(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5815856号
(P5815856)
(45)発行日　平成27年11月17日(2015.11.17)　(24)登録日　平成27年10月2日(2015.10.2)

(51)Int.Cl.　F I
G06F 11/28 (2006.01)　G06F 11/28 340A

請求項の数 12 (全 15 頁)

(21)出願番号	特願2014-515932(P2014-515932)
(86)(22)出願日	平成24年6月13日(2012.6.13)
(65)公表番号	特表2014-519671(P2014-519671A)
(43)公表日	平成26年8月14日(2014.8.14)
(86)国際出願番号	PCT/US2012/042127
(87)国際公開番号	WO2012/174033
(87)国際公開日	平成24年12月20日(2012.12.20)
審査請求日	平成25年12月12日(2013.12.12)
(31)優先権主張番号	13/160,021
(32)優先日	平成23年6月14日(2011.6.14)
(33)優先権主張国	米国(US)

(73)特許権者　502208397
グーグル　インコーポレイテッド
アメリカ合衆国　カリフォルニア州　９４０４３　マウンテン　ビュー　アンフィシアター　パークウェイ　１６００
(74)代理人　100092093
弁理士　辻居　幸一
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100086771
弁理士　西島　孝喜
(74)代理人　100109070
弁理士　須田　洋之

最終頁に続く

(54)【発明の名称】スクリプト依存関係をインライン化するためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
スクリプト依存関係をインライン化するためのコンピュータで実装される方法であって、
テストウェブページ（１６０）を定義する言語で指定されるテストリソース（１２２）を抽出することと、
前記テストページ（１６０）を定義する前記言語内で抽出された各テストリソース（１２２）の位置を識別するマーカー（１４２）を配置することと、
各テストリソース（１２２）のパスと関連付けられた外部リソース（１３２）を繰り返しロードすることと、
各テストリソース（１２２）を分析して、１つ以上の動的に追加された、各テストリソースの依存関係（１５２）を識別することと、
各マーカー（１４２）を、当該マーカー（１４２）それぞれを参照する外部リソース（１３２）および当該マーカー（１４２）それぞれを参照する依存関係（１５２）と置き換えて、更新されたテストページ（１６０、１６２）を定義する更新された言語を生成することと、を含み、
各ステップが、１つ以上のプロセッサ（１１０）を使用して行われる、方法。
【請求項２】
最上位の親リソース（１２２）の後または前にそれぞれ識別された依存関係（１５２）を追加することと、

新しい依存関係（１５２）が識別されなくなるまで、前記分析および前記追加を行うことと、
前記最上位の親リソース（１２２）と関連付けられた親マーカー（１４２）に対する参照を、それぞれの新しい依存関係（１５２）に提供することと、
をさらに含む、請求項１に記載の方法。
【請求項３】
前記パスが、外部リソース（１３２）をロードするために解決できないときに、前記繰り返しロードすることが、各テストリソース（１２２）およびそのパスの配置を維持することを含む、請求項１または２に記載の方法。
【請求項４】
前記分析することが、
スクリプトランタイムエンジン（１３４）を使用して、スクリプト（１２６）を評価することと、
前記スクリプト（１２６）を構文解析して、リソースの依存関係（１５２）の参照を識別することと、を含む、請求項１～３のいずれかに記載の方法。
【請求項５】
スクリプト依存関係をインライン化するためのコンピュータベースのシステムであって、
１つ以上のプロセッサ（３０６）と、
テストウェブページ（１６０）を定義する言語で指定されたテストリソース（１２２）を抽出するように構成される、テストリソース抽出器（１２０）と、
前記テストページ（１６０）を定義する前記言語内で各テストリソース（１２２）の位置を識別するマーカー（１４２）を維持するように構成される、マーカーモジュール（１４０）と、
各テストリソース（１２２）のパスと関連付けられた外部リソース（１３２）を繰り返しロードするように構成され、各テストリソース（１２２）を分析して、１つ以上の動的に追加された、各テストリソースの依存関係（１５２）を識別するように構成される、リソースローダーおよびアナライザ（１３０）と、
各マーカー（１４２）を、当該マーカー（１４２）それぞれを参照するリソース（１３２）および当該マーカー（１４２）それぞれ依存関係（１５２）と置き換えて、更新されたテストページ（１６０、１６２）を定義する更新された言語を生成するように構成される、インライン化モジュール（１５０）と、を備え、
前記テストリソース抽出器（１２０）、前記マーカーモジュール（１４０）、前記リソースローダーおよびアナライザ（１３０）、ならびに前記インライン化モジュール（１５０）は、前記１つ以上のプロセッサ（３０６）上で実装される、システム。
【請求項６】
前記リソースローダーおよびアナライザ（１３０）がさらに、
最上位の親リソース（１２２）の後または前に、それぞれの識別された依存関係（１５２）を追加し、
前記最上位の親リソース（１２２）と関連付けられた親マーカー（１４２）に対する参照を、それぞれの新しい依存関係（１５２）に提供するように構成される、
請求項５に記載のシステム。
【請求項７】
前記パスが外部リソースをロードするために解決できないときに、前記マーカーモジュール（１４０）がさらに、各テストリソース（１２２）およびそのパスの配置を維持するように構成される、請求項５または６に記載のシステム。
【請求項８】
前記リソースローダーおよびアナライザ（１３０）がさらに、
スクリプトランタイムエンジン（１３４）を使用してスクリプト（１２６）を評価し、
前記スクリプト（１２６）を構文解析して、リソース依存関係（１５２）の参照を識別す

