(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2005-11342
(P2005-11342A)
(43)公開日　平成17年1月13日(2005.1.13)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 11/28	G06F	11/28	315A	5B042
G06F 9/44	G06F	9/44	530P	

審査請求 未請求　請求項の数 20　ＯＬ　（全 18 頁）

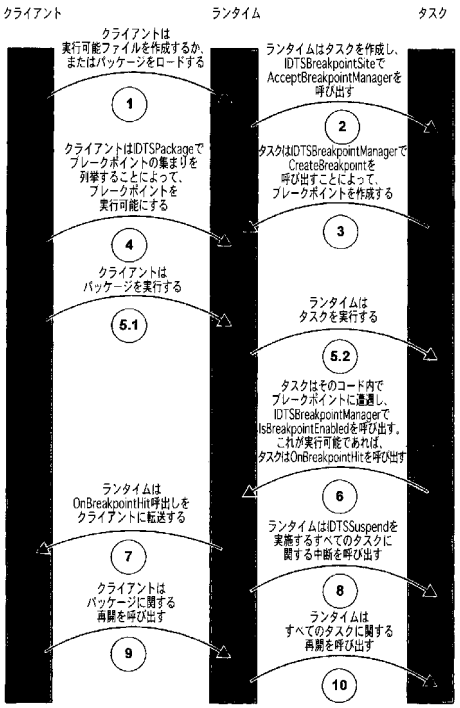
(21)出願番号　特願2004-178867(P2004-178867)
(22)出願日　平成16年6月16日(2004.6.16)
(31)優先権主張番号　10/600,066
(32)優先日　平成15年6月20日(2003.6.20)
(33)優先権主張国　米国(US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ジェイ．カーク　ハーゼルデン
アメリカ合衆国　９８０２９　ワシントン州　イサコア　サウスイースト　２２１　アベニュー　３５１３

最終頁に続く

(54)【発明の名称】プラグ可能構成要素におけるブレークポイントのデバッグ

(57)【要約】　（修正有）
【課題】プラグ可能構成要素上でのブレークポイントによるデバッグの提供。拡張可能なプラグ可能構成要素を有するコンピューティング環境では、エラーの観察、追跡、構成要素のランタイム中の様々なポイントでブレークポイントの設定を可能にする。
【解決手段】コンピューティング環境にはクライアント、ランタイム、およびタスクが含まれる。オペレーションでは、クライアントは、ランタイムおよびタスクと通信してデバッグオペレーションを実行する。タスクはブレークポイントを設定し、ランタイムを介してクライアントに送信される。ブレークポイントに遭遇するとタスクは中断され、タスクの実行に問題がある場合、クライアントはデバッグを実行することができる。満足すると、クライアントはランタイムを介してタスクオペレーションを再開する。
【選択図】　図３



【特許請求の範囲】
【請求項１】
オブジェクトモデルをデバッグするための方法であって、
実行環境のデバッグインターフェースを介してブレークポイントのセットを実現させる工程と、
所定の期間で前記ブレークポイントの状況をチェックする工程と
中断および再開の要求に応答する工程と
を具えたことを特徴とする方法。
【請求項２】
パッケージ配置構成要素とプラグ可能構成要素との間で通信する工程をさらに具えたことを特徴とする請求項１記載の方法。
【請求項３】
前記デバッグインターフェースによって処理するために入力を受け取る工程をさらに具え、
前記入力はパッケージを実行または配置するための命令を示すものであることを特徴とする請求項１記載の方法。
【請求項４】
少なくとも１つのタスクにおいて、前記ブレークポイントを設定する工程をさらに具えたことを特徴とする請求項１記載の方法。
【請求項５】
前記タスク実行中に、該タスクによって設定された前記ブレークポイントに遭遇することをさらに具えたことを特徴とする請求項４記載の方法。
【請求項６】
前記ブレークポイントを前記デバッグインターフェースに伝達する工程をさらに具えたことを特徴とする請求項５記載の方法。
【請求項７】
前記設定されたブレークポイントに遭遇する前に、前記タスクの状態を前記デバッグインターフェースによって分析する工程をさらに具えたことを特徴とする請求項６記載の方法。
【請求項８】
前記デバッグインターフェースによって前記タスクを再開する工程をさらに具えたことを特徴とする請求項７記載の方法。
【請求項９】
コンピュータに指示するための命令を有するコンピュータ読取り可能媒体であって、
実行環境のデバッグインターフェースを介してブレークポイントのセットを実現させる工程と、
所定の期間で前記ブレークポイントのセットの状況をチェックする工程と、
中断および再開の要求に応答する工程と
を具えたことを特徴とするコンピュータ読取り可能媒体。
【請求項１０】
オブジェクトモデルをデバッグするための方法であって、
前記オブジェクトモデルの１つまたは複数の構成要素と通信するインターフェースマネージャを提供する工程と、
ブレークポイントの場所を決定する工程と、
前記ブレークポイントに遭遇するようにランタイムを実行する工程と
を具えたことを特徴とする方法。
【請求項１１】
パッケージを実行する工程をさらに具えたことを特徴とする請求項１０記載の方法。
【請求項１２】
前記パッケージの実行の結果として、少なくとも１つのタスクを実行する工程をさらに

具えたことを特徴とする請求項１１記載の方法。
【請求項１３】
前記決定するステップは、前記インターフェースマネージャによってブレークポイントを設定する工程を具えたことを特徴とする請求項１０記載の方法。
【請求項１４】
ブレークポイントに遭遇すると、前記ブレークポイントを含むオブジェクトモデル構成要素を中断する工程をさらに具えたことを特徴とする請求項１０記載の方法。
【請求項１５】
ランタイムの問題が存在するかどうかを判別するために、中断中に前記オブジェクトモデル構成要素を分析する工程をさらに具えたことを特徴とする請求項１４記載の方法。
【請求項１６】
中断されたオブジェクトモデル構成要素を再開する工程をさらに具えたことを特徴とする請求項１４記載の方法。
【請求項１７】
コンピュータに指示するためのコンピュータ読取り可能命令を有するコンピュータ読取り可能媒体であって、
オブジェクトモデルの１つまたは複数の構成要素と通信するインターフェースマネージャを提供する工程と、
ブレークポイントの場所を決定する工程と、
前記ブレークポイントに遭遇するようにランタイムを実行する工程と
を具えたことを特徴とするコンピュータ読取り可能媒体。
【請求項１８】
プラグ可能構成要素においてブレークポイントをデバッグするためのシステムであって、
構成要素の挙動を観察するためおよび構成要素を制御するために、ランタイム中に前記プラグ可能構成要素と通信することが可能なデバッグインターフェースと、
ランタイム時にブレークポイントに遭遇すると、前記デバッグインターフェースが、プラグ可能構成要素のオペレーションを観察するために、前記プラグ可能構成要素のオペレーションを中断および／または再開することが可能であるように、前記プラグ可能構成要素内に設定されるブレークポイントと
を具えたことを特徴とするシステム。
【請求項１９】
前記デバッグインターフェースは、前記プラグ可能構成要素内に前記ブレークポイントを設定することを特徴とする請求項１８記載のシステム。
【請求項２０】
前記デバッグインターフェースは、ランタイム時および中断時に前記プラグ可能構成要素の状態を表示することを特徴とする請求項１９記載のシステム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明はデバッガに関し、より具体的には、オブジェクトモデルでのプラグ可能構成要素のブレークポイントのデバッグ（ｄｅｂｕｇｇｉｎｇ　ｂｒｅａｋｐｏｉｎｔ）に関する。
【背景技術】
【０００２】
デバッグには、必ずとは言えないが、通常は、プログラマがランタイムのプログラムの挙動を観察し、意味エラーの場所を特定することができるようにする強力なツールである、デバッガの使用が含まれる。言語およびその関連ライブラリに組み込まれた、一定のデバッグ特徴を使用することもできる。多くのプログラマが問題を特定しようとする場合、

