(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6195849号
(P6195849)
(45)発行日　平成29年9月13日(2017.9.13)　(24)登録日　平成29年8月25日(2017.8.25)

(51)Int.Cl.　　F I
G06F 9/44 (2006.01)　　G06F 9/06 620A
G06F 9/445 (2006.01)　　G06F 9/06 640A

請求項の数 13　(全 22 頁)

(21)出願番号　特願2014-557678 (P2014-557678)
(86)(22)出願日　平成25年2月4日 (2013.2.4)
(65)公表番号　特表2015-507310 (P2015-507310A)
(43)公表日　平成27年3月5日 (2015.3.5)
(86)国際出願番号　PCT/US2013/024559
(87)国際公開番号　WO2013/122758
(87)国際公開日　平成25年8月22日 (2013.8.22)
審査請求日　平成28年2月2日 (2016.2.2)
(31)優先権主張番号　13/371,479
(32)優先日　平成24年2月13日 (2012.2.13)
(33)優先権主張国　米国 (US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
ＭＩＣＲＯＳＯＦＴ　ＣＯＲＰＯＲＡＴＩＯＮ
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
One Microsoft Way, Redmond WA 98052-5321 United State of America
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
最終頁に続く

(54)【発明の名称】ソフトウェア・コードの生成およびキャッシング

(57)【特許請求の範囲】
【請求項１】
少なくとも部分的にコンピューターによりインプリメントされる方法であって、
ソース・コードを含むソフトウェアを実行するリクエストを受け取るステップであって、前記ソース・コードは、スクリプトファイルへの明確な参照に遭遇していないウェブサイトのページに対するスクリプトファイルである、ステップと、
前記リクエストに応答して、第２のコードが既に前記ソース・コードから生成され、不揮発性ストレージに格納されたかを判定するステップであって、前記第２のコードが既に前記ソース・コードから生成され、不揮発性ストレージに格納されているかどうかを判定するステップは、前記不揮発性ストレージの既知の場所において前記第２のコードを検査することを含み、前記既知の場所は前記ソース・コードのソースの場所を識別する参照から導き出され、前記ソースの場所へはネットワークを介して到達可能である、ステップと、
前記リクエストを受け取る前に前記第２のコードが既に生成されていた場合に、
前記第２のコードを得るステップと
前記第２のコード又はそこから導き出されたコードを実行するステップを含む、第１組のアクションを実行するステップと、
前記リクエストを受け取る前に前記第２のコードがまだ生成されていない場合に、
前記ソース・コードを得るステップと、
前記ソース・コードから第２のコードを生成するステップと、

前記第２のコード又はそこから導き出されたコードを実行するステップと
を含む、第２組のアクションを実行するステップと、
前記リクエストを受け取る前に前記第２のコードがまだ生成されていない場合に、
前記第２のコードを、前記ソフトウェアを後に実行する際に使用するために、前記不揮発性ストレージに格納するステップと、
前記第２のコードを含むファイルをメモリ・マッピングし、該メモリ・マッピングを介して複数のプロセスで前記ファイルを共有するステップと、
を含む第３組のアクションを実行するステップと
を含む、方法。
【請求項２】
前記第３組のアクションを実行するステップは、
前記ソース・コードを再び得るステップと、
前記ソース・コードから前記第２のコードを再び生成するステップと
をさらに含む、請求項１に記載の方法。
【請求項３】
前記第２のコードが既に前記ソース・コードから生成され、不揮発性ストレージに格納されたかを判定するステップは、
前記不揮発性ストレージの既知の場所で前記第２のコードを検査するステップをさらに含み、前記ソース・コードを含むパッケージは前記不揮発性ストレージをホストするターゲット・システムにインストールされている、請求項１に記載の方法。
【請求項４】
前記第２のコードが既に前記ソース・コードから生成され、不揮発性ストレージに格納されたかを判定するステップは、
前記不揮発性ストレージからデータを得るステップをさらに含み、前記データは前記ソース・コードを含むパッケージに対するバイトコードを前記パッケージに対するソース・コードと並べて配するファイルの一部であり、前記データは前記第２のコードが既に生成されてファイルに格納されているかどうかを示す、請求項１に記載の方法。
【請求項５】
前記第２のコードが、生成された後に変更されたかを検査するステップをさらに含み、そうである場合に、
前記ソース・コードを再び得るステップと、
前記ソース・コードから前記第２のコードを再び生成するステップと、
前記第２のコードを、前記ソフトウェアを後に実行する際に使用するために、前記不揮発性ストレージに格納するステップと、
を含む請求項１に記載の方法。
【請求項６】
計算環境におけるシステムであって、
メモリに結合された処理ユニットを備えるコンピューターを備え、前記メモリは、
ソフトウェアのソース・コードを含むパッケージのデータを格納するように動作するストアであって、前記ソース・コードはスクリプトファイルへの明確な参照に遭遇していないウェブサイトのページに対するスクリプトファイルである、ストアと、
前記ストアに前記パッケージをインストールするように動作するインストーラーであって、前記ソース・コードが第２のコードへとコンパイルされることを示すようにデータ構造をアップデートするように更に動作するインストーラーと、
前記ソース・コードを識別するために前記データ構造を検査するように、及び前記ソース・コードが前記第２のコードへとコンパイルされることを、前記データ構造において示すこと基づいて、前記ソース・コードを前記第２のコードへとコンパイルするように動作するコード・ジェネレーターと、
前記第２のコードを前記ストアへ永続的に格納するように、および前記ストアの前記第２のコードへのアクセスを提供するように動作するキャッシュ・マネージャーと、

前記ソフトウェアを実行するリクエストを受け取るように動作し、且つ前記コード・ジェネレーターが前記ソース・コードを前記第２のコードへと既にコンパイルしているかどうかを判定するように動作する実行マネージャーであって、前記コード・ジェネレーターが前記ソース・コードを前記第２のコードへと既にコンパイルしていると判定することは、不揮発性ストレージの既知の場所で前記第２のコードを検査することを含み、前記既知の場所は前記ソース・コードのソースの場所を識別する参照から導き出され、前記ソースの場所はネットワークを介して到達可能であり、

コンパイルしていた場合、

前記第２のコードを得ることと、

前記第２のコード又はそれから導き出されたコードを実行することと

を含むアクションを行い、

コンパイルしていない場合、

前記ソース・コードを得ることと、

前記ソース・コードが前記第２のコードへとコンパイルされるようにすることと、

前記ソース・コード又はそれから導き出されたコードを実行することと

を含むアクションを行う

実行マネージャーと

を含むシステム。

【請求項７】

前記実行マネージャーは、さらに前記リクエストが受け取られる前に前記コード・ジェネレーターがまだ前記ソース・コードを前記第２のコードへコンパイルしていない場合に、遅滞なく前記第２のコードを生成させるように動作する、請求項６に記載のシステム。

【請求項８】

前記コード・ジェネレーターは、前記システムが他のパッケージをインストールしていないとき又は前記システムの処理帯域を消費する他のタスクを実行していないときに行うプロセスとして実行される、請求項６に記載のシステム。

【請求項９】

前記コード・ジェネレーターは、１日のうちの構成可能な時間に行うプロセスとして実行される、請求項６に記載のシステム。

【請求項１０】

デバイスであって、

処理ユニットと、

ソフトウェアのソース・コードを含むパッケージのデータを格納するよう構成されたストアであって、前記ソース・コードは、スクリプトファイルへの明確な参照に遭遇していないウェブサイトのページに対するスクリプトファイルである、ストアと、

前記パッケージを前記ストアにインストールするよう構成されたインストーラーであって、該インストーラーは、さらに前記ソース・コードが第２のコードへとコンパイルされることを示すようにデータ構造をアップデートするよう動作する、インストーラーと、

前記ソース・コードを識別するために前記データ構造を検査するように、及び前記ソース・コードが前記第２のコードへとコンパイルされることを前記データ構造において示すことに基づいて、前記ソース・コードを前記第２のコードへとコンパイルするように構成されたコード・ジェネレーターと、