るように構成される、
請求項５～７のいずれかに記載のシステム。
【請求項９】
命令がそこに記憶された非一時的コンピュータ可読媒体であって、コンピューティングデバイス（３０２）により実行されるとき、前記コンピューティングデバイス（３０２）に、
テストウェブページ（１６０）を定義する言語で指定されるテストリソース（１２２）を抽出することと、
前記テストページ（１６０）を定義する前記言語内で抽出された各テストリソース（１２２）の位置を識別するマーカー（１４２）を配置することと、
各テストリソース（１２２）のパスと関連付けられた外部リソース（１３２）を繰り返しロードすることと、
各テストリソース（１２２）を分析して、１つ以上の動的に追加された、各テストリソースの依存関係（１５２）を識別することと、
各マーカー（１４２）を、当該マーカー（１４２）それぞれを参照する外部リソース（１３２）および当該マーカー（１４２）それぞれを参照する依存関係（１５２）と置き換えて、更新されたテストページ（１６０、１６２）を定義する更新された言語を生成することと、
を含む操作を行わせる、非一時的コンピュータ可読媒体。
【請求項１０】
前記コンピューティングデバイスに、
最上位の親リソース（１２２）の後または前にそれぞれの識別された依存関係（１５２）を追加することと、
新しい依存関係（１５２）が識別されなくなるまで、分析および追加を行うことと、
最上位の親リソース（１２２）と関連付けられた親マーカー（１４２）に対する参照を、それぞれの新しい依存関係（１５２）に提供することと、
を含む、操作を行わせる命令をさらに含む、請求項９に記載の非一時的コンピュータ可読媒体。
【請求項１１】
前記繰り返しロードすることが、前記コンピューティングデバイスに、前記パスが外部リソース（１３２）をロードするために解決できないときに、各テストリソース（１２２）およびそのパスの配置を維持することを含む、操作を行わせる命令をさらに含む、請求項９または１０に記載の非一時的コンピュータ可読媒体。
【請求項１２】
前記分析が、
前記コンピューティングデバイスに、
スクリプトランタイムエンジン（１３４）を使用して、スクリプト（１２６）を評価することと、
前記スクリプト（１２６）を構文解析して、リソース依存関係（１５２）の参照を識別することと、
を含む操作を行わせる命令をさらに含む、請求項９～１１のいずれかに記載の非一時的コンピュータ可読媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
この開示は、スクリプトテスティングなどのテスティングフレームワークに関する。
【背景技術】
【０００２】
ユニットテスティングは、ソースコード（またはスクリプト）の個別のユニットを試験して、それらが使用に適しているかどうかを決定する方法である。一例において、ユニッ

トは、アプリケーションの最小のテスト可能な部分であってよい。別の例において、ユニットは、個別の関数または手順であってもよい。ユニットテストは、典型的に、そのコードが実行されるときに、意図されたとおりに動作することを保証するように、ソフトウェア開発者によって書き込まれ、実行される。ユニットテストは、一般に、テスティングフレームワークを使用して実行される。テスティングフレームワークは、スクリプトを含むウェブページをテストして使用され得る。ウェブページのこのようなテスティングは、例えば、開発者が、ウェブページ内のスクリプトが適切にロードし、実行するかどうかを決定することを可能にし得る。

【０００３】

ＪｓＵｎｉｔは、ＪａｖａＳｃｒｉｐｔのフレームワークをテストする、公的に使用可能なクライアント側（またはブラウザ側）ユニットの一例である。ＪｓＵｎｉｔフレームワークを使用するユニットテストは、テストのＪａｖａＳｃｒｉｐｔコードを含む必須ファイル（または「依存関係」）が適切に実行しているかを決定し、検索する。コード依存関係をテストのブラウザにロードする第１の手技において、ＪｓＵｎｉｔテスティングフレームワークは、依存関係を参照するテストページのハイパーテキストマークアップ言語（ＨＴＭＬ）部分にスクリプトタグ（例えば、「＜ｓｃｒｉｐｔ＞」を配置することができる。第２の手技において、ＪｓＵｎｉｔテスティングフレームワークは、ロードされたテストコードを使用して、依存関係を参照するスクリプトタグを動的に追加することができる。第３の手技において、ＪｓＵｎｉｔテスティングフレームワークは、サーバから（例えば、ｉＦｒａｍｅまたはＸＭＬＨｔｔｐＲｅｑｕｅｓｔを使用して）依存関係を動的に読み出すことができる。

【０００４】

これらの手技のそれぞれは、サーバ、ネットワーク、およびブラウザリソースを必要とし、テストセットアップに顕著により長い時間がかかる。テストの数が非自明になったときに、テストセットアップ時間は、テスト依存関係をロードするために必要な、ブラウザおよびサーバ内のリソースの消費に起因して増加する。加えて、減少した計算リソースおよびネットワークリソースのために、実際のテスト実行時間は増加し得、不適切なテスト障害をもたらす。上記の第２および第３の手技は、ブラウザリソースも消費し、より遅いテスト実行および場合によっては完全なブラウザ障害をも生じ得る。

【０００５】

いくつかのテスティングフレームワークは、サーバにおいて依存関係をキャッシュするか、またはブラウザの内蔵キャッシュに依存して依存関係を記憶し得る。サーバのキャッシングは、ブラウザに対する依存関係に寄与するために必要な時間およびリソースを低減するだけであり、依然としてネットワーク接続およびブラウザリソースを必要とする。さらに、ブラウザのキャッシュは一定でないため、ブラウザの内蔵キャッシュにおける依存関係のキャッシングは、スクリプトテストが適切にエラーなく実行していることを保証しない。

【発明の概要】

【０００６】

本開示の一態様は、スクリプト依存関係をインライン化する方法を提供し、テストウェブページを定義する言語で指定されたテストリソースを抽出することと、テストページを定義する言語内で抽出された各テストリソースの位置を識別するマーカーを配置することと、各テストリソースのパスと関連付けられた外部リソースを繰り返しロードすることと、各テストリソースを分析して１つ以上の動的に追加された依存関係を識別することと、各マーカーを、それらのそれぞれのマーカーを参照する外部リソースおよび依存関係と置き換えて、更新されたテストウェブページを定義する更新された言語を生成することと、を含む。本方法はさらに、最上位の親リソースの後または前にそれぞれ識別された依存関係を追加することと、新しい依存関係が識別されなくなるまで分析および追加を行うことと、最上位の親リソースと関連付けられた親マーカーに対する参照を、それぞれの新しい依存関係に提供することと、を含む。