第一に、出力機能への呼出しをそのコードに追加することによるデバッグを経験することになる。これはまったく正当なデバッグ技法であるが、いったん問題がみつかって処理されてしまうと、そうした特別な機能の呼出しをコードからすべて除去しなければならなくなる。たとえ単一の呼出しであっても、新しいコードを追加することによってデバッグ中のコードの挙動を変えてしまう場合もあり得る。

【０００３】

これに代わる方法が、問題およびエラーに関するコードを読み取って分析するコンピューティングアプリケーションである、デバッガを使用することである。デバッガを使用するプログラマは、値を出力するために追加の呼出しを挿入する必要なしに、プログラム中の変数のコンテンツを検査する機能が使用できる。さらに、デバッガを使用して、プログラムコードにブレークポイントを挿入し、該当するポイントで実行を中止させることができる。プログラムが中止される（ブレークモード）と、デバッガ機構（たとえば監視ウィンドウ（ｗａｔｃｈ　ｗｉｎｄｏｗ））を使用して、ローカル変数および他の関連データを検査することができる。ブレークモード中は、コンテンツを表示できるだけでなく、変更および／または編集することもできる。

【０００４】

具体的に言えば、ブレークポイントとは、ある一定のポイントでプログラムの実行を一時的に中断するようにデバッガに伝える信号のことである。ブレークポイントで実行が中断されると、プログラムにはブレークモードに入ったことが伝えられる。ブレークモードに入ることは、プログラムの実行を打切るかまたは終了することではない。実行はいつでも再開（続行）可能である。

【０００５】

ブレークモードは、スポーツイベントのタイムアウトのようなものと考えることができる。プレーヤはすべてフィールド上に残ったまま（機能、変数、およびオブジェクトはメモリ内に残ったまま）であるが、その動きや動作は中断される。タイムアウト中は、審判員が彼らの位置や状態を調べて、違反（バグ）を探すことができる。審判員には、タイムアウト中にプレーヤを調整する権限が与えられている。ブレークモード中は、プログラムへの調整が実行できる。たとえば、変数の値を変更することができる。また、実行ポイントは移動可能であり、これによって、次に実行が再開されたときに実行される予定のステートメントが変更される。

【０００６】

ブレークポイントは、所望の場所および時間にプログラムの実行を中断することができる強力なツールを提供するものである。プログラムは、コードを１行ごとまたは命令ごとに進んでいくのではなく、ブレークポイントにヒットするまで実行し、その後デバッグを開始するように設定することができる。これによって、デバッグプロセスは大幅にスピードアップされる。この機能なしで大規模なプログラムをデバッグするのは、事実上不可能であろう。

【発明の開示】

【発明が解決しようとする課題】

【０００７】

多くのプログラム言語は、実行を中断し、プログラムをブレークモードにする、ステートメントまたは構成体（ｃｏｎｓｔｒｕｃｔ）を有する。たとえばＶｉｓｕａｌ　Ｂａｓｉｃは、Ｓｔｏｐステートメントを有する。

【０００８】

ブレークポイントは、プログラムに追加しなければならない実際のソースコードではないため、これらのステートメントとは異なる。ブレークポイントステートメントは、ソースウィンドウに入力されるのではなく、いくつかの一般的なデバッガインターフェースを介して要求され、次にデバッガがブレークポイントを設定する。

【０００９】

プラグ可能構成要素（たとえばデータ変換システム（ＤＴＳ））を有するコンピューテ

ィング環境のランタイム状況では、デバッグはより複雑になる。
【００１０】
具体的に言えば、ランタイムは対抗して働く構成要素のソースコードを持たないことから、プラグ可能構成要素を備えたランタイムは一意のデバッグ問題であるため、デバッグは困難である。
【００１１】
さらに、ランタイムパッケージ（たとえばＤＴＳパッケージ）の実行の多くが、ランタイム外で発生する。
【００１２】
また、タスクはそれら自体の実行を制御するものであり、これは、それらを（たとえばデバッグの目的で）いつ一時停止するかを制御することを意味する。特定のタスクを一時停止するためのランタイムの場合、ランタイム（データファイルの表示および／または実行のために、データファイルに付随するコンピューティングアプリケーション）は、その特定のタスクと協働していることが必要である。
【００１３】
現在の実施においては、ランタイムでのデバッグには対処しているが、ランタイムとタスクの間の協働は対象とされていないため、ランタイムでのプラグ可能構成要素のデバッグには取り組んでいない。
端的に言えば、ランタイムはプラグ可能構成要素の実行をほとんど制御しない。
【００１４】
以下の内容から、従来技術を克服するシステムおよび方法が求められていることを理解されよう。
【課題を解決するための手段】
【００１５】
プラグ可能構成要素上でのブレークポイントのデバッグが提供される。拡張可能なプラグ可能構成要素を有するコンピューティング環境では、エラーの観察、追跡、および解決が可能なように、構成要素のランタイム（たとえば実行）中の様々なポイントでブレークポイントの設定を可能にするためのシステムおよび方法が提供される。
【００１６】
例示的な実施では、コンピューティング環境にはクライアント、ランタイム、およびタスクが含まれる。オペレーションでは、クライアントが実行可能ファイルを作成するか、またはパッケージをランタイムにロードする。ランタイムはタスクを作成し、そのタスクがブレークポイントを作成できるように構成する。
【００１７】
次に、タスクはブレークポイントを作成し、ブレークポイント情報をランタイムに送信する。
【００１８】
次に、クライアントは、ブレークポイントの集まりを列挙することによって、ブレークポイントを実行可能にする。タスクはブレークポイントに遭遇すると、実行を停止し、ランタイムに制御を与える。
【００１９】
ランタイムは、ブレークポイント呼出しをクライアントに転送する。さらに、ランタイムは、こうしたオペレーションをサポートしているすべてのタスク上で、中断呼出しも実行する。クライアントが満足すると、ランタイムと通信することによってパッケージを再開し、次にすべてのタスクの再開をトリガする。
【００２０】
本発明の一態様として、以下、プラグ可能構成要素上でのブレークポイントのデバッグに関するシステムおよび方法について、添付の図面を参照して説明する。
【発明を実施するための最良の形態】
【００２１】