前記第２のコードを前記ストアへ永続的に格納するように、および前記ストアの前記第２のコードへのアクセスを提供するように構成されたキャッシュ・マネージャーと、

前記ソフトウェアを実行するリクエストを受け取るように構成され、且つ前記コード・ジェネレーターが前記ソース・コードを前記第２のコードへと既にコンパイルしたかどうかを判定するように構成された実行マネージャーであって、前記コード・ジェネレーターが前記ソース・コードを前記第２のコードへと既にコンパイルしたかを判定することは、不揮発性ストレージの既知の場所で前記第２のコードを検査することを含み、前記既知の

場所は前記ソース・コードのソースの場所を識別する参照から導き出され、前記ソースの場所はネットワークを介して到達可能であり、
コンパイルしていた場合、
前記第２のコードを得ることと、
前記第２のコード又はそれから導き出されたコードを実行することと
を含むアクションを行い、
コンパイルしていない場合、
前記ソース・コードを得ることと、
前記ソース・コードが前記第２のコードへとコンパイルされるようにすることと、
前記ソース・コード又はそれから導き出されたコードを実行することと
を含むアクションを行う
実行マネージャーと
を含むデバイス。
【請求項１１】
前記リクエストを受け取る前に前記コード・ジェネレーターが前記ソース・コードを前記第２のコードへとまだコンパイルしていない場合に、前記実行マネージャーは、遅延なく前記第２のコードを生成させるようにさらに構成される、請求項１０に記載のデバイス。
【請求項１２】
前記デバイスが他のパッケージをインストールしていないとき又は前記デバイスの処理帯域を消費する他のタスクを実行していないときに行うプロセスとして、前記コード・ジェネレーターが実行される、請求項１０に記載のデバイス。
【請求項１３】
前記コード・ジェネレーターは、１日のうちの構成可能な時間に行うプロセスとして実行される、請求項１０に記載のデバイス。
【発明の詳細な説明】
【背景技術】
【０００１】
[0001] スクリプト言語は、様々な環境で見つけることができる。例えば、多くのインターネット・ブラウザーは、ユーザー入力または他のデータに基づいてウェブ・ページがその挙動をカスタム化することを可能にするスクリプト言語を有する。スクリプト言語はまた、インターネット・ブラウザーの外の環境でも見つけることができる。スクリプト言語と関連する１つの問題は、特に、スタート・アップのとき、また、可能性としては、それぞれの実行のときに、スクリプト言語は、コンパイルされている従来のプログラムと比較して遅くなり得ることである。
【０００２】
[0002] ここで特許請求される主題事項は、何れかの欠点を解決する実施形態には限定されず、また、上記で説明した環境などのみで動作する実施形態にも限定されない。むしろ、この背景は、単に、ここで説明される幾つかの実施形態を実施でき得る１つの例示的な技術範囲を例示するために提供されている。
【発明の概要】
【０００３】
[0003] 簡潔には、ここで説明する主題事項の構成は、ソフトウェア・コードの生成およびキャッシングに関する。構成では、ターゲット・デバイスは、インストールするソフトウェアを受け取ることができる。ソフトウェアは、コンパイルされていないソース・コードを含み得る。ターゲット・デバイスは、ソフトウェアをインストールすることができ、且つパッケージのソース・コードが、永続的に格納される中間または実行可能なコードへとコンパイルされることを、示すことができる。ターゲット・デバイスが、ソフトウェアがコンパイルされる前にそのソフトウェアの実行を要求するリクエストを受け取った場

合、ターゲット・デバイスは、遅れることなくソフトウェアをコンパイルして実行することができる。ターゲット・デバイスが、ソフトウェアがコンパイルされた後にそのソフトウェアの実行を要求するリクエストを受け取った場合、ターゲット・デバイスは、コンパイルされたコードを入手して実行することができる。上記の動作はまた、ターゲット・デバイスから離れたサーバーから得られるスクリプト・コードにも、適用することができる。

【０００４】

[0004] この概要は、後の詳細な説明で更に記載する主題事項の幾つかの構成を簡単に示すために提供される。この概要は、特許請求の範囲に記載の主題事項の鍵となる特徴や本質的な特徴を特定することを意図しておらず、また、特許請求の範囲に記載の主題事項の範囲を決定する際に用いることを意図していない。

【０００５】

[0005] 「ここで説明する主題事項」という記載は、内容が明らかに別の事項を示さないかぎり、詳細な説明に記載される主題事項を指す。「構成」という用語は、「少なくとも１つの構成」と解釈すべきである。詳細な説明に記載される主題事項の示される構成は、特許請求の範囲に記載の主題事項の鍵となる特徴や本質的な特徴を特定することを意図していない。

【０００６】

[0006] 主題事項の上記の構成およびここで説明する他の構成は、例として示され、添付の図面のものに限定されない。図面では、同じ参照番号は類似のエレメントを示す。

【図面の簡単な説明】

【０００７】

【図１】図１は、ここで説明する主題事項の構成を組み込むことができる例示の汎用計算環境を表すブロック図である。

【図２】図２は、ここで説明する主題事項の構成を動作させることができるシステムのコンポーネントの例示の構成を表すブロック図である。

【図３】図３は、ここで説明する主題事項の構成に従って使用できる例示のデータ構造を示す。

【図４】図４は、ここで説明する主題事項の構成を動作させることができる環境のコンポーネントの例示の構成を表すブロック図である。

【図５】図５および図６は、ここで説明する主題事項の構成に従って生じ得る例示のアクションを概略的に表すフロー図である。

【図６】図５および図６は、ここで説明する主題事項の構成に従って生じ得る例示のアクションを概略的に表すフロー図である。

【発明を実施するための形態】

【０００８】

定義

[0012] 「含む」という用語およびその異形は、制限のない用語（open-ended term）であり、「含むが、それに限定されない」という意味である。「または」という用語は、内容により明らかに要求されないかぎり、「および／または」と解釈する。「基づく」という用語は、「少なくとも部分的に基づく」と解釈する。「１つの実施形態」および「実施形態」という用語は、「少なくとも１つの実施形態」と解釈する。「別の実施形態」という用語は、「少なくとも１つの別の実施形態」と解釈する。

【０００９】

[0013] 「１つ（a）」、「１つ（an）」、および「その（the）」という用語は、示されるアイテムやアクションが１以上であることを含む。特に、特許請求の範囲では、アイテムへの参照は、一般に、少なくとも１つのそのようなアイテムが存在することを意味し、アクションへの参照は、アクションの少なくとも１つのインスタンスが行われることを意味する。

【００１０】

[0014] ここでは、「第１」、「第２」、「第３」などの用語が時には用いられる。追加のコンテキストが無い場合、特許請求の範囲におけるこれらの用語の使用は、順を暗示することを意図しておらず、識別するために用いている。例えば、「第１のバージョン」および「第２のバージョン」というフレーズは、第１のバージョンが最初のバージョンであることや、第２のバージョンの前に作成されたことや、第２のバージョンの前に第１のバージョンが要求されることや動作させられることを、必ずしも意味しない。これらのフレーズは、異なるバージョンを識別するために用いられる。

【００１１】

[0015] 見出しは、利便性のためのみのものであり、１つのトピックにおける情報は、そのトピックを示す見出しを持つセクション外でも見つけることができる。

[0016] 明示的または暗黙的な他の定義は下記に含まれているであろう。

【００１２】

例示の動作環境

[0017] 図１は、ここで説明する主題事項の構成を実施できる適切な計算システム環境１００の例を示す。計算システム環境１００は、適切な計算システム環境の単なる一例であり、ここで説明する主題事項の構成の使用や機能の範囲を制限することを意図するものでも提案するものでもない。計算環境１００は、例示的な動作環境１００に示されたコンポーネントの何れかのものやコンポーネントの組み合わせと関連する依存性や必要条件を持たないことを理解すべきである。