【０００７】
このように、実装は、スクリプトテストのロードおよび実行時間を低減するために、スクリプト依存関係のインライン化を可能にする。
【０００８】
本開示の１つ以上の実装の詳細は、添付の図面および以下の説明に記載される。他の態様、特徴、および利点は、説明および図面から、ならびに請求項から明らかになるであろう。
【図面の簡単な説明】
【０００９】
要素が最初に現れる図面は、概して、対応する参照番号の左端の桁によって表示される。
【００１０】
【図１】テストコード内にスクリプト依存関係をインライン化するための例示のシステムを示す。
【図２Ａ】テスティングエンジンの例示の全体操作を示す。
【図２Ｂ】テスティングエンジンによって行われる例示の操作を示す。
【図３】論じられる実装のコンポーネントを実装するために有用なコンピュータの一例を示す。
【００１１】
様々な図面における同様の参照記号は、同様の要素を示す。
【発明を実施するための形態】
【００１２】
実施形態は、スクリプト依存関係のインライン化に関する。実施形態は、テストページを定義するＨＴＭＬ内に、テストリソースおよびそれらの依存関係をインライン化する（または含める）ことができる。テストリソース（例えば、スクリプト）およびそれらの依存関係（例えば、外部ファイル）は、テストページをテストする前に検索され、テストページを定義するＨＴＭＬ内に直接含まれるため、実施形態は、テストランタイムにテストリソースおよびそれらの依存関係をメモリにロードするために、サーバおよびブラウザリソースを消費する必要がない。このように、実施形態は、スクリプトテストのロードおよび実行時間を低減する。
【００１３】
実施形態は、テストページを定義する言語で指定されたテストリソースを抽出し、テストページを定義するＨＴＭＬ内に、抽出された各テストリソースの位置を識別するマーカーを配置する。このマーカーは、実施形態が抽出されたテストリソースの位置を追跡することを可能にする一方で、テストリソースと関連付けられた依存関係が検索される。さらに、マーカーは、実施形態が、検索された親子の依存関係がテストページを定義するＨＴＭＬ内の正しい階層に適切に含まれるように、検索された親子の依存関係の正しい階層を維持することも可能にする。
【００１４】
実施形態は、各テストリソースのパスと関連付けられた外部リソースをロードし、各テストリソースを分析して、動的に追加された依存関係を識別する一方で、各マーカーを、それらのそれぞれのマーカーを参照する外部リソースおよび依存関係と置き換えて、更新された（またはインライン化された）テストページを定義する更新された言語を生成する。次いでスクリプトテストは、インライン化されたテストページを使用して実行され得る。
【００１５】
実施形態はさらに、最上位の親リソースの後または前に、それぞれの識別された依存関係を追加することと、新しい依存関係が識別されなくなるまで、各テストリソースの分析および追加を行うことと、を含む。実施形態は、最上位の親リソースと関連付けられた親マーカーに対する参照を、それぞれの新しい依存関係に提供する。

【００１６】
以下は、特定適用の例示の実施形態を参照して本明細書において論じられるが、実施形態はそれに限定されないことを理解すべきである。本明細書において提供される教示にアクセスできる当業者であれば、その範囲内の追加の修正および適用、ならびに実施形態が重要なユーティリティとなる追加の分野を認識するであろう。
【００１７】
システム
【００１８】
このセクションは、図１に示される実施形態による、スクリプト依存関係をインライン化するためのシステムを説明する。図１は、スクリプト依存関係をインライン化するためのシステム１００の図である。以下はＪａｖａＳｃｒｉｐｔおよびＨＴＭＬに関して説明されるが、実施形態は、ＪａｖａＳｃｒｉｐｔおよびＨＴＭＬに限定されず、実行可能なコンテンツ、コード、またはコンテンツを定義する（またはマークアップ）言語の任意の他の形態に適用され得る。実施形態は、概して図１の構造を有する任意のシステムに適用可能であるか、または本明細書に記載される操作、方法、および関数から利益を享受する。
【００１９】
図１は、一実施形態によるテスティングエンジン１１０のアーキテクチャ図である。図１に示されるように、テスティングエンジン１１０は、テストリソース抽出器１２０、リソースローダーおよびアナライザ１３０、マーカーモジュール１４０、ならびにインライン化モジュール１５０を含む。図１はまた、テストページ１６０およびインライン化されたテストページ１６２を含む。一実施形態において、テストリソース抽出器１２０は、テストページ１６０を定義する言語（例えば、ＨＴＭＬ）で指定されたテストリソース１２２を抽出するように構成される。マーカーモジュール１４０は、テストページ１６０を定義する言語内で、各テストリソース１２２の位置を識別するためにマーカー１４２を配置するように構成される。リソースローダーおよびアナライザ１３０は、各テストリソース１２２のパスと関連付けられたメモリ外部リソース１３２に繰り返しロードし、各テストリソース１２２を分析して動的に追加された依存関係１５２を識別するように構成される。一実施形態において、インライン化操作は、関数呼び出しサイトを呼び出された関数の本体（またはコード）と置き換えることを含んでよい。インライン化モジュール１５０は、各マーカー１４２を、それらのそれぞれのマーカー１４２を参照するリソース１３２および依存関係１５２と置き換えて、インライン化されたテストページ１６２を定義する更新された言語を生成する。一実施形態において、次いでテスティングエンジン１１０は、インライン化されたテストページ１６２を使用してスクリプトテストを行う。テストリソース１２２（例えば、スクリプト）およびそれらの依存関係１５２（例えば、外部ファイル）は、テストページ１６０をテストする前に検索され、インライン化されたテストページ１６２を定義するＨＴＭＬ内に直接含まれるため、実施形態は、テストランタイムにテストリソースおよびそれらの依存関係１５２をメモリにロードするために、サーバおよびブラウザリソースを消費する必要がない。このように、実施形態は、スクリプトテストのロードおよび実行時間を低減する。
【００２０】
テストリソース抽出器１２０、リソースローダーおよびアナライザ１３０、マーカーモジュール１４０、ならびにインライン化モジュール１５０の操作は、以下でさらに論じられる。
【００２１】
テスティングエンジン１１０は、１つ以上のプロセッサを有する任意の種類のプロセッシング（またはコンピューティング）デバイスであり得る。例えば、テスティングエンジン１１０は、ワークステーション、モバイルデバイス、コンピュータ、コンピュータのクラスタ、セットトップボックス、または少なくとも１つのプロセッサを有する他のデバイスであり得る。このようなデバイスは、これに限定されないが、命令を実行および記憶す

るためのプロセッサおよびメモリを有するデバイスを含んでよい。このようなデバイスは、ソフトウェア、ファームウェア、およびハードウェアを含んでよい。ソフトウェアは、１つ以上のアプリケーションおよびオペレーティングシステムを含んでよい。ハードウェアは、これらに限定されないが、プロセッサ、メモリ、およびユーザインターフェースディスプレイを含み得る。任意の入力デバイス、例えば、マウス、タッチセンサー式スクリーン、または任意の今後開発される相互作用方法が使用されてよい。

