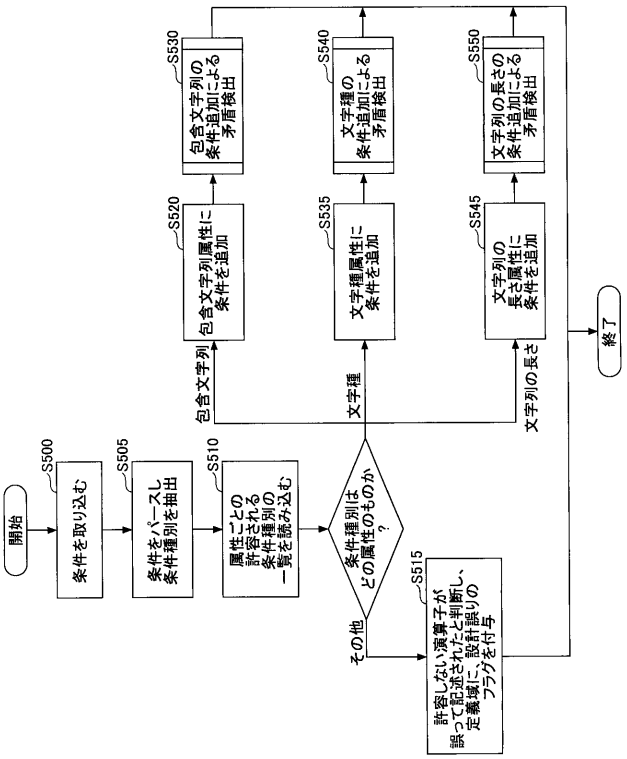
【図７】

本発明の一実施の形態における図5のS240の詳細なフローチャート



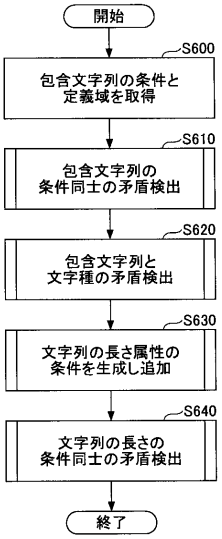
【図８】

本発明の一実施の形態における図7のS420の詳細なフローチャート



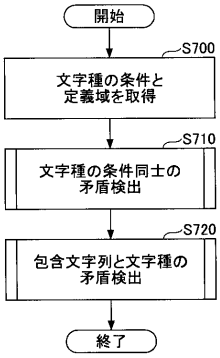
【図９】

本発明の一実施の形態における図8のS530の詳細なフローチャート



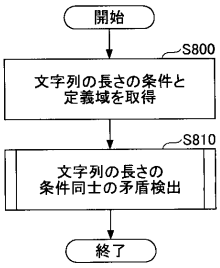
【図１０】

本発明の一実施の形態における図8のS540の詳細なフローチャート



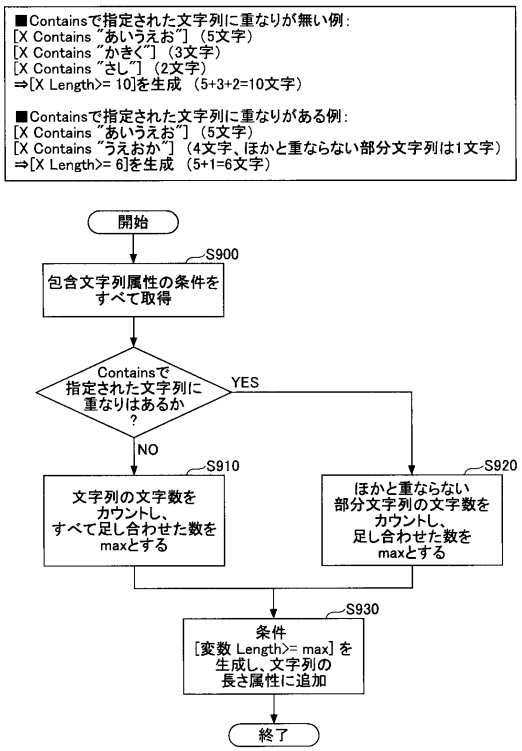
【図１１】