【００１３】

[0018] ここで説明する主題事項の構成は、他の様々な汎用や特定用途向けの計算システムの環境または構成で動作することができる。ここで説明する主題事項の構成を用いるのに適切な良く知られた計算システム、環境、または構成の例は、パーソナル・コンピューター、サーバ・コンピューター、手持型またはラップトップ型のデバイス、マルチプロセッサー・システム、マイクロコントローラー・ベースのシステム、セットトップ・ボックス、プログラマブルの大衆消費電子製品、ネットワークＰＣ、ミニコンピューター、メインフレーム・コンピューター、パーソナル・デジタル・アシスタント（ＰＤＡ）、ゲーム用デバイス、プリンター、セットトップ・ボックスを含むアプライアンス、メディア・センター、または他のアプライアンス、自動車への埋め込み型または取り付け型の計算デバイス、他のモバイル・デバイス、上記のシステムやデバイスの何れかを含む分散型計算環境などを含む。

【００１４】

[0019] ここで説明する主題事項の構成は、コンピューターにより実行されているプログラム・モジュールなどのようなコンピューター実行可能命令の一般的コンテキストで説明することができる。一般に、プログラム・モジュールは、ルーチン、プログラム、オブジェクト、コンポーネント、データ構造などを含み、それらは、特定のタスクを行うか、または特定の抽象データ型を実現する。ここで説明する主題事項の構成はまた、通信ネットワークを通じてリンクされたリモート処理デバイスによりタスクが行われる分散型計算環境でも実施することができる。分散型計算環境では、プログラム・モジュールは、メモリ・ストレージ・デバイスを含むローカルおよびリモートのコンピューター・ストレージ媒体に配される。

【００１５】

[0020] 図１を参照すると、ここで説明する主題事項の構成をインプリメントするための例示のシステムは、コンピューター１１０の形の汎用計算デバイスを含む。コンピューターは、命令を実行できる任意の電子デバイスを含むことができる。コンピューター１１０のコンポーネントは、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・コンポーネントを処理ユニット１２０と結合するシステム・バス１２１とを含む。システム・バス１２１は、様々なバス構造のうちの任意のものを用いるメモリ・バスまたはメモリ・コントローラー、周辺バス、およびローカル・バスを含む幾つかのタイプのバス構造のうちの、任意のものとすることができる。例として、そ

のようなアーキテクチャーは、業界標準アーキテクチャー（ＩＳＡ）バス、マイクロ・チャネル・アーキテクチャー（ＭＣＡ）バス、エンハンストＩＳＡ（ＥＩＳＡ）バス、ビデオ・エレクトロニクス・スタンダーズ・アソシエーション（ＶＥＳＡ）ローカル・バス、およびメザニン・バスとしても知られているペリフェラル・コンポーネント・インターコネクト（ＰＣＩ）バス、アドバンスト・グラフィックス・ポート（ＡＧＰ）、およびＰＣＩエクスプレイ（ＰＣＩｅ）を含むが、これらには限定されない。

【００１６】

[0021] 処理ユニット１２０は、ハードウェア・セキュリティ・デバイス１２２と接続することができる。セキュリティ・デバイス１２２は、コンピューター１１０の様々な構成を守るために用いることができる暗号鍵を格納すること及び生成することができる。１つの実施形態では、セキュリティ・デバイス１２２は、トラステッド・プラットフォーム・モジュール（ＴＰＦ）チップ、ＴＰＭセキュリティ・デバイスなどを含むことができる。

【００１７】

[0022] コンピューター１１０は、典型的には、様々なコンピューター可読媒体を含む。コンピューター可読媒体は、コンピューター１１０によりアクセス可能な任意の適切な媒体とすることができ、揮発性および不揮発性、取り外し可能および取り外し不可能な媒体の双方を含む。限定ではなく例として、コンピューター可読媒体は、コンピューター・ストレージ媒体および通信媒体を含む。

【００１８】

[0023] コンピューター・ストレージ媒体は、コンピューター可読命令、データ構造、プログラム・モジュール、または他のデータなどのような情報を格納するために任意の方法や技術で実装される揮発性および不揮発性、取り外し可能および取り外し不可能の双方の媒体を含む。コンピューター・ストレージ媒体は、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、ソリッド・ステート・ストレージ、フラッシュ・メモリ、または他のメモリ技術、ＣＤ－ＲＯＭ、デジタル・バーサタイル・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装置、または望まれる情報を格納するために使用でき且つコンピューター１１０によりアクセスできる他の任意の媒体を含む。

【００１９】

[0024] 通信媒体は、典型的には、搬送波などのような変調されたデータ信号または他のトランスポート機構においてコンピューター可読命令、データ構造、プログラム・モジュール、または他のデータを実現するものであり、任意の情報配信媒体を含む。「変調されたデータ信号」という用語は、信号内で情報をエンコードするように、その信号の特性のうちの１以上の特性が設定または変更された信号を意味する。限定ではなく例として、通信媒体は、有線ネットワークや直接有線接続などのような有線媒体と、音響、ＲＦ、赤外線、および他のワイヤレス媒体などのようなワイヤレス媒体とを含む。上記のものの任意のものの組み合わせも、コンピューター可読媒体の範囲内に含まれるべきである。

【００２０】

[0025] システム・メモリ１３０は、リード・オンリー・メモリ（ＲＯＭ）１３１およびランダム・アクセス・メモリ（ＲＡＭ）１３２などのような、揮発性および／または不揮発性のメモリの形態のコンピューター・ストレージ媒体を含むことができる。スタート・アップの時などにコンピューター１１０内のエレメント間での情報の転送を手助けする基本的ルーチンを含むベーシック入出力システム（ＢＩＯＳ）１３３は、典型的にはＲＯＭ１３１に格納される。ＲＡＭ１３２は、典型的には、処理ユニット１２０により即座にアクセス可能な及び／又は現在動作させられているデータおよび／またはプログラム・モジュールを含む。限定ではなく例として、図１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を示す。

【００２１】

[0026] コンピューター１１０はまた、他の取り外し可能／取り外し不可能な揮発性／不揮発性のコンピューター可読媒体を含むことができる。単なる例として、図１は、取り外し不可能で不揮発性の磁気媒体に対する読み出しおよび書き込みを行うためのハード・ディスク・ドライブ１４１と、取り外し可能で不揮発性の磁気ディスク１５２に対する読み出しおよび書き込みを行うための磁気ディスク・ドライブ１５１と、ＣＤ－ＲＯＭや他の光媒体などのような取り外し可能で不揮発性の光ディスク１５６に対する読み出しおよび書き込みを行うための光ディスク・ドライブ１５５とを示す。例示的な動作環境で用いることができる他の取り外し可能／取り外し不可能で揮発性／不揮発性のコンピューター・ストレージ媒体は、磁気テープ・カセット、フラッシュ・メモリ・カード、および他のソリッド・ステート・ストレージ・デバイス、デジタル・バーサタイル・ディスク、他の光ディスク、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などを含む。ハード・ディスク・ドライブ１４１は、インターフェース１４０を介してシステム・バス１２１へ接続することができ、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、インターフェース１５０などのような取り外し可能で不揮発性のメモリのためのインターフェースにより、システム・バス１２１へ接続することができる。

【００２２】

[0027] ドライバーと、それらと関連するものであり上記で説明し且つ図１で示したコンピューター・ストレージ媒体とは、コンピューター可読命令、データ構造、プログラム・モジュール、およびコンピューター１１０に対する他のデータのストレージを提供する。例えば、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を格納するものとして示されている。これらのコンポーネントは、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と同じである場合も異なる場合もあることに留意されたい。ここでは、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は、少なくとも、別のコピーであることを示すために、異なる番号が付されている。

【００２３】

[0028] ユーザーは、キーボード１６２や、マウスやトラックボールやタッチ・パッドなどと一般的に呼ばれるポインティング・デバイス１６１などのような入力デバイスを用いて、コンピューター１１０へ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（示さず）としては、マイクロフォン、ジョイスティック、ゲーム・パッド、サテライト・ディッシュ、スキャナー、接触感応型スクリーン、書き込みタブレット、ジェスチャー・キャプチャ・デバイスなどが含まれ得る。これらおよび他の入力デバイスは、システム・バスと結合されたユーザー入力インターフェース１６０を通じて処理ユニット１２０と接続されることが多いが、他のインターフェースおよびバス構造、例えば、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）などにより接続されることもできる。