概要

ソフトウェアデバッグは、開発者がコンピュータプログラムから符号化の不具合を除去しようとする際に使用するプロセスである。ソフトウェア開発のデバッグ段階が、開発全体の６０～７０％の時間を占めることは、めずらしくない。実際に、ソフトウェアプロジェクトオーバラン全体の約８０％は、デバッグが原因である。結局のところ、ソフトウェアデバッグのきわめて重要なプロセスは、多くの困難や不確実性に取り囲まれている。これは、エラー検出プロセスの各段階で、不具合が実際に修正されるのか否かはおろか、エラーを見つけて修正するのにどのぐらいの時間がかかるのかを特定することが困難であるためである。ソフトウェアからバグを除去するために、開発者は第一に、問題が存在することを発見し、次いでエラーを分類し、実際に問題がコード内のどこにあるのかを突き止め、最終的に（他の問題を誘導することなく）状況を修復する解決策を作成しなければならない。問題の中には見つかりにくいものもあるため、プログラマがそれらを見つけるために何カ月も、極端な場合には何年も費やす場合もある。開発者は、ソフトウェアデバッグのプロセスを改良および合理化するための方法を常に探し求めている。同時に、エラー検出に使用される技法を自動化することも試みられてきている。

【００２２】

何年にもわたり、デバッグ技術は大幅に改良されてきており、近未来に向かって大幅な発展が続くであろう。

【００２３】

デバッグ技術の研究により、興味深い傾向が見えてきた。デバッグ改革のほとんどは、人間の能力および対話への依存度を削減することに集中してきた。デバッグ技術は、いくつかの段階を通じて開発されてきた。コンピュータ時代の創成期には、実行したプログラムに関する出力をコンピュータに生成させることは、プログラマにとって困難であった。プログラマは、使用したプログラムに関する情報を得るための様々な方法を開発せざるを得なかった。バグを修復するだけでなく、バグを見つけるツールも構築しなければならなかった。スコープやプログラム制御バルブなどのデバイスが、初期のデバッグ技法として使用された。

【００２４】

やがてプログラマは、プログラム内に印刷命令を入れることによって、バグを検出し始めた。このようにしてプログラマは、プログラム経路および重要な変数の値を追跡できるようになった。印刷ステートメントの使用によって、プログラマは、自分専用のデバッグツールを構築するという時間のかかるタスクから解放された。この技法は今でも一般に使用されており、実際に特定の種類の問題には好適である。

【００２５】

印刷ステートメントはデバッグ技法における改良点であったが、それでもなお、プログラマはかなりの時間と努力を要求された。プログラマに求められたものは、一度に１つのプログラム命令を実行し、プログラム内の任意の変数の値を印刷できるツールであった。これは、プログラムを進行させながら実行するものであるため、印刷ステートメントを入れる時点を事前に決定する必要性からプログラマを解放するものであった。このようにして生まれたのが、ランタイムデバッガである。原則として、ランタイムデバッガは自動印刷ステートメント以外の何者でもない。これは、プログラマが、コード内に印刷ステートメントを入れる必要なしに、プログラム経路および変数を追跡できるようにするものである。

【００２６】

今日では、事実上、市場のあらゆるコンパイラはランタイムデバッガに付属して売られている。デバッガは、プログラムのコンパイル時にコンパイラに渡されるスイッチとして実施される。このスイッチは、ほとんどの場合に「－ｇ」スイッチと呼ばれている。スイッチは、ランタイムデバッガを使用して実行できるように十分な情報を実行可能ファイルに組み込むよう、コンパイラに指示する。ランタイムデバッガは、プログラマがエラーを絞り込もうとする際に、ソースの修正および再コンパイルを行うのではなく、単一のコン

パイルでコンパイルおよび実行できるようにするものであるため、印刷ステートメントにとっては大きな改良点であった。ランタイムデバッガは、プログラム内のエラーの検出を容易にするものではあったが、エラーの原因を見つけられるものではなかった。プログラマには、ソフトウェアの不具合を突き止めて修正するためのさらに良いツールが必要であった。

【００２７】

ソフトウェア開発者は、メモリ破損およびメモリリークなどのいくつかのクラスのエラーが、自動的に検出できることを発見した。これはバグの発見プロセスを自動化するものであったため、デバッグ技法を前進させるステップとなった。ツールが開発者にエラーを通知するので、開発者はそれを修正するだけでよかった。自動デバッガにはいくつかの種類がある。最も単純なものは、プログラムにリンク可能な機能の単なるライブラリである。プログラムが実行されるとこれらの機能が呼び出され、デバッガがメモリの破損をチェックする。この状態を見つけると報告する。こうしたツールの弱点は、実際にメモリの破損が発生しているプログラム内のポイントを検出できないことである。これはデバッガが、プログラムが実行するあらゆる命令を監視しておらず、わずかな数のエラーしか検出できないためである。

【００２８】

ランタイムデバッガの次のグループは、ＯＣＩ技術に基づくものである。これらのツールは、コンパイラによって生成されたオブジェクトコードを読み取り、プログラムがリンクされる前に取り付けられる。これらのツールの基本原理は、メモリにアクセスするプロセッサ命令を探すことである。オブジェクトコードでは、メモリにアクセスするどのような命令も破損をチェックするように修正される。これらのツールは、ライブラリ技法に基づくものよりも有用であるが、それでもまだ完全とは言えない。これらのツールはメモリ命令によってトリガされるため、メモリに関するエラーしか検出できない。これらのツールは、動的メモリ内のエラーは検出できるが、スタック上での検出機能は制限されており、静的メモリでは動作しない。ＯＣＩ技術の弱点により、他のタイプのエラーは検出できない。オブジェクトレベルでは、ソースコードに関する多くの重要な情報が永久に失われ、エラーを突き止める一助としては使用できない。これらのツールの他の欠点は、メモリリークの発生を検出できないことである。オブジェクトレベルではポインタと整数が区別できないため、リークが検出できない原因となる。

【００２９】

ランタイムデバッガの第３のグループは、ＳＣＩ技術に基づくものである。このツールはプログラムのソースコードを読み取り、これを分析し、あらゆるプログラム命令がツールの命令の間に入るようにこれを取り付ける。ツールはソースコードを読み取るため、メモリに関するエラーおよび他の多数のクラスのエラーを発見することができる。さらに、メモリ破損エラーの場合、ツールは、ヒープ、静的、およびスタックメモリを含むすべてのメモリセグメントでエラーを検出することができる。このツールの大きな利点は、プログラム内部のポインタを追跡できること、および、リークを追跡してどこで発生したかを示すことができることである。この世代のツールは、今なお進化を続けている。これらのツールは、メモリエラーを探すことに加えて、言語特有のエラーおよびアルゴリズムのエラーを検出することができる。これらのツールは、次の段階の技術開発に基づくものとなろう。

【００３０】

これらすべてのツールに共通の欠点が１つある。それは、依然としてプログラマが、プログラムがコンパイルされた後にランタイムエラーを見つける特別なステップを実行しなければならないことである。ある意味でこのプロセスは、デバッグの石器時代からそれほど変化していないと言える。第１にコードが作成され、その後コードのエラーがチェックされる。この２段階のプロセスは、高水準にのみ依然として存在している。これを１つの段階に統合する手順が求められている。

【００３１】

ブレークポイントは、実行をデバッグ目的で一時的に休止することができる。典型的なソフトウェア開発環境では、アプリケーションが実行されているスレッドを一時停止し、ソースコード内でアプリケーションが停止する場所を様々な状態に基づいて決定することによって、ブレークポイントの挙動を提供する。ランタイムは、以下の３つの理由でこれを実行することができない。１）タスクは構成要素であるため、いったん実行が中止されると参照するソースコードがなくなる。２）ＤＴＳパッケージの実行の多くは、ランタイムの制御外で発生する。タスクはそれ自体の実行を制御する。ＤＴＳランタイムはそれらを停止するために、タスクが実行されているスレッドを一時停止する必要がある。３）タスクはどのような時でも、一時中止のできない、影響を受けやすい状態にある可能性がある。