【００２２】

試験エンジン１１０は、ブラウザ内で実装されてもよい。このようなブラウザは、１つ以上のプロセッサを有する任意の種類のプロセッシング（またはコンピューティング）デバイス上で実装され得る。一実施形態において、テスティングエンジン１１０は、ネットワーク上でテスティングエンジン１１０と通信するサーバ（図示せず）上で例示化される、Ｊａｖａ「サーブレット」としても実装されてよい。テスティングエンジン１１０は、スタンドアロンアプリケーションまたはモジュールとして実装されてもよい。

【００２３】

一実施形態において、テスティングエンジン１１０は、テストページ１６０を読み出し、スクリプト依存関係１５２がテスティングエンジン１１０によってインライン化された、インライン化テストページ１６２を生成するように構成される。一実施形態において、テストページ１６０は、ＨＴＭＬファイルまたは任意の他のファイルであってよい。一実施形態において、テストページ１６０は、１つ以上のテストリソース１２２を含んでよい。テストリソース１２２は、テストページ１６０がテスティングエンジン１１０によって読み出されるときに実行され得る、スクリプトコード（例えば、ＪａｖａＳｃｒｉｐｔ）を含むテスト関数であってよい。一実施形態において、テストページ１６０内のテストリソース１２２は、テストリソース１２２に隣接して現れる（または被包する）タグ（または識別子）を使用して、テスティングエンジン１１０により識別されてよい。テストリソース１２２がＪａｖａＳｃｒｉｐｔである一例において、このようなタグは、スクリプトタグ１２４（例えば、「＜ｓｃｒｉｐｔ＞」）であってよい。当業者に知られているように、スクリプトタグは、ＪａｖａＳｃｒｉｐｔなどのクライアント側スクリプトを定義および識別するために使用される。スクリプトタグは、スクリプト命令文を含み得るか、または「ｓｒｃ」属性を通じて外部スクリプトファイルを指し示し得る。当然のことながら、実施形態は、スクリプトタグおよびＪａｖａＳｃｒｉｐｔに限定されず、現在知られているか、または今後開発される他の形態のテストリソース識別子とともに使用されてよい。

【００２４】

一実施形態において、テストリソース抽出器１２０は、テストページ１６０を定義する言語（例えば、ＨＴＭＬ）に含まれるか、または指定されたテストリソース１２２を抽出するように構成される。テストリソース１２２は、テストページ１６０がテストのためにテスティングエンジン１１０によって読み出されるときに実行され得る、スクリプトコード（例えば、ＪａｖａＳｃｒｉｐｔ）を含むテスト関数であってよい。一実施形態において、テストページ１６０内のテストリソース１２２は、テストリソース１２２に隣接して現れる（または被包する）タグ（または識別子）によって識別されてよい。上記のように、このようなスクリプトタグは、テストページ１６０を定義するＨＴＭＬ内に組み込まれてよい。一実施形態において、テストリソース抽出器１２０は、テストページ１６０を構文解析して、スクリプトタグおよび関連付けられたスクリプトを識別し、抽出する。

【００２５】

一実施形態において、マーカーモジュール１４０は、テストページ１６０を定義するＨＴＭＬ内で抽出された各テストリソース１２２の位置を識別するマーカー１４２を配置するように構成されてよい。マーカー１４２は、抽出された各テストリソース１３２の位置を識別および維持するためにマーカーモジュール１４０によって使用される、任意の形態の識別子またはデータ構造であり得る。

【００２６】

一実施形態において、リソースローダーおよびアナライザ１３０は、テストページ１６０に指定された各テストリソース１２２のパスと関連付けられたメモリ外部リソース１３２（例えば、ファイル、画像、スクリプトなど）にロードするように構成され得る。一実施形態において、リソースローダーおよびアナライザ１３０は、各テストリソース１２２のパス上で繰り返し、テストリソース１２２のパスによって参照され得る外部依存関係１５２をロードしてよい。シナリオ例において、テストリソース１２２は、テストリソース１２２が適切に実行するために、外部リソース１３２（例えば、ファイル、画像、スクリプトなど）を必要とし得る。このような外部リソース１３２は、テスティングエンジン１１０から独立したサーバに位置し得るか、またはさらにはテスティングエンジン１１０が作動しているコンピューティングデバイス上に位置してもよい。テストリソース１２２により必要とされる外部リソース１３２（または他の依存関係１５２）を、事前にメモリに繰り返しロードすることによって、外部リソース１３２のこのようなロードは、テストが行われているときに実時間で発生する必要がないため、テストを実行するために要する時間がより少なくなる。

【００２７】

一実施形態において、リソースローダーおよびアナライザ１３０が、テストリソースパスが解決可能でない（またはアクセス可能でない）ことを決定した場合、リソースローダーおよびアナライザ１３０およびマーカーモジュール１４０は、テストリソース１２２の配置を維持し、解決不可能なパスとともに、テスティングエンジン１１０がランタイムに、テストリソースパスと関連付けられたリソースをロードすることを可能にする。

【００２８】

一実施形態において、リソースローダーおよびアナライザ１３０は、ロードされた各テストリソース１２２を分析して、動的に追加された依存関係１５２を識別する。動的に追加された依存関係１５２は、ランタイムにテストリソース１２２により必要とされ得るスクリプトまたは任意の他のファイルであってよい。一実施形態において、リソースローダーおよびアナライザ１３０は、スクリプトランタイムエンジン（例えば、ＪａｖａＳｃｒｉｐｔランタイムエンジン）を使用してスクリプトを評価し、テストリソース１２２を構文解析して依存関係のヒントを識別することによって、動的に追加された依存関係１５２を識別し得る。このような依存関係のヒントは、例えば、リソース要件呼び出しを含んでよい。

【００２９】

一実施形態において、リソースローダーおよびアナライザ１３０は、最上位親リソース１２２の後（すなわち、ブラウザのランタイムＨＴＭＬロード順序を模倣する）、または最上位親リソース１２２の前（すなわち、動的解決ロード順序を模倣する）のいずれかに、それぞれの新しく識別された依存関係１５２を追加してよい。マーカーモジュール１４０はまた、最上位親リソース１２２と関連付けられた親マーカー１４２に対する参照を、それぞれの新しい依存関係１５２に提供する。