【００２４】

[0029] モニター１９１や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などのようなインターフェースを介してシステム・バス１２１と接続される。コンピューターは、モニターに加えて、出力用周辺機器インターフェース１９５を介して接続できるスピーカー１９７やプリンター１９６などのような他の周辺出力デバイスを含むこともできる。

【００２５】

[0030] コンピューター１１０は、ネットワーク化された環境で、リモート・コンピューター１８０などのような１以上のリモート・コンピューターへの論理接続を用いて、動作することができる。リモート・コンピューター１８０は、パーソナル・コンピューター

、サーバー、ルーター、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・ノードとすることができ、図１ではメモリ・ストレージ・デバイス１８１のみを示しているが、典型的には、コンピューター１１０と関連して説明した多くのまたは全てのエレメントを含むことができる。図１に示す論理接続は、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を含むが、他のネットワークを含むこともできる。そのようなネットワーク環境は、オフィス、エンタープライズ・ワイド・コンピューター・ネットワーク、イントラネット、およびインターネットにおいて一般的である。

【００２６】

[0031] コンピューター１１０は、ＬＡＮネットワーク環境で用いる場合、ネットワーク・インターフェースまたはアダプター１７０を通じてＬＡＮ１７１と接続される。コンピューター１１０は、ＷＡＮネットワーク環境で用いる場合、モデム１７２、またはインターネットなどのようなワイド・エリア・ネットワーク１７３を通じての接続を確立するための他の手段を含む。内部または外部に備えられ得るモデム１７２は、ユーザー入力インターフェース１６０または他の適切な機構を介してシステム・バス１２１と接続することができる。ネットワーク化された環境では、コンピューター１１０と関連して示されているプログラム・モジュールまたはその一部は、リモート・メモリ・ストレージ・デバイスに格納することができる。限定ではない例として、図１は、メモリ・ストレージ・デバイス１８１に存在するものとしてリモート・アプリケーション・プログラム１８５を示す。示されたネットワーク接続が例示であり、コンピューター間の通信リンクを確立するために別の手段も使用できることは、理解できる。

【００２７】

コードの生成およびキャッシング

[0032] 先に述べたように、コンパイルされている従来のプログラムと比較して、スクリプト言語は、特に最初の始動時には、ユーザーからは遅く見え得る。

【００２８】

[0033] 図２は、ここで説明する主題事項の構成を動作させることができるシステムのコンポーネントの例示の構成を表すブロック図である。図２に示すコンポーネントは、例であり、必要とされ得る又は含まれ得るコンポーネントを全て含んでいることを意味していない。別の実施形態では、図２と関連して説明されるコンポーネントおよび／または機能は、ここで説明する主題事項の構成の精神または範囲から離れずに、他のコンポーネント（示されたもの又は示されていないもの）に含ませること、またはサブコンポーネントに配することができる。幾つかの実施形態では、図２と関連して説明するコンポーネントおよび／または機能は、複数のデバイスにわたって分散させることができる。

【００２９】

[0034] 図２を参照すると、システム２０５は、スクリプト・コンポーネント２１０、ストア２２０、通信機構２２５、および他のコンポーネント（示さず）を含むことができる。システム２０５は、１以上の計算デバイスを含むことができる。そのようなデバイスは、例えば、パーソナル・コンピューター、サーバ・コンピューター、手持型またはラップトップ型のデバイス、マルチプロセッサー・システム、マイクロコントローラー・ベースのシステム、セットトップ・ボックス、プログラマブルの大衆消費電子製品、ネットワークＰＣ、ミニコンピューター、メインフレーム・コンピューター、セル電話、パーソナル・デジタル・アシスタント（ＰＤＡ）、ゲーム用デバイス、プリンター、セットトップを含むアプライアンス、メディア・センター、または他のアプライアンス、自動車への埋め込み型または取り付け型の計算デバイス、他のモバイル・デバイス、上記のシステムやデバイスの何れかを含む分散型計算環境などを含む。

【００３０】

[0035] システム２０５が単一のデバイスを含む場合、システム２０５として作動するように構成され得る例示のデバイスは、図１のコンピューター１１０を含む。システム２０５が複数のデバイスを含む場合、複数のデバイスのそれぞれは、図１のコンピューター

１１０を同様に又は異なって構成したものを含むことができる。
【００３１】
[0036] スクリプト・コンポーネント２１０は、インストーラー２１５、コード・ジェネレーター２１６、キャッシュ・マネージャー２１７、実行マネージャー２１８、および他のコンポーネント（示さず）を含むことができる。ここで用いられるコンポーネントという用語は、デバイスの全てまたは一部、１以上のソフトウェア・モジュールまたはその一部のコレクション、１以上のソフトウェア・モジュールまたはその一部と１以上のデバイスまたはその一部との何らかの組み合わせなどを含むものと、解釈される。
【００３２】
[0037] 通信機構２２５は、システム２０５が他のエンティティと通信することを可能にする。例えば、通信機構２２５は、システム２０５が他のエンティティと通信して、システム２０５でキャッシュされ得るパッケージおよび／またはスクリプト・コードを得ることを、可能にする。通信機構２２５は、ネットワーク・インターフェースまたはアダプター１７０、モデム１７２、または図２と関連して説明した通信を確立するための任意の他の機構とすることができる。
【００３３】
[0038] ストア２２０は、データへのアクセスを提供することができる任意のストレージ媒体である。ここで用いるアクセスとは、データを読み出すこと、データを書き込むこと、データを削除すること、データを更新すること、上記の２以上のことを含む組み合わせなどを含む。ストアは、揮発性メモリ（例えば、ＲＡＭ、メモリ内キャッシュなど）および不揮発性メモリ（例えば、永続ストレージ）を含むことができる。
【００３４】
[0039] データという用語は、１以上のコンピューター・ストレージ・エレメントにより表すことができる何れのものも含むように、広く解釈される。論理的に、データは揮発性または不揮発性のメモリにおいて、「１」および「０」の列で表すことができる。非バイナリー・ストレージ媒体を有するコンピューターでは、データは、ストレージ媒体の能力に従って表すことができる。データは、異なるタイプのデータ構造に分けて体系づけることができ、データ構造は、数や文字などのような単純なデータ・タイプ、階層やリンクや他の関連型のデータ・タイプ、複数の他のデータ構造や単純なデータ・タイプを含むデータ構造などを含む。データの幾つかの例は、情報、プログラム・コード、プログラム状態、プログラム・データ、他のデータなどを含む。
【００３５】
[0040] ストア２２０は、ハード・ディスク・ストレージ、他の不揮発性ストレージ、ＲＡＭなどのような揮発性メモリ、他のストレージ、上記のものの何らかの組み合わせなどを含むことができ、また、複数のデバイスにわたって分散させることもできる。ストア２２０は、システム２０５の外部または内部のものとすること、または外部と内部との双方のコンポーネントを含むことができる。
【００３６】
[0041] 始動および実行の時間を低減するため、メモリ・フットプリントを低減するため、不正行為に対する保護を可能にするため、および他の理由のために、コード・ジェネレーター２１６は、ソース・コードを、バイトコードまたは何らかの他の中間コードまたは実行可能コードへとコンパイルすることができる。コードは、コンピューターの行うアクションを示す命令を含む。コードはまた、コンピューターの行うアクション以外の情報を含むデータ、リソース、変数、定義、関係、関連付けなどを含むことができる。例えば、コードは、ウェブ・ページ、ＨＴＭＬ、ＸＭＬ、他のコンテンツなどを含むことができる。実施形態では、コードは、ソフトウェア・プロジェクトに含まれ得る。ソフトウェアは、コンピューターの行うアクション以外の情報を含む１以上のコード部分、データ、リソース、変数、定義、関係、関連付けなど、および構成情報などを、含むこと又は参照することができる。
【００３７】