【００３２】

本明細書に記載されたシステムおよび方法は、ブレークポイントにヒットしたことをタスクがランタイムに通知した場合、ブレークポイントを実現させる（ｅｘｐｏｓｅ）ことおよび正しく応答することによってタスクがランタイムと協働できるようにし、かつ、プラグ可能構成要素がそれらの例示中のいつでもブレークポイントを作成できるようにする、メカニズムを提供することによって、既存の実施の欠点を改善することを目的とする。言い換えれば、デバッガは、拡張可能構成要素と協働して、ランタイム時の問題の追跡および解決に使用できるブレークポイントを設定する。

【００３３】

本明細書に記載されたシステムおよび方法は、プラグ可能構成要素でのブレークポイントのデバッグに関連して説明しているが、デバッグ技法は、提示した例の範囲を超えた様々な方法で使用できるものであることを理解されよう。

【００３４】

Ａ．例示的なコンピューティング環境

図１は、本発明が実施できる好適なコンピューティングシステム環境１００の一例を示す図である。コンピューティングシステム環境１００は好適なコンピューティング環境の一例に過ぎず、本発明の使用または機能の範囲に関していかなる制限も示唆することを意図するものではない。また、コンピューティング環境１００は、例示的なオペレーティング環境１００に示された構成要素のいずれか１つまたはいずれかの組合せに関して、どのような依存性または要件をも有するものとして解釈されるべきではない。

【００３５】

本発明は、多くの他の汎用または特定用途向けのコンピューティングシステム環境または構成で動作可能である。本発明と共に使用するのに好適な場合がある、よく知られたコンピューティングシステム、環境、および／または構成の例には、パーソナルコンピュータ、サーバコンピュータ、ハンドヘルドまたはラップトップデバイス、マルチプロセッサシステム、マイクロプロセッサベースシステム、セットトップボックス、プログラム可能大衆消費電子製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、上記システムまたはデバイスのうちのいずれかを含む分散コンピューティング環境、およびその他が含まれるが、これらに限定されるものではない。

【００３６】

本発明は、コンピュータによって実行される、プログラムモジュールなどのコンピュータ実行可能命令の一般的な状況で説明することができる。一般に、プログラムモジュールには、特定のタスクを実行するかまたは特定の抽象データ型を実施する、ルーチン、プログラム、オブジェクト、構成要素、データストラクチャなどが含まれる。本発明は、通信ネットワークまたは他のデータ伝送媒体を介してリンクされたリモート処理デバイスによってタスクが実行される、分散コンピューティング環境でも実施可能である。分散コンピューティング環境では、プログラムモジュールおよび他のデータを、メモリ記憶デバイスを含むローカルおよびリモートの両方のコンピュータ記憶媒体内に配置することができる。

【００３７】

図１を参照すると、本発明を実施するための例示的システムは、コンピュータ１１０の形の汎用コンピューティングデバイスを含む。コンピュータ１１０の構成要素は、処理ユニット１２０と、システムメモリ１３０と、システムメモリを含む様々なシステム構成要素を処理ユニット１２０に結合するシステムバス１２１とを含むことができるが、これらに限定されるものではない。システムバス１２１は、メモリバスまたはメモリコントローラ、周辺バス、および様々なバスアーキテクチャのうちのいずれかを使用するローカルバスを含む、いくつかのタイプのバスストラクチャのうちのいずれかであってよい。例を挙げると、こうしたアーキテクチャには、Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ（ＩＳＡ）バス、Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ（ＭＣＡ）バス、拡張ＩＳＡ（ＥＩＳＡ）バス、Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ（ＶＥＳＡ）ローカルバス、およびＰｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ（ＰＣＩ）バス（メザニンバスとも呼ばれる）が含まれるが、これらに限定されるものではない。

【００３８】

コンピュータ１１０は、通常、様々なコンピュータ読取り可能媒体を含む。コンピュータ読取り可能媒体は、コンピュータ１１０によってアクセス可能な任意の使用可能媒体であってよく、揮発性および不揮発性の媒体、取外し可能および取外し不能の媒体の、両方を含む。例を挙げると、コンピュータ読取り可能媒体はコンピュータ記憶媒体および通信媒体を含むことができるが、これらに限定されるものではない。コンピュータ記憶媒体は、コンピュータ読取り可能命令、データストラクチャ、プログラムモジュール、または他のデータなどの、情報を記憶するための任意の方法または技術で実施された、揮発性および不揮発性、取外し可能および取外し不能の、両方の媒体を含む。コンピュータ記憶媒体は、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）または他の光ディスクストレージ、磁気カセット、磁気テープ、磁気ディスクストレージまたは他の磁気記憶デバイス、あるいは、所望の情報を記憶するために使用可能かつコンピュータ１１０によってアクセス可能な任意の他の媒体を含むが、これらに限定されるものではない。通信媒体は、典型的には、コンピュータ読取り可能命令、データストラクチャ、プログラムモジュール、または他のデータを、搬送波または他の移送メカニズムなどの変調データ信号で具体化するものであり、任意の情報送達媒体を含む。「変調データ信号」という用語は、その特徴のうちの１つまたは複数を、信号内で情報を符号化するような方式で設定または変更した信号を意味する。例を挙げると、通信媒体は、有線ネットワークまたはダイレクトワイヤード接続などの有線媒体、ならびに、音波、ＲＦ、赤外線、および他の無線媒体などの無線媒体を含むが、これらに限定されるものではない。上記のいずれかの組合せも、コンピュータ読取り可能媒体の範囲に含まれるものとする。

【００３９】

システムメモリ１３０は、ＲＯＭ １３１およびＲＡＭ １３２などの揮発性および／または不揮発性メモリの形のコンピュータ記憶媒体を含む。起動時などにコンピュータ１１０内の要素間での情報転送を助ける基本ルーチンを含む、基本入出力システム１３３（ＢＩＯＳ）は、典型的にはＲＯＭ １３１に格納される。ＲＡＭ １３２は、典型的には、処理ユニット１２０によって即時アクセス可能であり、かつ／またはその上で現在動作中の、データおよび／またはプログラムモジュールを含む。例を挙げると、図１には、オペレーティングシステム１３４、アプリケーションプログラム１３５、他のプログラムモジュール１３６、およびプログラムデータ１３７が示されているが、これらに限定されるものではない。

【００４０】

コンピュータ１１０は、他の取外し可能／取外し不能、揮発性／不揮発性の、コンピュータ記憶媒体も含むことができる。単なる例として、図１には、取外し不能不揮発性磁気媒体からの読取りまたはこれへの書込みを行うハードディスクドライブ１４０と、取外し可能不揮発性磁気ディスク１５２からの読取りまたはこれへの書込みを行う磁気ディスク

ドライブ１５１と、ＣＤ－ＲＯＭまたは他の光媒体などの取外し可能不揮発性光ディスク１５６からの読取りまたはこれへの書込みを行う光ディスクドライブ１５５が示されている。例示的オペレーティング環境で使用可能な、他の取外し可能／取外し不能、揮発性／不揮発性コンピュータ記憶媒体には、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭなどが含まれるが、これらに限定されるものではない。ハードディスクドライブ１４１は、通常、インターフェース１４０などの取外し不能メモリインターフェースを介してシステムバス１２１に接続され、磁気ディスクドライブ１５１および光ディスクドライブ１５５は、通常、インターフェース１５０などの取外し可能メモリインターフェースによってシステムバス１２１に接続される。