【００３０】

一実施形態において、リソースローダーおよびアナライザ１３０は、新しいリソース１２２が識別されなくなるまで、最上位レベルの親リソース１２２に対するそれぞれの新しい依存関係１５２を識別および追加し続ける。このように、テストリソース１２２により必要とされる外部リソース１３２（または他の依存関係）を、テストに先立って繰り返しロードすることによって、外部リソース１３２のこのようなロードは、テストが行われているときに実時間で発生する必要がないため、テストを実行するために要する時間がより少なくなる。

【００３１】

上記のように、マーカーモジュール１４０は、テストページ１６０を定義するＨＴＭＬ内で抽出された各テストリソース１３２の位置を識別するマーカー１４２を配置するように構成され得る。マーカー１４２は、抽出された各テストリソース１２２の位置を識別および維持するために、マーカーモジュール１４０により使用される、任意の形態の識別子

またはデータ構造であり得る。一実施形態において、インライン化モジュール１５０は、各マーカー１４２を、それらのそれぞれのマーカー１４２を参照する抽出されたリソース１３２および依存関係１５２と置き換えて、インライン化されたテストページ１６２を生成する。

【００３２】

一実施形態において、任意の未解決のマーカー１４２（または未解決のパスと関連付けられたマーカー）は、テスティングエンジン１１０がランタイムにそれらの未解決のパスと関連付けられたリソース１２２をロードし得るように変化していない。次いでテスティングエンジン１１０は、インライン化されたテストページ１６２を使用してスクリプトテストを行う。テストリソース１２２（例えば、スクリプト）およびそれらの依存関係１５２（例えば、外部ファイル）は、テストページ１６０を試験する前に検索され、インライン化されたテストページ１６２を定義するＨＴＭＬ内に直接含まれるため、実施形態は、テストランタイムにテストリソース１２２およびそれらの依存関係１５２をメモリにロードするために、サーバおよびブラウザリソースを消費する必要がない。このように、実施形態は、スクリプトテストのロードおよび実行時間を低減する。

【００３３】

図２Ａは、方法２００を示すフローチャートである。方法２００は、実施形態によるテスティングエンジン１１０の例示の全体操作である。

【００３４】

方法２００は、テストページ１６０を読み出すテスティングエンジン１１０から始まる（ステップ２０２）。一例として、このようなテストページ１６０は、ＨＴＭＬファイルまたは任意の他のファイルであってよい。一実施形態において、テストページ１６０は、１つ以上のテストリソース１２２を含む。テストリソース１２２は、スクリプトコード（例えば、ＪａｖａＳｃｒｉｐｔ）を含んでよい。

【００３５】

テストリソース抽出器１２０は、テストページ１６０のテストリソース１２２と関連付けられたスクリプトタグ１２４を抽出する（ステップ２０４）。上述のように、このようなスクリプトタグ１２４は、テストページ１６０を定義するＨＴＭＬ内に組み込まれてよい。一実施形態において、テストリソース抽出器１２０は、テストページ１６０を構文解析して、スクリプトタグ１２４および関連付けられたスクリプト１２６を識別および抽出する。

【００３６】

リソースローダーおよびアナライザ１３０は、各テストリソース１２２のパスと関連付けられた外部リソース１３２をロードする（ステップ２０６）。シナリオ例において、テストリソース１２２は、テストリソース１２２が適切に実行するために、外部リソース１３２（例えば、ファイル）を必要とし得る。このような外部リソース１３２は、テスティングエンジン１１０から独立したサーバに位置し得るか、またはさらにはテスティングエンジン１１０が作動しているコンピューティングデバイス上に位置してもよい。このような外部リソース１３２は、リソースローダーおよびアナライザ１３０によって繰り返しロードされてよい。

【００３７】

リソースローダーおよびアナライザ１３０は、解決できない任意の外部テストリソースパスが存在するかどうかもチェックする（ステップ２０８）。リソースローダーおよびアナライザ１３０が、テストリソースパスが解決可能でないことを決定した場合（ステップ２０８）、リソースローダーおよびアナライザ１３０ならびにマーカーモジュール１４０は、解決不可能なパス（ステップ２１０）と共にテストリソース１２２の配置を維持し、そして方法２００はステップ２１２に進む。

【００３８】

ステップ２０８に戻ると、リソースローダーおよびアナライザ１３０が、すべての外部テストリソースパスが解決されたことを決定した場合（ステップ２０８）、リソースロー

ダーおよびアナライザ１３０は、ロードされた各テストリソース１２２を分析して、動的に追加された依存関係１５２を識別する（ステップ２１２）。一例として、動的に追加された依存関係１５２は、ランタイムにテストリソース１２２によって必要とされ得る、スクリプト１２６または任意の他のファイルであり得る。

【００３９】

リソースローダーおよびアナライザ１３０は、新しい依存関係１５２がステップ２１２において識別されたかどうかをチェックする（ステップ２１４）。リソースローダーおよびアナライザ１３０が新しい依存関係１５２を識別した場合（ステップ２１４）、方法２００はステップ２０６に進み、リソースローダーおよびアナライザ１３０は、各テストリソース１２２のパスと関連付けられた外部リソース１３２をロードする。リソースローダーおよびアナライザ１３０が新しい依存関係を識別しない場合（ステップ２１４）、インライン化モジュール１５０は、各マーカー１４２を、それらをそれぞれのマーカー１４２を参照するリソース１２２および依存関係１５２と置き換えて、インライン化されたテストページ１６２を生成する（ステップ２１６）。

【００４０】

図２Ｂは、実施形態による方法２００のステップ２１４を詳細に示すフローチャートである。一実施形態において、リソースローダーおよびアナライザ１３０は、スクリプトランタイムエンジン１３４（例えば、ＪａｖａＳｃｒｉｐｔランタイム）を使用してスクリプト１２６を評価し、テストリソース１２２（例えば、スクリプト）を構文解析して、依存関係の「ヒント」を識別することによって、動的に追加された依存関係１５２を識別する（ステップ２２０）。このようなヒントは、例えば、リソース要件呼び出しを含んでよい。