本発明の一実施の形態における図8のS550の詳細なフローチャート



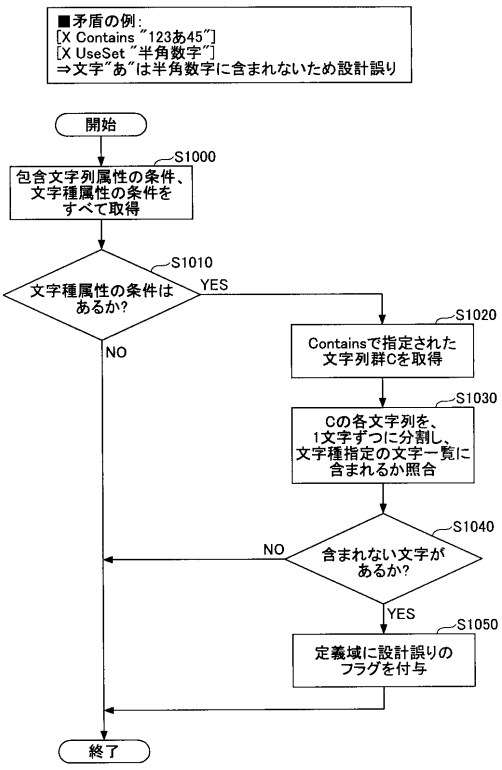
【図１２】

本発明の一実施の形態における図9のS630の詳細なフローチャート



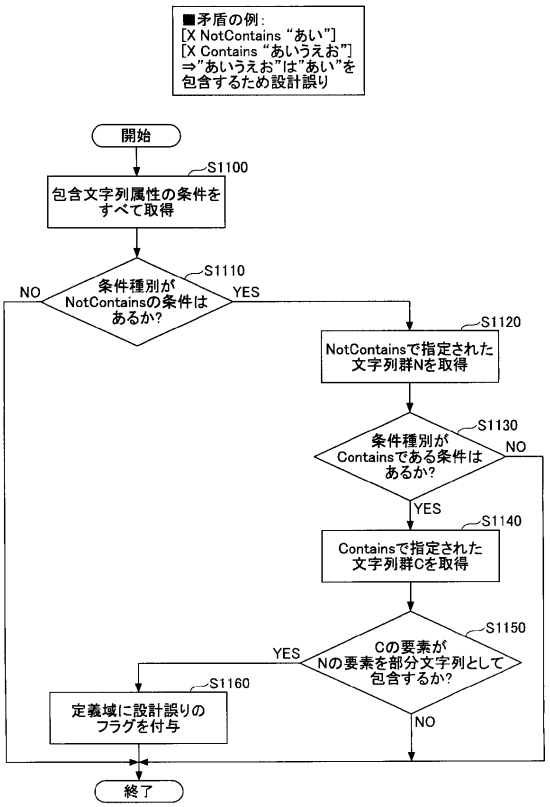
【図１３】

本発明の一実施の形態における
図9のS620, 図10のS720の詳細なフローチャート



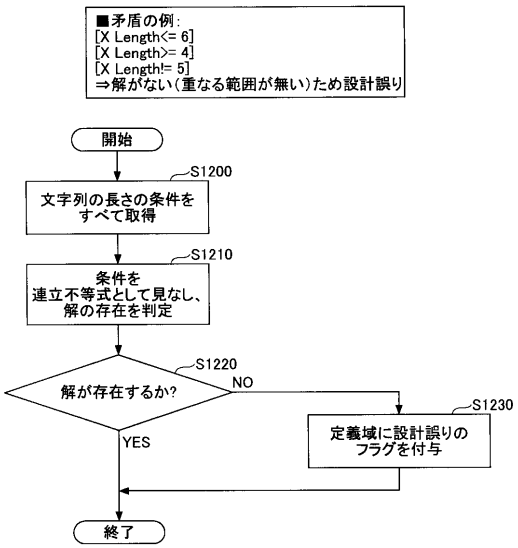
【図１４】

本発明の一実施の形態における図9のS610の詳細なフローチャート



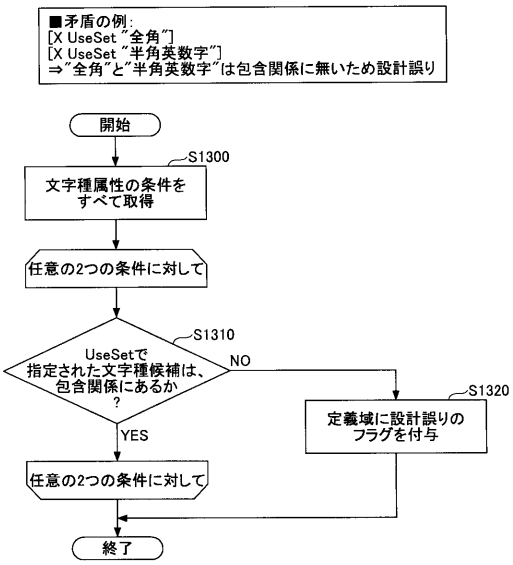
【図１５】

本発明の一実施の形態における
図9のS640，図11のS810の詳細なフローチャート