【００４１】

上記で論じ図１に示された、ドライブおよびそれらに関連付けられたコンピュータ記憶媒体は、コンピュータ読取り可能命令、データストラクチャ、プログラムモジュール、およびコンピュータ１１０に関する他のデータのストレージを提供する。たとえば図１では、ハードディスクドライブ１４１は、オペレーティングシステム１４４、アプリケーションプログラム１４５、他のプログラムモジュール１４６、およびプログラムデータ１４７を格納するように示されている。これらの構成要素は、オペレーティングシステム１３４、アプリケーションプログラム１３５、他のプログラムモジュール１３６、およびプログラムデータ１３７と同じであるかまたは異なるかの、いずれの場合も可能なことに留意されたい。オペレーティングシステム１４４、アプリケーションプログラム１４５、他のプログラムモジュール１４６、およびプログラムデータ１４７には、少なくとも異なるコピーであることを示すために、異なる番号が与えられている。ユーザは、キーボード１６２、および一般にマウス、トラックボール、またはタッチパッドと呼ばれるポインティングデバイス１６１などの入力デバイスを介して、コンピュータ１１０にコマンドおよび情報を入力することができる。他の入力デバイス（図示せず）は、マイクロフォン、ジョイスティック、ゲームパッド、衛星放送用パラボラアンテナ、スキャナなどを含むことができる。これらおよび他の入力デバイスは、システムバスに結合されたユーザ入力インターフェース１６０を介して処理ユニット１２０に接続されることが多いが、パラレルポート、ゲームポート、またはユニバーサルシリアルバス（ＵＳＢ）などの、他のインターフェースおよびバスストラクチャによって接続することもできる。モニタ１９１または他のタイプのディスプレイデバイスも、ビデオインターフェース１９０などのインターフェースを介して、システムバス１２１に接続される。コンピュータは、モニタに加えて、出力周辺インターフェース１９５を介して接続可能な、スピーカ１９７およびプリンタ１９６などの他の周辺出力デバイスも含むことができる。

【００４２】

コンピュータ１１０は、リモートコンピュータ１８０などの１つまたは複数のリモートコンピュータへの論理接続を使用する、ネットワーク環境で動作可能である。リモートコンピュータ１８０は、パーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピアデバイス、または他の共通ネットワークノードであってよく、図１にはメモリ記憶デバイス１８１のみが示されているが、通常は、コンピュータ１１０に関して上記で述べた要素のうちの多くまたはすべてが含まれる。示された論理接続には、ローカルエリアネットワーク（ＬＡＮ）１７１およびワイドエリアネットワーク（ＷＡＮ）１７３が含まれるが、他のネットワークを含むこともできる。こうしたネットワーキング環境は、オフィス、企業規模コンピュータネットワーク、イントラネット、およびインターネットでよく見られるものである。

【００４３】

ＬＡＮネットワーキング環境で使用される場合、コンピュータ１１０はネットワークインターフェースまたはアダプタ１７０を介してＬＡＮ　１７１に接続される。ＷＡＮネットワーキング環境で使用される場合、コンピュータ１１０は通常、インターネットなどのＷＡＮ　１７３を介した通信を確立するためのモデム１７２または他の手段を含む。モデ

ム１７２は内蔵または外付けであってよく、ユーザ入力インターフェース１６０または他の適切なメカニズムを介してシステムバス１２１に接続することができる。ネットワーク環境では、コンピュータ１１０に関して示されたプログラムモジュールまたはその一部を、リモートメモリ記憶デバイスに格納することができる。例を挙げると、図１では、リモートアプリケーションプログラム１８５がメモリデバイス１８１上に常駐しているように示されているが、これに限られるものではない。示されたネットワーク接続は例示的なものであり、コンピュータ間の通信リンクを確立する他の手段が使用できることを理解されよう。

【００４４】

Ｂ．例示的なネットワークコンピューティング環境

前述のコンピュータ環境１００は、コンピュータネットワークの一部として配置することができる。一般に、コンピュータに関する上記の記述は、ネットワーク環境内に配置されたサーバコンピュータとクライアントコンピュータの両方に適用する。図２は、本発明が使用可能な、ネットワークを介してクライアントコンピュータと通信しているサーバを備えた、例示的なネットワーク環境を示す図である。図２に示されるように、いくつかのサーバ１０ａ、１０ｂなどが、通信ネットワーク１４（ＬＡＮ、ＷＡＮ、イントラネット、インターネット、または他のコンピュータネットワークであってよい）を介して、いくつかのクライアントコンピュータ２０ａ、２０ｂ、２０ｃ、または携帯電話１５、陸線電話１６、および携帯情報端末１７などのコンピューティングデバイスと相互接続される。通信ネットワーク１６０がたとえばインターネットであるネットワーク環境では、サーバ１０は、クライアント２０がハイパーテキスト転送プロトコル（ＨＴＴＰ）または無線アプリケーションプロトコル（ＷＡＰ）などのいくつかの知られたプロトコルのいずれかを介して通信する際に使用する、Ｗｅｂサーバであってよい。各クライアントコンピュータ２０は、サーバ１０へのアクセスを得るために、ブラウザ１８０ａを備えることができる。同様に、様々なデータを表示および受信するために、携帯情報端末１７はブラウザ１８０ｂを備え、携帯電話１５はブラウザ１８０ｃを備えることができる。

【００４５】

オペレーション時に、ユーザ（図示せず）は、プラグ可能構成要素内でブレークポイントのデバッグを実行しているクライアントコンピューティングデバイス上で実行中のコンピューティングアプリケーションと対話することができる。デバッグアクティビティはサーバコンピュータ上に格納され、通信ネットワーク１４を介してクライアントコンピューティングデバイスを通じて協働するユーザに送信することができる。ユーザは、クライアントコンピューティングデバイス上のコンピューティングアプリケーションとインターフェースすることによって、デバッグに従事することができる。これらのトランザクションは、クライアントコンピューティングデバイスによって、処理および格納のためにサーバコンピュータへ送ることができる。サーバコンピュータは、プラグ可能構成要素のデバッグに従事するために、コンピューティングアプリケーションをホストすることができる。

【００４６】

したがって、本発明は、ネットワークにアクセスしこれと対話するためのクライアントコンピューティングデバイスと、クライアントコンピュータと対話するためのサーバコンピュータとを有する、コンピュータネットワーク環境で使用可能である。ただし、本明細書に記載されたシステムおよび方法は、様々なネットワークベースアーキテクチャで実施可能であるため、示された例に限定されるものではない。次に、本明細書に記載されたシステムおよび方法について、現在の例示的な実施を参照しながらより詳細に説明しよう。

【００４７】

Ｃ．ブレークポイントのデバッグ

サーフェス上のＤＴＳ（データ変換サービス）でのデバッグの概念は、他の環境でのそれと同様である。パッケージ作成者は、パッケージの実行を停止し、変数を検査および変更して、実行を続行するための方法を必要とする。プログラマは、ブレークポイントを介して実行を制御し、監視を介して変数の検査および修正を行うことによって、これを実行