【００４１】

リソースローダーおよびアナライザ１３０は、最上位親リソース１２２の後（すなわち、ブラウザのランタイムＨＴＭＬロード順序を模倣する）、または親リソース１２２の前（すなわち、動的解決ロード順序を模倣する）のいずれかに、それぞれの新しく識別された依存関係１５２を追加する（ステップ２２２）。マーカーモジュール１４０はまた、最上位親リソース１２２と関連付けられた親マーカー１４２に対する参照を、それぞれの新しい依存関係１５２に提供する（ステップ２２４）。

【００４２】

このように、テストリソース１２２により必要とされる外部リソース１３２（または他の依存関係）を、テストページ１６０のテストに先立って繰り返しロードすることによって、外部リソース１３２のこのようなロードは、テストが行われているときに実時間で発生する必要がないため、テストを実行するために要する時間がより少なくなる。

【００４３】

コンピュータ実施形態例

【００４４】

一実施形態において、本明細書に記載される実施形態のシステムおよびコンポーネントは、図３に示される例示のコンピュータ３０２などの１つ以上のコンピュータを使用して実装される。例えば、テスティングエンジン１１０は、コンピュータ（複数可）３０２を使用して実装され得る。

【００４５】

コンピュータ３０２は、本明細書に記載される関数を行うことができる、任意の市販のよく知られているコンピュータであり得る。

【００４６】

コンピュータ３０２は、プロセッサ３０６などの１つ以上のプロセッサ（中央処理ユニットまたはＣＰＵとも呼ばれる）を含む。プロセッサ３０６は、通信インフラストラクチャ３０４に接続される。

【００４７】

コンピュータ３０２は、ランダムアクセスメモリ（ＲＡＭ）などのメインメモリまたは

一次メモリ３０８も含む。一次メモリ３０８は、そこに記憶された制御論理３６８Ａ（コンピュータソフトウェア）、およびデータを有する。

【００４８】

コンピュータ３０２は、１つ以上の二次記憶デバイス３１０も含む。二次記憶デバイス３１０は、例えば、ハードディスクドライブ３１２および／またはリムーバブル記憶デバイスまたはドライブ３１４、ならびに他の種類の記憶デバイス、例えば、メモリカードおよびメモリスティックを含む。リムーバブル記憶ドライブ３１４は、フロッピーディスクドライブ、磁気テープドライブ、コンパクトディスクドライブ、光学記憶ドライブ、テープバックアップなどを表す。

【００４９】

リムーバブル記憶ドライブ３１４は、リムーバブル記憶ユニット３１６と相互作用する。リムーバブル記憶ユニット３１６は、そこに記憶されたコンピュータソフトウェア３６８Ｂ（制御論理）および／またはデータを有する、コンピュータ使用可能または読み出し可能な記憶媒体３６４Ａを含む。リムーバブル記憶ユニット３１６は、フロッピーディスク、磁気テープ、コンパクトディスク、ＤＶＤ、光学記憶ディスク、または任意の他のコンピュータデータ記憶デバイスを表す。リムーバブル記憶ドライブ３１４は、よく知られている方法で、リムーバブル記憶ユニット３１６から読み出し、および／またはそこに書き込む。

【００５０】

コンピュータ３０２は、入力／出力／表示デバイス３６６、例えば、モニター、キーボード、ポインティングデバイス、Ｂｌｕｅｔｏｏｔｈデバイスなども含む。

【００５１】

コンピュータ３０２はさらに、通信またはネットワークインターフェース３１８も含む。ネットワークインターフェース３１８は、コンピュータ３０２が遠隔デバイスと通信することを可能にする。例えば、ネットワークインターフェース３１８は、コンピュータ３０２が、通信ネットワークまたは媒体３６４Ｂ（コンピュータで使用可能または読み出し可能な媒体の形態を表す）、例えば、ＬＡＮ、ＷＡＮ、インターネット上で通信することを可能にする。ネットワークインターフェース３１８は、有線または無線接続を介して、遠隔サイトまたはネットワークと連動し得る。

【００５２】

制御論理３６８Ｃは、通信媒体３６４Ｂを介してコンピュータ３０２を往復して伝達され得る。

【００５３】

制御論理（ソフトウェア）がそこに記憶されたコンピュータで使用可能または読み出し可能な媒体を備える任意の有形装置または製造品は、コンピュータプログラム製品またはプログラム記憶デバイスと称される。これには、それらに限定されないが、コンピュータ３０２、メインメモリ３０８、二次記憶デバイス３１０、およびリムーバブル記憶ユニット３１６が挙げられる。１つ以上のデータ処理デバイスにより実行されるとき、このようなデータ処理デバイスを本明細書に記載されるように動作させる、制御論理がそこに記憶されたこのようなコンピュータプログラム製品は、実施形態を表す。

【００５４】

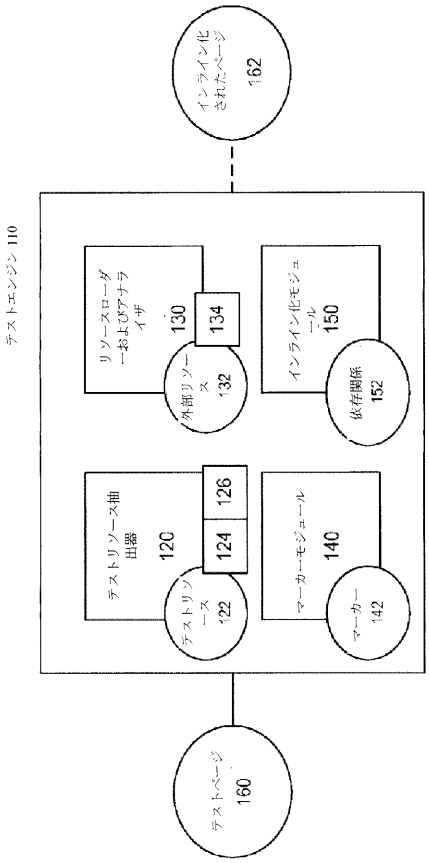
実施形態は、本明細書に記載されるもの以外のソフトウェア、ハードウェア、および／またはオペレーティングシステムと連携することができる。本明細書に記載される関数を行うために適した任意のソフトウェア、ハードウェア、およびオペレーティングシステムの実装が使用され得る。実施形態は、クライアントおよびサーバの両方、またはそれらの組合せに適用可能である。