[0042] コードにより示されるアクションは、スクリプト言語および非スクリプト言語、中間言語、アセンブリ言語、バイナリー・コード、他の言語、上記のものの何らかの組み合わせなどを含むソース・コード言語にエンコードすることができる。

【００３８】

[0043] インストーラー２１５は、システム２０５へパッケージをインストールすることができる。パッケージは、１以上のソフトウェア・アプリケーションを含むことができる。インストーラー２１５は、配置拡張ハンドラーを含むことができ、配置拡張ハンドラーは、パッケージがインストールされたターゲット・マシンで、パッケージがスクリプト・コードを含むか他の基準を含むかにかかわらず、インストールされたパッケージに基づいてカスタムのアクションが行われることを可能にする。

【００３９】

[0044] １つのインプリメンテーションでは、インストーラー２１５が、スクリプト・コードを持つパッケージと遭遇したとき、インストーラー２１５は、パッケージのコンポーネントに対応するエレメントを、そのパッケージのコードをプリコンパイルするためのキューまたは他のデータ構造へ付加することができる。

【００４０】

[0045] コード・ジェネレーター２１６は、データ構造からエレメントを取り出すことができ、それぞれのエレメントに対応するコードから、バイトコード、実行可能コード、または何らかの他のコードを生成することができる。エレメントはファイルを含むことができ、ファイルは、スクリプト、ファイルの一部（例えば、ＨＴＭＬページ内に埋め込まれたスクリプト）、コンパイルされたコードなどを含む。簡単にするため、ここでは、バイトコードという用語は、コード・ジェネレーター２１６の生成するコードを示すためにしばしば用いられる。しかし、コード・ジェネレーター２１６は、バイトコードを生成およびキャッシングするものに限定されず、別の実施形態では、先に述べたタイプのコードを含む他のタイプのコードを生成およびキャッシングすることが、理解される。

【００４１】

[0046] １つの実施形態では、コード・ジェネレーター２１６は、バイトコードを生成する前に、パッケージが完全にインストールされて、パッケージに対応するエレメントがコンパイルのためのデータ構造へ配されるまで、待つことができる。別の実施形態では、コード・ジェネレーター２１６は、データ構造においてエレメントが使用可能になるとすぐに、またはソース・コード・モジュールがインストールされるとすぐに、バイトコードの生成を開始することができる。この別の実施形態では、１つの例において、コード・ジェネレーター２１６は、パッケージのインストールに失敗した場合に、生成したバイトコードを処分することができる。別の例では、コード・ジェネレーター２１６は、生成を止めた時点（例えば、電力が回復してマシンがリブートした後）から生成を再開することができる。１つのインプリメンテーションでは、コード・ジェネレーター２１６は、システム２０５がアイドルのとき（例えば、パッケージをインストール中ではないときや、システム２０５の処理帯域を消費する他のタスクを行っていないとき）や、システム２０５が電線の電力で（例えば、バッテリーの電力ではなく）動作しているときや、一日のうちの定められた又は構成可能な時間などに行うプロセスとして、インプリメントすることができる。

【００４２】

[0047] ここで用いる「プロセス」という用語およびその異形は、１以上の従来型のプロセス、スレッド、コンポーネント、ライブラリー、タスクを行うオブジェクトなどを含むことができる。プロセスは、ハードウェア、ソフトウェア、またはハードウェアとソフトウェアとの組み合わせでインプリメントすることができる。実施形態では、プロセスは任意の機構であるが、呼び出されると、アクションを行うこと又はアクションを行う際に用いることができる。プロセスは、複数のデバイスにわたって分散させること、または１つのデバイスに配することができる。

【００４３】

[0048] １つの実施形態では、コード・ジェネレーター２１６はまた、コードをオン・ザ・フライで（例えば、コードに遭遇したときに）生成するために用いることができる。例えば、ユーザーが、ソフトウェアをインストールし、そのソフトウェアを含むパッケージ全体に対するバイトコードが生成される前にそのソフトウェアを実行したい場合、コード・ジェネレーター２１６を用いて、そのソフトウェアに対するコードを必要に応じて生成することができる。換言すると、ユーザーは、ユーザーがパッケージのソフトウェアを実行することを可能とされる前に、コード・ジェネレーター２１６がパッケージ全体のバイトコードを生成するのを待つ必要がない。更に、コンポーネントに対するソース・コードが変更されているという別の状況があり得、その場合、コード・ジェネレーター２１６は、コンポーネントに対するバイトコードを迅速に生成するために用いることができる。

【００４４】

[0049] 更に、コード・ジェネレーター２１６は、コードをコンパイルするため及びコードをキャッシュするために、オンデマンドで用いることができる。例えば、インターネット・ブラウザー・アプリケーションでは、新たなコードへのリンクは、動的に発見することができる。例えば、ウェブ・ドキュメントは、ダウンロードされ実行される他のコードへのリンクを含むことができる。コード・ジェネレーター２１６は、それらのリンクにより参照されたコードをコンパイルするために、およびそのコードの後の実行を速くするように、コンパイルされたコードをキャッシュするために、用いることができる。

【００４５】

[0050] コード・ジェネレーター２１６は、コードを生成するために、「サンド・ボックス」において又は「サンド・ボックス」を用いてインプリメントすることができる。サンド・ボックスは、それがアクセスできるデータに関する権限に対しての制限のある環境である。１つのインプリメンテーションでは、コード・ジェネレーター２１６は、権限を制限されたプロセスとしてインプリメントすることができる。プロセスは、プロセスが読み出し又は書き込みを行えるバイトコード・ファイルに対するハンドルを渡され得、ソース・コードに対してのリード・オンリー・アクセスが与えられ得る。セキュリティの目的のため、プロセスは、上述のリソース以外の他のリソースへのアクセスを有さない。

【００４６】

[0051] １つのインプリメンテーションでは、コード・ジェネレーター２１６は、仮想環境においてホストすることができる。仮想環境は、コンピューターによりシミュレートまたはエミュレートされた環境である。仮想環境は、物理マシン、オペレーティング・システム、１以上のインターフェースの組、上記のものの部分、上記のものの組み合わせなどを、シミュレートまたはエミュレートすることができる。マシンがシミュレートまたはエミュレートされたとき、そのマシンは時には仮想マシンと呼ばれる。仮想マシンは、仮想マシン上で実行されているソフトウェアには、物理マシンのように見えるマシンである。ソフトウェアは、仮想ハード・ドライブ、仮想フロッピー（登録商標）・ディスクなどのような仮想ストレージ・デバイスにファイルをセーブすること、仮想ＣＤからファイルを読み出すこと、仮想ネットワーク・アダプターを介して通信することなどができる。

【００４７】

[0052] 仮想環境は、仮想環境の外部のデータや他のリソースに対して、アクセスを制限するか又はアクセスできないようにすることができる。従って、コードが仮想環境のホストを汚染するかどうかを心配する必要無しに、仮想環境は、信頼できないコードのコンパイルに対しての適切な環境を提供することができる。

【００４８】

[0053] バイトコードをファイルへ書き込んだ後、プロセスは、電子的にファイルにサインまたは「封印」し、それにより、ファイルに対する何れの変更も検出可能とされる。ファイルの封印は、多くの暗号化および／またはファイル・システムの形態をとり、これは当業者には理解できるであろう。例えば、１つのインプリメンテーションでは、ファイル・システムは、封印したときに、ファイルに拡張した属性を設定することができる。その後にファイルが変更されると、拡張した属性は変更され、ファイルがもはや封印されて

いないことを、示すことができる。

【００４９】

[0054] ストア２２０へバイトコードを書き込むことを、ここでは時には、バイトコードをキャッシングすると言う。キャッシュ・マネージャー２１７は、バイトコードをストア２２０へ格納するように、および要求されたときにバイトコードへのアクセスを提供するように、動作することができる。キャッシュ・マネージャー２１７は、システム２０５の再起動後にバイトコードが使用可能であるように、バイトコードがストア２２０に永続的に格納されることを、確実にすることができる。キャッシュ・マネージャー２１７はまた、バイトコードへのアクセスを速くするために、望まれる場合には、バイトコードのコピーを、メイン・メモリ、高速キャッシュ・メモリ、ビデオ・メモリ、揮発性および不揮発性のメモリを含む他のメモリなどへ、格納させることができる。