する。ＤＴＳでの異なる点は、実行の多くがランタイムの制御の外にあるパッケージで発生することである。ＤＴＳはＴａｓｋ上のインターフェースメソッドへの呼出し間を除き、またＴａｓｋのスレッドの中断などの強力な戦術を使用した、Ｔａｓｋへのアクセスまたはその制御を持つことがない。ブレークポイントを設定できる場所にわたる、より詳細な制御が望まれる。したがって、オブジェクトモデルは、好ましくは何の変数、イベント、およびメソッドをブレークポイントターゲットにできるかを記述するための方法をＴａｓｋに提供する。インターフェース定義では、デバッグインターフェースを実施することになるエンティティを記述するために、総称「オブジェクト」が使用される。デバッグおよびブレークポイント機能を実現させることを望むオブジェクトモデルにおけるＴａｓｋとオブジェクトは、どちらもこれらのインターフェースを使用しなければならないため、これは、熟慮の上でのことである。

【００４８】

例示的な実施では、ユーザはＴａｓｋを有するパッケージを作成する。ランタイム中に問題が発生し、プログラマはなぜＴａｓｋが自分の考えた通りに動作していないのかを理解しようとする。ブレークポイントは、３行を１行に集約する変換のＴａｓｋ上に設定される。協働するユーザインターフェースを使用することによって、プログラマはその変換の実現されたプロパティおよび変数を見ることができる。変換されている行および結果として生じる実行途中の変換を見ることによって、集約に関する論理が損なわれるのを減らすことができる。

【００４９】

第２の例示的実施では、プログラマは、１つのループ内で実行される２つのＴａｓｋを備えたパッケージを有する。第１のＴａｓｋは、第２のＴａｓｋが使用するグローバル変数を設定する。何からの理由で、ループが部分的にしか完了されていないときに第２のＴａｓｋは失敗する。プログラマは、その理由をグローバル変数が無効な値に設定されようとしているからであると考え、疑わしい反復上にあるときにループ上にブレークポイントを設定し、グローバル変数を検査する。プログラマは、疑わしいグローバル変数の値を見つけるまで、その反復の各ステップを実行していく。疑わしい値を見つけると、プログラマはループに入って、第２のＴａｓｋが失敗するのを見る。

【００５０】

一般に、ブレークポイントは、タスクがその実施および使用において課題に直面することがないことを意図して設計される。ランタイムは、どのブレークポイントをタスクが実現しているか、およびクライアントがそれらを実行可能であるかどうか、に関するすべての情報を維持し、タスク作成者がほとんど問題なく作業できるようにする。ランタイム時のすべてのブレークポイントの状態を維持することによって、それが正しく実行されていることを知るという追加の利点が与えられる。

【００５１】

例示的な実施に従ってブレークポイントを実現させるために、タスクはIDTSBreakpointSiteを実施することが好ましい。このインターフェースは、IDTSSuspendから導出する。タスクが第１に作成されると、ランタイムはタスク上のIDTSBreakpointSite.AcceptBreakpointManagerを呼び出し、タスクが実現させて、それらのブレークポイントがクライアントによって実行可能になっているかどうかをチェックしたいブレークポイントをランタイムに通知するために使用できる、ブレークポイントマネージャに移る。典型的には、タスクは、AcceptBreakpointManager呼出し中にいくつかのブレークポイントを作成し、その後、実行中にIsBreakpointEnabledを数回呼び出すことになる。

【００５２】

各ブレークポイントは、タスク定義済みおよびタスク範囲指定済み（ｔａｓｋ－ｓｃｏｐｅｄ）ＩＤによって識別される。ブレークポイントの実現例は、以下の通りである。

```
AcceptBreakpointManager(IDTSBreakpointManager*pManager)
{
m_pManager=pManager;
```

```
m_pManager->CreateBreakpoint(1,"Break when the task is about to do X", NULL);
}

Execute(...)
{
     VARIANT_BOOL isEnabled;
     m_pManager->IsBreakpointEnabled(1,&isEnabled);
}
```

【００５３】

ブレークポイントマネージャは、実行中のブレークポイントの作成および削除などの、より複雑なシナリオをサポートする。さらに、クライアントは、パッケージ内で実現されたすべてのブレークポイントのリスト、またはIDTSPackage上でGetBreakpoints()を呼び出すことによって特に実行可能なもののみのリストを取り出すことができる。集まりの中の各ブレークポイントオブジェクトは、ブレークポイントの記述と、設定されるとタスクを実行中にそのポイントで停止させることになる実行可能フラグとを含む。

【００５４】

図３は、ブレークポイントのデバッグを理解するための、拡張可能構成要素とコンピューティング環境内での通信とを有する、例示的なコンピューティング環境を示すブロック図である。ＭＩＣＲＯＳＯＦＴ（登録商標）によってサポートされているＤＴＳは、こうしたコンピューティング環境の一例である。

【００５５】

多くの組織では、企業の意思決定を改善するために、データを集中化する必要がある。ただしそのデータは、様々なフォーマットで、様々な場所に格納することができる。データ変換サービス（ＤＴＳ）は、異種のソースからのデータを、ＤＴＳの接続性によってサポートされる単一または複数の宛先に抽出、変換、および統合することを可能にする、ツールのセットを提供することによって、この不可欠な業務のニーズに対処する。ＤＴＳパッケージをグラフィカルに構築するためにＤＴＳツールを使用することによって、またはＤＴＳオブジェクトモデルを使ってパッケージをプログラミングすることによって、組織の特化された業務ニーズに合わせたカスタムデータの動きの解決策を作成することができる。

【００５６】

ＤＴＳパッケージとは、ＤＴＳツールを使用するかまたはプログラムに基づくかのいずれかでアセンブルされ、かつ、ＭＩＣＲＯＳＯＦＴ（登録商標）ＳＱＬ Ｓｅｒｖｅｒ（商標）、ＳＱＬ Ｓｅｒｖｅｒ ２０００ Ｍｅｔａ Ｄａｔａ Ｓｅｒｖｉｃｅｓ、構造化ストレージファイル、またはＭｉｃｒｏｓｏｆｔ Ｖｉｓｕａｌ Ｂａｓｉｃ（登録商標）のファイルに保存された、接続、ＤＴＳタスク、ＤＴＳ変換、およびワークフロー制約の、組織化された集まりである。一般に、各パッケージには、パッケージが実行されたときに順次または並列に実行される、１つまたは複数のステップが含まれる。実行されると、パッケージは正しいデータソースに接続し、データおよびデータベースオブジェクトをコピーし、データを変換し、他のユーザまたはプロセスにイベントを通知する。パッケージは、編集、パスワードの保護、実行のスケジューリング、およびバージョンごとの取出しが可能である。

【００５７】

ＤＴＳパッケージと共に、ＤＴＳタスクも存在する。ＤＴＳタスクは、パッケージ内で単一のステップとして実行される機能の離散セットである。各タスクは、実行される作業項目を、データの動きおよびデータの変換プロセスの一部として、または実行されるジョブとして定義する。一般に使用されるＤＴＳタスクの例には、１）データのインポートおよびエクスポート、２）データの変換、３）データベースオブジェクトのコピー、ならびに４）他のユーザおよびパッケージとの間でのメッセージの送信および受信、が含まれる。