【００５５】

まとめおよび要約セクションは、発明者（複数可）により意図される１つ以上であるがすべてではない例示の実施形態について説明され得るため、いかなる方法においても本実施形態および添付の請求項を制限することを意図しない。

【００５６】
本実施形態は、特定された関数およびその関係の実装を示す、関数構成要素を用いて上述された。これらの関数構成要素の境界は、説明の便宜上、本明細書において任意に定義された。代替境界は、特定の関数およびその関係が適切に実行される限り定義され得る。
【００５７】
特定の実施形態に関する前記の説明は、それらの一般的な性質を完全に明らかにするため、当該技術分野内の知識を適用することによって、不要な実験を行うことなく、それらの一般的な概念から逸脱することなく、このような特定の実施形態を容易に修正し、および／または様々な適用に適合させることができる。したがって、このような適合および修正は、本明細書に提示される教示およびガイダンスに基づいて、開示される実施形態に相当する意味および範囲内であることが意図される。本明細書における表現または用語は、説明の目的であって限定するものではなく、本明細書の用語または表現が、その教示およびガイダンスに照らして熟練者により解釈されるものであることを理解されたい。
【００５８】
実施形態の広さおよび範囲は、上述の例示の実施形態のいずれかによって制限されるべきではないが、以下の請求項およびそれらに相当するものに従ってのみ定義されるべきである。