【００５０】

[0055] 更に、ここではファイルという用語が時には使用されるが、ここで説明する主題事項の構成の精神および範囲から離れずに、別の実施形態では、バイトコードは、揮発性および／または不揮発性のメモリ、データベースまたはその一部（例えば、レコード）、または何らかの他のストアへ書き込まれ得ることが、理解される。

【００５１】

[0056] １より多くのユーザーがパッケージをインストールする場合、インプリメンテーションは、パッケージに対してのバイトコードの複数のコピーまたは１つのみのコピーを、ストア２２０へキャッシュすることができる。パッケージが１より多くのプロセッサー・アーキテクチャー（例えば、３２ビットおよび６４ビット）をターゲットにする場合、コード・ジェネレーター２１６は、各プロセッサー・アーキテクチャーに対してのターゲットとされたバイトコードを生成してキャッシュすることができる。

【００５２】

[0057] 幾つかのインプリメンテーションでは、パッケージのバイトコードを再生成するための幾つかのトリガがあり得る。例えば、バイトコードを含むファイルの封印が壊されているか（例えば、不正が検出されるか）、ストレージ媒体が破損されているか、またはキャッシュが無効または不完全であることを他のデータが示す場合、これは、パッケージのバイトコードを再生成させるトリガとして働くことができる。この場合におてい、実行マネージャー２１８が、封印が壊されていると判定した場合、実行マネージャー２１８は、コード・ジェネレーター２１６に命令して、パッケージに対するバイトコードを再生成させることができる。コード・ジェネレーター２１６は、コード・ジェネレーター２１６が最初にバイトコードを生成した様式（例えば、パッケージのコンポーネントに対応するエレメントを、キューまたは他のデータ構造に配し、バイトコードの生成が必要なことを示す）と似た様式で、バイトコードを生成することができる。

【００５３】

[0058] 別の例として、バイトコードを用いる環境が新たなバージョンへとアップグレードされた場合、これは、バイトコードを再生成することをトリガすることができる。例えば、インターネット・ブラウザーまたは別の実行環境が、異なるバイトコードの文法やシンタックスを有する新たなバージョンへとアップグレードされた場合、これは、その新たなバージョンに対して適切なバイトコードを再生成することをトリガすることができる。この例では、トリガは、ユーザーがそのバイトコードと関連するソフトウェアの実行を試みたときに、生じ得る。これは、使用のときにバイトコードを再生成させることができ、全てのインストールされたパッケージのバイトコードの一度での再生成を避けることができる。

【００５４】

[0059] 別の例として、周期的なメンテナンス・タスクが、異なるバイトコードの文法やシンタックスを有する新たなバージョンへと実行環境が更新されたことを、検出した場合、これは、その新たなバージョンに対して適切なバイトコードを再生成することをトリガすることができる。

【００５５】
[0060] 別の例では、新たなバージョンのパッケージがインストールされる場合、これは、バイトコードを再生成させるためのトリガとして働くことができる。
[0061] 別の例では、バイトコードが何らかの理由で削除される場合、これは、バイトコードを再生成させるためのトリガとして働くことができる。
【００５６】
[0062] ストア２２０では、ソース・コードおよび対応するバイトコードは、１つの例では、図３に示すように格納することができる。図３は、ここで説明される主題事項の構成に従って用いることができる例示のデータ構造を示す。１つのインプリメンテーションでは、図３に示すデータ構造は、データ構造において使用可能なソース・コード・ファイルおよびバイトコード表現を示すテーブルを有することができる。テーブルはまた、データ構造におけるソース・コード・ファイルおよびバイトコード表現の場所を示すデータを含むことができる。データ構造は、ここで説明する主題事項の構成の精神および範囲から離れずに、ファイルに格納すること又はファイルとして表すこと、データベースに格納すること、または別のストアに格納することができる。
【００５７】
[0063] 図３を見ると、データ構造３０５は、並べて配されたソース・コードと、対応するバイトコードとを有することができる。ストア２２０がファイル・システムとしてインプリメントされる場合、データ構造３０５は、バイトコードへアクセスするために必要なファイル・ハンドラーなどの数を低減することができるので、データを得る効率を改善することができる。ソース・コードが、１以上のウェブ・ページなどではなくパッケージ内にある場合、ソース・コードおよび対応するバイトコードを並べて配することは、デフォルトの挙動であり得る。望まれる場合には、このデフォルトの挙動をオーバーライドすることができる。
【００５８】
[0064] 別のインプリメンテーションでは、ストア２２０がファイル・システムとしてインプリメントされる場合、パッケージの１以上のコンポーネントのそれぞれに対して個別のバイトコード・ファイルがあり得る。これは、幾つかのシナリオでは効率が低下し得るが、別の利点および用法を有し得る。このインプリメンテーションは、例えば、ソース・コードがパッケージの外部（例えば、１以上のウェブ・サイト）に存在し得るとき、および／またはソース・コードが動的に生成され得るときに、用いることができる。
【００５９】
[0065] 別の例として、これは、ホストまたはユーザーが１つのウェブ・サイトと関連する全情報を消去することを望み得る、自然に分離されたエクスペリエンスの場合に、用いることができる。
【００６０】
[0066] 別の例として、これは、ブラウザーに対して、所与のウェブ・サイトのリソースが予め知らされているというシナリオにおいて、用いることができる。この例では、ブラウザーは、ウェブ・サイトのリソースをプリフェッチして、ネットワーク接続の無いときでもウェブ・サイトを用いることができる。この例では、ウェブ・サイトにおける深いページに対してでさえ、ＨＴＭＬマークアップのページに対するスクリプト・ファイルへの明確な参照に遭遇していなくとも、バイトコードを生成することができる。
【００６１】
[0067] １つのファイルにおけるソース・コードおよび対応するバイトコードを捜すか、またはそれぞれのソース・コード／バイトコード・ペアに対しての個別のファイルを捜すかは、パッケージを実行している環境に基づき得る。環境がウェブ・ブラウザーである場合、キャッシュ・マネージャーは、各ソース・コード・コンポーネントに対応するバイトコードに対する個別のファイルを捜し得る。環境が、全ソース・コード・コンポーネントがパッケージから来たものであるのでそれらが予め知られているといったアプリケーション・フレームワークまたは別の環境（例えば、上記の、予め知らされているウェブ・サ

イトの例）である場合、キャッシュ・マネージャーは、１つのファイルに並べて配されるパッケージ全体に対するバイトコードを捜し得る。

【００６２】

[0068] データ構造３０５において、パッケージの全てのコード・コンポーネントが対応するバイトコードを有さないこともあり得る。例えば、様々な理由で、パッケージの１以上のコンポーネントに対してバイトコードを生成しないことが望ましいことがある。そのような場合、そのコンポーネント（１以上）に対するソース・コードおよびバイトコードは、データ構造３０５から省かれ得る。

【００６３】

[0069] 更に、データ構造３０５は、２以上のパッケージ間での共有のために、メモリ・マップされ得る。例えば、幾つかの場合、複数の異なるパッケージは、１以上の同一のソース・コンポーネントを有し得る。これらの場合、メモリ・マッピングは、同一のソース・コンポーネントに対するバイトコードを共有するために用いることができ、それにより、メモリにおいてバイトコードの複数のコピーを必要としなくなる。更に、データ構造３０５におけるバイトコードはリード・オンリーとすることができ、それにより、共有のためにメモリ・マッピングされたときに変化しなくなる。