【００５８】
ＤＴＳの構成要素およびオペレーションを完了することが、ＤＴＳ変換である。ＤＴＳ変換とは、データが宛先に到達する前に１つのデータに対して適用される、１つまたは複数の機能またはオペレーションのことである。
【００５９】
図３に示されるように、例示的なコンピューティング環境には、クライアント、ランタイム、およびタスクが含まれる。矢印および数字は、クライアント、ランタイム、およびタスクの間の、通信およびオペレーションの順序を示す。１番の矢印によって示される第１の通信はクライアントとランタイムとの間で行われるものであり、クライアントは実行可能ファイルを作成するか、あるいはランタイムにパッケージをロードする。ランタイムはクライアントに応答して、２番の矢印によって示されるようにタスクを作成する。その後タスクは、３番の矢印によって示されるように、適切と思われる場所にブレークポイントを作成し、そのブレークポイントの場所をランタイムに伝達する。実際には、ブレークポイントは、パッケージおよび／または実行可能ファイルが作成された時点でプログラマによって作成される。タスクは、実質上、コンピューティング環境内にブレークポイントを設定し、その実行に備えるだけである。
【００６０】
タスクのブレークポイント設定に応答して、クライアントは、４番の矢印によって示されるように、ブレークポイントの集まりを列挙することでブレークポイントを実行可能にする。この状況では、クライアントはブレークポイントを実行可能にするためにランタイムと通信する。次にクライアントは、５．１番の矢印によって示されるように、ランタイムでパッケージおよび／または実行可能ファイルを実行する。次にランタイムは、５．２番の矢印によって示されるように、実行されたパッケージのタスクを実行する。タスクは実行中に、以前に設定されたブレークポイントに遭遇することがある。この状況では、６番の矢印で示されるように、タスクはブレークポイントとの遭遇をランタイムに伝達する。ランタイムは、７番の矢印で示されるように、ブレークポイント呼出しをクライアントに転送する。これでクライアントは、問題に遭遇したかどうかを判別し、発見された任意の問題の出所を調査するために、ブレークポイントまでの実行を検査する状況になる。クライアントは満足すると、９番の矢印で示されるようにパッケージに関して再開を呼び出す。この呼出しはランタイムに対して実行され、次にランタイムは、１０番の矢印で示されるように、すべてのタスクについて再開を呼び出す。
【００６１】
図４は、プラグ可能構成要素でのブレークポイントのデバッグを理解するために実行されるプロセスを示す図である。図に示されるように、プロセスはブロック４００で開始されてブロック４０５へ進み、クライアントはオブジェクトモデルにおいてパッケージを作成する。次にプロセスはブロック４１０へ進み、ランタイムはタスクを作成する。次にタスクはブロック４１５でブレークポイントを作成する。このブレークポイント作成に応答して、ブロック４２０でクライアントはブレークポイントを実行可能にする。クライアントは４２５でパッケージを実行し、ランタイムはブロック４３０でこのタスクを実行する。タスクは、設定されたブレークポイントにブロック４３５で遭遇し、ブレークポイントとの遭遇はブロック４４０でクライアントに通知される。ランタイムはブロック４４５でオペレーションの中断を呼び出す。満足すると、クライアントはブロック４５０でタスクの再開を呼び出し、ランタイムはブロック４５５でタスクの再開を呼び出す。その後、プロセスはブロック４６０に進んで終了する。
【００６２】
Ｄ．結論
要するに、本明細書に記載されたシステムおよび方法は、プラグ可能構成要素でのブレークポイントのデバッグを提供するものである。ただし、本発明は様々な修正形態および代替の構成が可能であることを理解されよう。本発明を、本明細書に記載された特定の構成に限定する意図はない。反対に、本発明は、本発明の範囲および精神に当てはまるすべ

ての修正形態、代替構成、および均等物をカバーするものと意図される。

【００６３】

本発明は、様々なコンピュータ環境（非無線および無線の両方のコンピュータ環境を含む）、部分的なコンピューティング環境、および実世界環境で実施可能であることも留意されたい。本明細書に記載された様々な技法は、ハードウェアまたはソフトウェア、あるいはその両方の組合せで実施可能である。好ましくは、この技法は、それぞれがプロセッサ、プロセッサによる読取りが可能な記憶媒体（揮発性および不揮発性のメモリおよび／またはストレージ要素を含む）、少なくとも１つの入力デバイス、ならびに少なくとも１つの出力デバイスを含む、プログラム可能コンピュータ上で実行中の、コンピュータプログラム内で実施される。プログラムコードは、前述の機能を実行するためおよび出力情報を生成するために、入力デバイスを使用して入力されたデータに適用される。出力情報は、１つまたは複数の出力デバイスに適用される。各プログラムは、コンピュータシステムと通信するために、高水準の手続き型またはオブジェクト指向のプログラミング言語で実施されることが好ましい。ただしプログラムは、所望であればアセンブリまたはマシン言語で実施することができる。いずれの場合も、言語はコンパイル済みまたは解釈済みの言語であってよい。こうしたコンピュータプログラムは、それぞれ、記憶媒体またはデバイスがコンピュータによって読み取られたときに前述の手順を実行するように、コンピュータを構成および動作するために、汎用または特定用途向けのプログラム可能コンピュータによって読み取り可能な、記憶媒体またはデバイス（たとえばＲＯＭまたは磁気ディスク）上に格納されることが好ましい。さらにシステムは、コンピュータプログラムを使用して構成されたコンピュータ読取り可能記憶媒体として実施されるものとみなすことも可能であり、そのように構成された記憶媒体は、特定の事前に定義された方法でコンピュータを動作させる。

【００６４】

以上、本発明の例示的な実施について詳細に述べてきたが、当業者であれば、この例示的な実施形態では、本発明の新規な教示および利点から著しく逸脱することなしに、多くの追加の修正が可能であることを容易に理解されよう。したがって、これらおよびすべてのこうした修正は、本発明の範囲内に含められるものであることが意図される。本発明は、添付の例示的な特許請求の範囲によって、より適切に定義することができる。

【図面の簡単な説明】

【００６５】

【図１】本発明に好適な例示的コンピューティング環境を示す概略図である。

【図２】例示的なネットワーク化されたコンピューティング環境を示す図である。

【図３】プラグ可能構成要素環境においてブレークポイントを実行する場合に実行される、例示的な処理を示すブロック図である。

【図４】本明細書に記載されたシステムおよび方法に従ってブレークポイントをデバッグする場合に実行される処理を示す流れ図である。

【符号の説明】

【００６６】

１　クライアントは実行可能ファイルを作成するか、またはパッケージをロードする

２　ランタイムはタスクを作成し、IDTSBreakpointSiteでAcceptBreakpointManagerを呼び出す

３　タスクはIDTSBreakpointManagerでCreateBreakpointを呼び出すことによって、ブレークポイントを作成する

４　クライアントはIDTSPackageでブレークポイントの集まりを列挙することによって、ブレークポイントを実行可能にする

５．１　クライアントはパッケージを実行する

５．２　ランタイムはタスクを実行する

６　タスクはそのコード内でブレークポイントに遭遇し、IDTSBreakpointManagerでIsBreakpointEnabledを呼び出す。これが実行可能であれば、タスクはOnBreakpointHitを呼