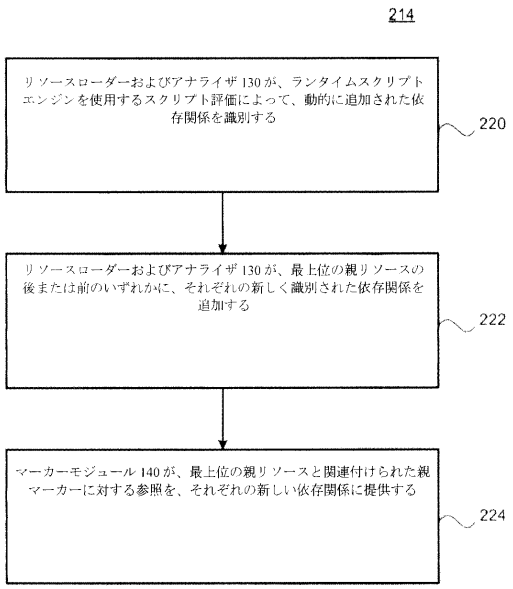
【図１】



【図１】

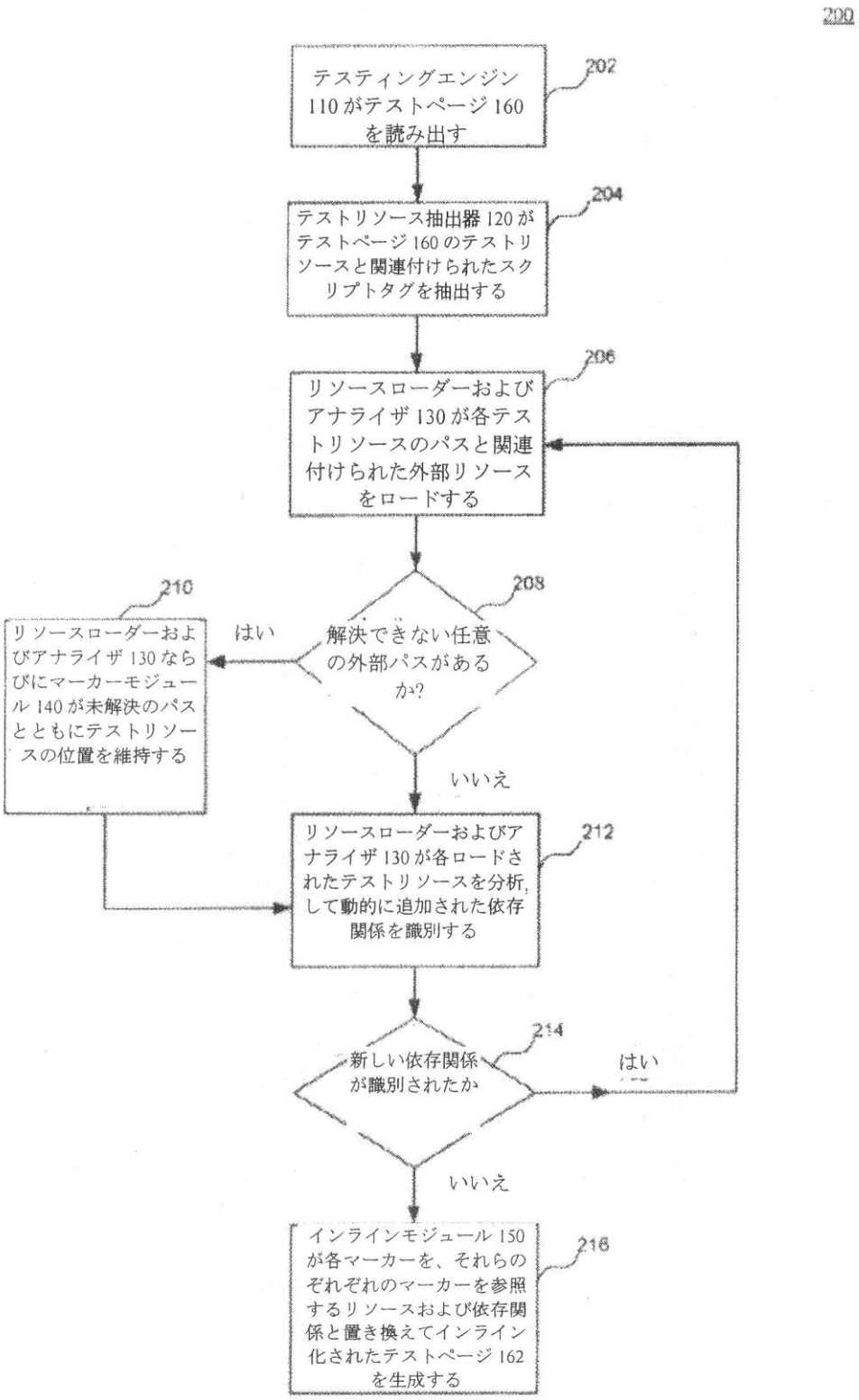
【図２Ｂ】

【図２Ｂ】



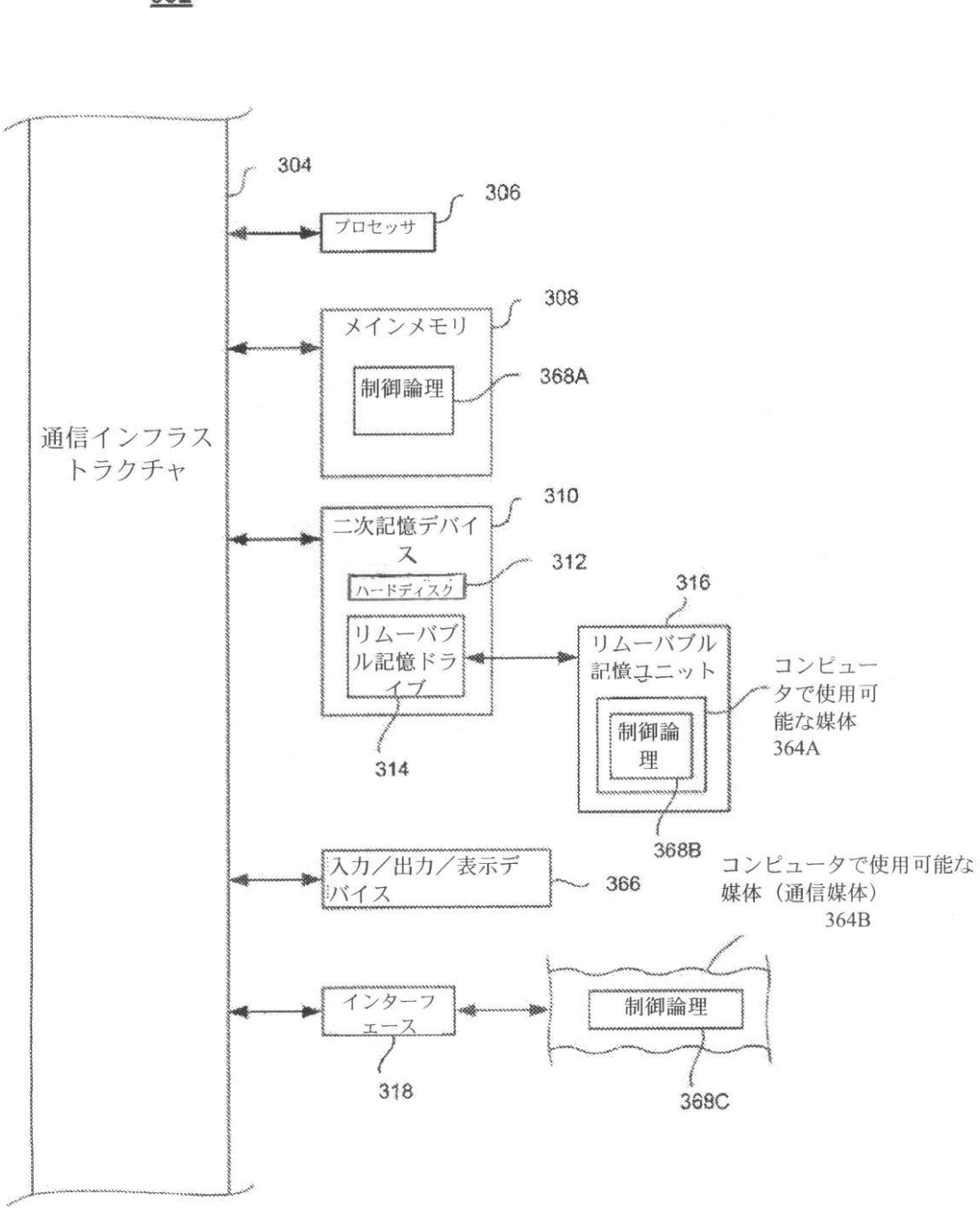
【図２Ａ】

【図２Ａ】


200
テスティングエンジン110がテストページ160を読み出す
202
テストリソース抽出器120がテストページ160のテストリソースと関連付けられたスクリプトタグを抽出する
204
リソースローダーおよびアナライザ130が各テストリソースのパスと関連付けられた外部リソースをロードする
206
解決できない任意の外部パスがあるか?
208
はい
リソースローダーおよびアナライザ130ならびにマーカーモジュール140が未解決のパスとともにテストリソースの位置を維持する
210
いいえ
リソースローダーおよびアナライザ130が各ロードされたテストリソースを分析して動的に追加された依存関係を識別する
212
新しい依存関係が識別されたか
214
はい
いいえ
インラインモジュール150が各マーカーを、それらのぞれぞれのマーカーを参照するリソースおよび依存関係と置き換えてインライン化されたテストページ162を生成する
216

【図３】

【図３】


302
304
306
プロセッサ
308
メインメモリ
制御論理
368A
通信インフラストラクチャ
310
二次記憶デバイス
312
ハードディスク
316
リムーバブル記憶ユニット
リムーバブル記憶ドライブ
コンピュータで使用可能な媒体
364A
制御論理
314
368B
コンピュータで使用可能な媒体（通信媒体）
364B
入力／出力／表示デバイス
366
制御論理
インターフェース
318
368C

フロントページの続き

(74)代理人 100109335
弁理士 上杉 浩
(74)代理人 100158551
弁理士 山崎 貴明
(72)発明者 スミス コービン
アメリカ合衆国 ニューヨーク州 ブルックリン グラハム アベニュー ４４１ アパートメント ２アール
(72)発明者 セシャドリ シャム
インド ムンバイ カールガール セクター ７ プロット １ １８０４エイ プライド

審査官 多胡 滋

(56)参考文献 特開２００７－１７９１５３（ＪＰ，Ａ）
伊藤千光，Ｇｏｏｇｌｅ ＡＰＩ Ｅｘｐｅｒｔが解説する Ｃｌｏｓｕｒｅ Ｌｉｂｒａｒｙ プログラミングガイド，日本，株式会社インプレスジャパン，２０１０年１２月１１日，第1版，pp.35-36

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １１／２８
Ｇ０６Ｆ １１／３６