【００６４】

[0070] 図２および図３を参照すると、コードの実行において、実行マネージャー２１８は、実行マネージャーがパースしているドキュメント（例えば、ＨＴＭＬ、ＸＭＬ、ワード・プロセッシング、または他のドキュメント）におけるソース・コードを参照する参照（例えば、ＨＴＴＰまたは他の参照）を、見つけることができる。実行マネージャー２１８は、最初に、その参照に対応するバイトコードの現在のバージョンがキャッシュに存在するかを確かめるために、データ構造を調べることにより検査を行うことができる。バイトコードがキャッシュ内に存在する場合、実行マネージャー２１８は、実行のためにそのバイトコードを得ることができる。バイトコードがキャッシュ内に存在しない場合、実行マネージャー２１８は、ソース・コードの場所からソース・コードを得ること、およびソース・コードをコンパイルさせて遅延無く（例えば、コード・ジェネレーター２１６が、キューにある未処理のコンパイル・リクエストの何れのものをも完了させることを待つこと無しに）実行させることができる。

【００６５】

[0071] １つのインプリメンテーションでは、ソース・コードに対応するバイトコードは、ソース・コードに対する並列のディレクトリーに存在し得る。例えば、ソース・コードが、Ｃ：＼ＰａｃｋａｇｅＮａｍｅ＼ＳｏｕｒｃｅＣｏｄｅ＼ｓｏｕｒｃｅｃｏｄｅｎａｍｅ．ｓｃｒｉｐｔｆｉｌｅに存在する場合、バイトコードは、存在する場合には、Ｃ：＼ＰａｃｋａｇｅＮａｍｅ＼ＳｏｍｅＮａｍｅ＼ｂｙｔｅｃｏｄｅｎａｍｅ．ｂｙｔｅｃｏｄｅｆｉｌｅに存在し得る。

【００６６】

[0072] 別のインプリメンテーションでは、ソース・コードはリモート・デバイスに存在し得る。

[0073] 図４は、ここで説明する主題事項の構成が動作できる環境のコンポーネントの例示の構成を表すブロック図である。図４に示すコンポーネントは、例示であり、必要とされ得る又は含まれ得るコンポーネントを全て含んでいることを意味していない。別の実施形態では、図４と関連して説明するコンポーネントおよび／または機能は、ここで説明する主題事項の構成の精神または範囲から離れずに、他のコンポーネント（示されたもの又は示されていないもの）に含ませること、またはサブコンポーネントに配することができる。幾つかの実施形態では、図４と関連して説明するコンポーネントおよび／または機能は、複数のデバイスにわたって分散させることができる。

【００６７】

[0074] 図４を見ると、環境４０５は、ターゲット・デバイス４１０、ネットワーク４１５、コード・サーバー４１７、および他のコンポーネント（示さず）を含むことができ

る。ターゲット・デバイス４１０およびコード・サーバー４１７は、１以上の計算デバイスを含むことができる。そのようなデバイスは、例えば、パーソナル・コンピューター、サーバ・コンピューター、手持型またはラップトップ型のデバイス、マルチプロセッサー・システム、マイクロコントローラー・ベースのシステム、セットトップ・ボックス、テレビジョン、プログラマブルの大衆消費電子製品、ネットワークＰＣ、ミニコンピューター、メインフレーム・コンピューター、セル電話、パーソナル・デジタル・アシスタント（ＰＤＡ）、ゲーム用デバイス、プリンター、セットトップを含むアプライアンス、メディア・センター、または他のアプライアンス、自動車への埋め込み型または取り付け型の計算デバイス、他のモバイル・デバイス、上記のシステムやデバイスの何れかを含む分散型計算環境などを含む。ターゲット・デバイス４１０またはコード・サーバー４１７として作動するように構成することができる例示のデバイスは、図１のコンピューター１１０を含む。

【００６８】

[0075] １つの実施形態では、ネットワーク４１５はインターネットを含むことができる。１つの実施形態では、ネットワーク４１５は、１以上のローカル・エリア・ネットワーク、ワイド・エリア・ネットワーク、直接接続、仮想接続、プライベート・ネットワーク、仮想プライベート・ネットワーク、上記のものの何らかの組み合わせなどを、含むことができる。

【００６９】

[0076] コード・サーバー４１７は、ターゲット・デバイス４１０へコードを提供することができる。１つの実施形態では、コード・サーバー４１７はウェブ・サーバーとすることができる。別の実施形態では、コード・サーバー４１７は、組織のプライベート・ネットワークの内部のマシンとすることができ、それはコードを含む。別の実施形態では、コード・サーバー４１７は、ソース・コードをターゲット・デバイス４１０へ提供することができる任意のデバイスを含むことができる。

【００７０】

[0077] ターゲット・デバイス４１０は、コードを実行する　ことができるデバイスである。ターゲット・デバイス４１０は、コンパイル環境４２０と、実行環境４２５と、インストーラー４３０と、ストア４３５とを含むことができる。インストーラー４３０は、図２のインストーラー２１５と類似にインプリメントすることができ且つ作動することができ、ストア４３５は、図２のストア２２０と類似にインプリメントすることができ且つ作動することができる。

【００７１】

[0078] コンパイル環境４２０は、権限を制限された環境を含むことができ、そこでソース・コードをバイトコードへとコンパイルすることができる。１つの実施形態では、コンパイル環境４２０は、上記のような仮想環境を含むことができる。別の実施形態では、コンパイル環境４２０は、先に説明したような、権限を制限された異なる環境を含むことができる。

【００７２】

[0079] 実行環境４２５は、パッケージのソフトウェアが実行される環境を含むことができる。１つのインプリメンテーションでは、実行環境４２５は、インターネット・ブラウザーを含むことができる。別の実施形態では、実行環境４２５は、ソース・コードを含み得るパッケージのソフトウェアを実行することができるホスト・プロセスを、含むことができる。

【００７３】

[0080] 図５および図６は、ここで説明する主題事項の構成に従って生じ得る例示のアクションを概略的に表すフロー図である。説明を簡単にするために、図５および図６と関連して説明するメソドロジーは、一連の行為として示され説明される。ここで説明する主題事項の構成は、示される行為および／または行為の順番により限定されないことを、理解および解釈すべきである。１つの実施形態では、行為は、後に説明する順に生じる。し

かし、別の実施形態では、行為は、並列、別の順、および／またはここで呈示も説明もしていない他の行為と共に、生じ得る。更に、ここで説明する主題事項の構成に従ってメソドロジーをインプリメントするために、全ての示した行為が必要ではない場合もあり得る。更に、メソドロジーは、状態図を介して一連の相互に関係した状態として又はイベントとして代替的に表せ得ることを、当業者は理解および解釈するであろう。

【００７４】

[0081] 図５を見ると、ブロック５０５においてアクションが開始する。

[0082] ブロック５１０において、パッケージが受け取られ、これは、ターゲット・デバイスでインストールするソフトウェアのソース・コードを含む。例えば、図４を参照すると、ターゲット・デバイス４１０は、コード・サーバー４１７からパッケージを得ることができる。別の例としては、インストーラー４３０は、ストア４３５に存在するパッケージをインストールするように命令され得る。

【００７５】

[0083] ブロック５１５において、パッケージは、ターゲット・デバイス４１０にインストールされる。例えば、図２を参照すると、インストーラー２１５は、ストア２２０にパッケージをインストールすることができる。

【００７６】

[0084] ブロック５２０において、キューなどのようなデータ構造においてインジケーターが配される。データ・インジケーターは、パッケージのソース・コードがバイトコードへとコンパイルされることを示す。パッケージのソース・コードの各ファイルに対するデータ構造において配される別個のインジケーターが存在し得る。例えば、図２を参照すると、インストーラー２１５は、ストア２２０に格納されたデータ構造において１以上のフラグを配することができる。

【００７７】

[0085] ブロック５２５において、アクションを開始または完了する前にパッケージのソフトウェアを実行するリクエストが受け取られないかぎり、ソース・コードを最初にコンパイルするアクションが行われる。ソース・コードをコンパイルするためのこれらのアクションは、例えば、下記のことを含むことができる。