び出す

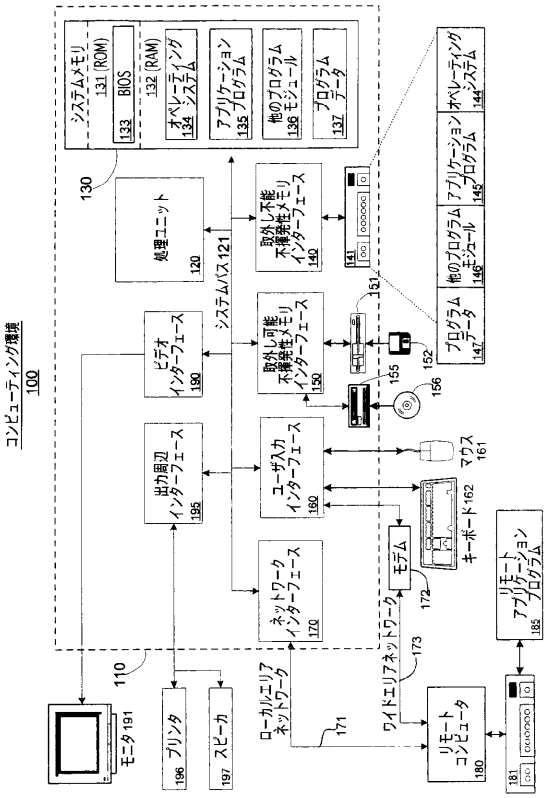
７　ランタイムはOnBreakpointHit呼出しをクライアントに転送する

８　ランタイムはIDTSSuspendを実施するすべてのタスクに関する中断を呼び出す

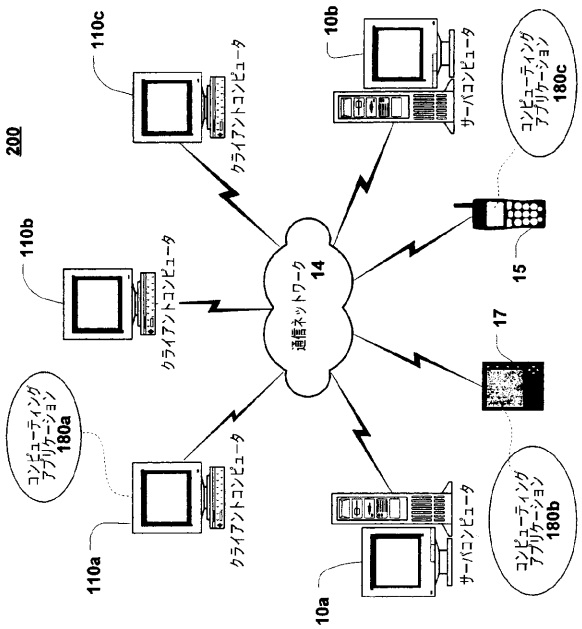
９　クライアントはパッケージに関する再開を呼び出す

１０　ランタイムはすべてのタスクに関する再開を呼び出す

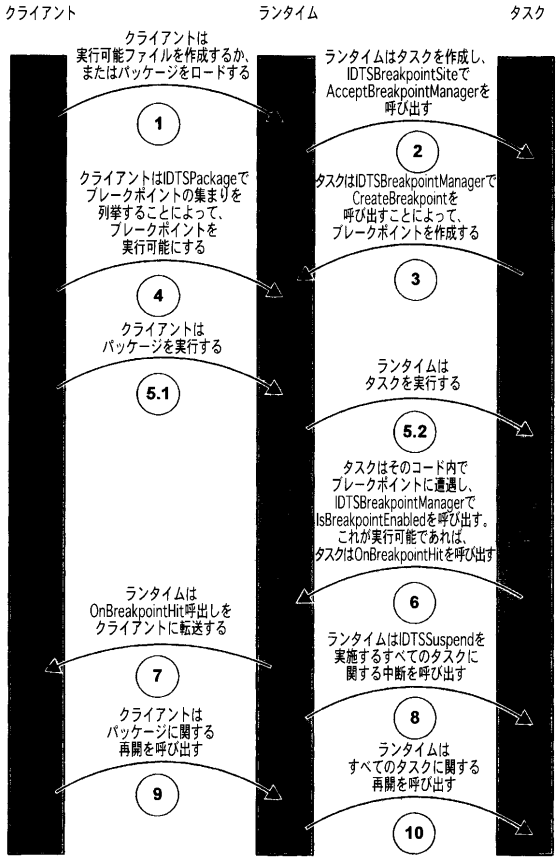
【図１】



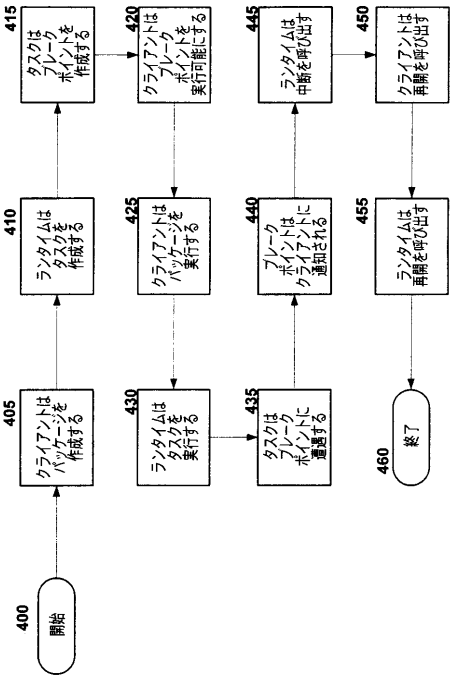
【図２】



【図３】


クライアント
ランタイム
タスク
クライアントは
実行可能ファイルを作成するか、
またはパッケージをロードする
1
ランタイムはタスクを作成し、
IDTSBreakpointSiteで
AcceptBreakpointManagerを
呼び出す
2
クライアントはIDTSPackageで
ブレークポイントの集まりを
列挙することによって、
ブレークポイントを
実行可能にする
4
タスクはIDTSBreakpointManagerで
CreateBreakpointを
呼び出すことによって、
ブレークポイントを作成する
3
クライアントは
パッケージを実行する
5.1
ランタイムは
タスクを実行する
5.2
タスクはそのコード内で
ブレークポイントに遭遇し、
IDTSBreakpointManagerで
IsBreakpointEnabledを呼び出す。
これが実行可能であれば、
タスクはOnBreakpointHitを呼び出す
6
ランタイムは
OnBreakpointHit呼出しを
クライアントに転送する
7
ランタイムはIDTSSuspendを
実施するすべてのタスクに
関する中断を呼び出す
8
クライアントは
パッケージに関する
再開を呼び出す
9
ランタイムは
すべてのタスクに関する
再開を呼び出す
10

【図４】


400
開始
405
クライアントは
パッケージを
作成する
410
ランタイムは
タスクを
作成する
415
タスクは
ブレーク
ポイントを
作成する
420
クライアントは
ブレーク
ポイントを
実行可能にする
425
クライアントは
パッケージを
実行する
430
ランタイムは
タスクを
実行する
435
タスクは
ブレーク
ポイントに
遭遇する
440
ブレーク
ポイントは
クライアントに
通知される
445
ランタイムは
中断を呼び出す
450
クライアントは
再開を呼び出す
455
ランタイムは
再開を呼び出す
460
終了

フロントページの続き

(72)発明者　ブライアン　ジェイ．ハートマン
アメリカ合衆国　９８００７　ワシントン州　ベルビュー　サウスイースト　１４２　プレイス
２９６９　アパートメント　ナンバー９

Ｆターム(参考) 5B042 GA08 HH25 LA00