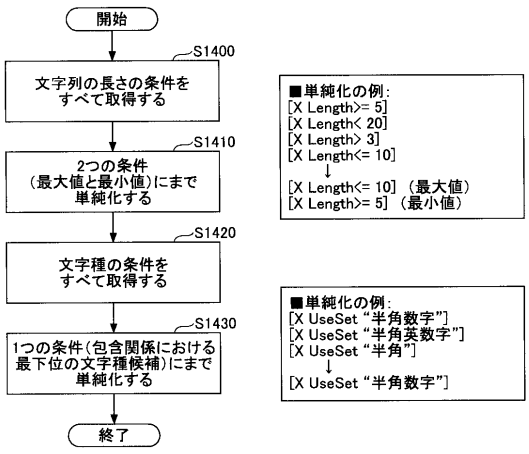
【図１６】

本発明の一実施の形態における図10のS710の詳細なフローチャート



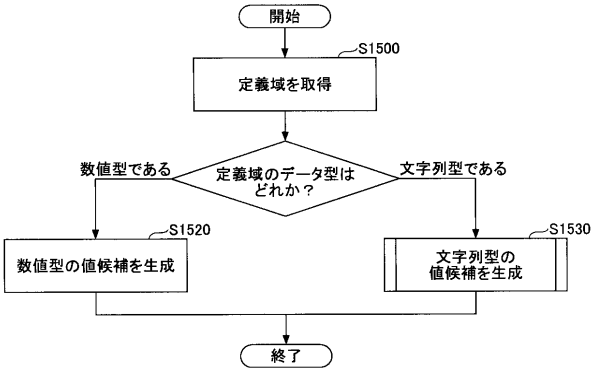
【図１７】

本発明の一実施の形態における図5のS250の詳細なフローチャート



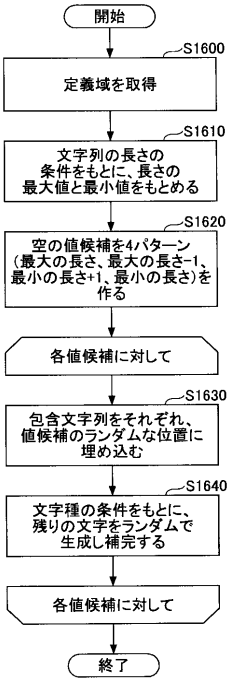
【図１８】

本発明の一実施の形態における値候補生成部のフローチャート



【図１９】

本発明の一実施の形態における図18のS1530の詳細なフローチャート



フロントページの続き

(72)発明者　星野　隆
東京都千代田区大手町二丁目３番１号　日本電信電話株式会社内

Ｆターム(参考) 5B042 HH08　HH17　KK13