【００７８】

[0086] １． データ構造を通して反復して、コンパイルが必要であることを示すインジケーター（１以上）を見つける。

[0087] ２． 上記のステップ１で見つけたそれぞれの適用可能なソース・コード・エレメントを、バイトコードまたは何らかの他のコードへとコンパイルする。

【００７９】

[0088] 例えば、図２を参照すると、コード・ジェネレーター２１６は、データ構造を通して反復して、データ構造内で見つかったフラグに対してのソース・コード・ファイルをコンパイルすることができる。別の例として、コード・ジェネレーター２１６は、ファイルのリストを通して反復して、所与の拡張子（例えば、「．ｊｓ」や、ソース・ファイルを示す別の拡張子）を有する何れのファイルをもコンパイルすることができる。

【００８０】

[0089] ブロック５３０において、コンパイルされたコードは、次に、不揮発性メモリにおいて保持させることができる。例えば、図２を参照すると、キャッシュ・マネージャー２１７は、コード・ジェネレーター２１６により生成されたコードを、ストア２２０に格納することができる。コードはまた、素早いロードや実行のために、メモリ内キャッシュまたはＲＡＭなどのような揮発性メモリに格納することもできる。

【００８１】

[0090] ブロック５３５は、アクション５２０～５３０の横側に置かれており、パッケージのソフトウェアを実行するリクエストが、それらのアクションの前、最中、または後に生じ得ることを示している。そのようなリクエストが受け取られると、他のアクションは、５２０～５３０のアクションの開始または完了の前に行われ得る。例えば、そのよう

なリクエストが、ブロック５２０～５３０のアクションの開始または完了の前に受け取られた場合、ソース・コードは、迅速にコンパイルおよび実行され得る。そのようなリクエストが受け取られた場合に生じることを記述する他のアクションは、図６と関連して説明する。

【００８２】

[0091] ブロック５４０において、他のアクションがある場合には、それを行うことができる。例えば、再生成トリガ・イベントを、受け取ることができる。それに応じて、ブロック５２０～５３０のアクションを、可能性としては、異なるソース・コード（ソース・コードが変更された場合）およびその異なるソース・コードからコンパイルされた異なる第２のコードを用いて、再び実行することができる。

【００８３】

[0092] 図６を見ると、ブロック６０５においてアクションが開始される。

[0093] ブロック６１０において、ソース・コードを含むソフトウェアを実行するリクエストが受け取られる。例えば、図２を参照すると、実行マネージャー２１８は、ストア２２０にインストールされたパッケージのソフトウェアを実行するリクエストを受け取る。

【００８４】

[0094] ブロック６１５において、コードが既にコンパイルされて不揮発性ストレージに格納されているかどうかに関する判定が、行われる。そうである場合、アクションはブロック６４０で継続され、そうではない場合、アクションはブロック６２０で継続される。例えば、図２を参照すると、実行マネージャー２１８は、キャッシュ・マネージャー２１７を用いて、パッケージが既にコンパイルされてストア２２０に格納されているかどうかを、判定する。

【００８５】

[0095] ソース・コードが既にコンパイルされているかどうかを判定することは、不揮発性ストレージの既知の場所においてコンパイル済みコードを検査することを含み得る。既知の場所は、パッケージの名前または他の識別子に対応し得る。例えば、既知の場所は、そのパッケージに因んで名付けられたディレクトリーや、そのディレクトリーの子孫ディレクトリーであり得る。別の例として、既知の場所は、データ構造において参照される場所であり得、それは、そのコードに対してのコンパイル済みコードが存在する場合には、それの配される場所を示す。別の例として、既知の場所は、ソース・コードのソースの場所（例えば、コード・サーバー）を識別する参照から導き出される場所であり得、ソースの場所へは、ネットワークを介して到達可能である。更に別の例として、ソース・コードが既にコンパイルされているかどうかを判定することは、パッケージに対するバイトコードをパッケージに対するソース・コードと並べて配するファイルに格納されたデータ構造を検査することを、含み得る。データ構造は、第２のコードが既に生成されてファイルに格納されているかどうかを、示すことができる。

【００８６】

[0096] ブロック６２０において、ソース・コードが得られる。例えば、図２を参照すると、コード・ジェネレーター２１６は、ストア２２０からソース・コードを得る。

[0097] ブロック６２５において、バイトコード（または他のコード）が、ソース・コードから生成される。例えば、図２を参照すると、コード・ジェネレーター２１６は、バイトコード、機械実行可能コード、または何らかの他の中間コードを、上記で得たソース・コードから作成する。

【００８７】

[0098] ブロック６３０において、コード（又は実行可能コードや他の中間コードなどのような、それから導き出されたコード）が実行される。例えば、図２を参照すると、実行マネージャー２１８は、上記の生成されたコードを実行する。

【００８８】

[0099] ブロック６３５において、コードは不揮発性ストレージで保持される。ブロッ

ク６３５により表されるアクションは、上記の又は後の期間に行われるアクションと関連して生じ得る。例えば、図２および図３を参照すると、コード・ジェネレーター２１６は、キャッシュ・マネージャー２１７を用いて、データ構造３０５において、パッケージに対するコードを並べて配すること、およびそのデータ構造３０５を、ソフトウェアを後に実行する際に使用するために、ストア２２０に格納することができる。別の例としては、後に、コード・ジェネレーター２１６が、ソース・コードを再び得て、コンパイルされたコードを再び生成して、そのコンパイルされたコードをストア２２０に格納することができる。

【００８９】

[00100] ブロック６４０において、コードが既にコンパイルされていた場合、そのコンパイルされたコードが得られる。例えば、図２を参照すると、実行マネージャー２１８は、キャッシュ・マネージャー２１７からバイトコードを得ることができる。

【００９０】

[00101] ブロック６４５において、得られたコード（又はバイナリー・コードなどのような、それから導き出されたコード）が実行される。例えば、図４を参照すると、コードは、実行環境４２５において実行され得る。

【００９１】

[00102] ブロック６５０において、他のアクションがある場合、それを実行することができる。例えば、バイトコードを含むファイルを、メモリ・マップすることができ、そのメモリ・マッピングを介して複数のプロセスで共有することができる。

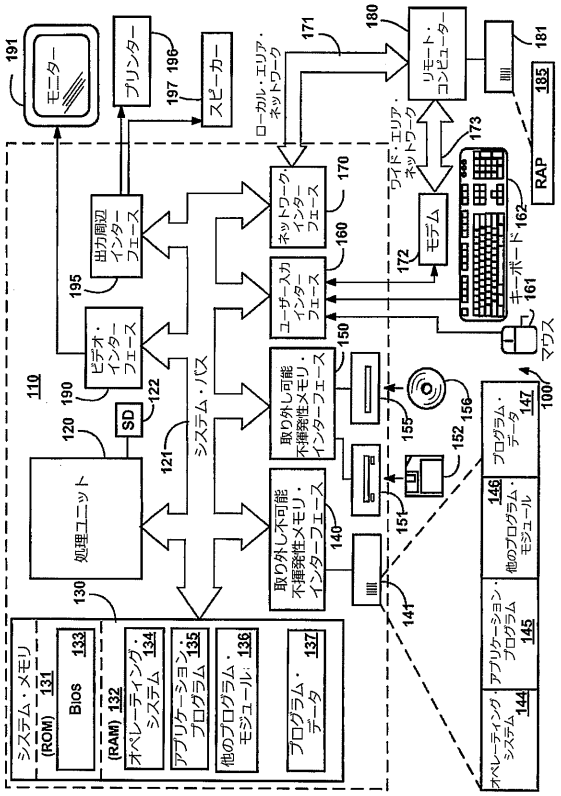
【００９２】

[00103] 別の例として、他のアクションは、コンパイルされたコードが、生成された後に変更されたかどうかを検査することを含むことができ、そうである場合に、ソース・コード（これは元のソース・コードとは異なり得る）を再び得ることと、そのソース・コードを再びコンパイルすることと、そのコンパイルされたコードを、ソフトウェアを後に実行する際に使用するために、不揮発性ストレージに格納することとを、含むことができる。

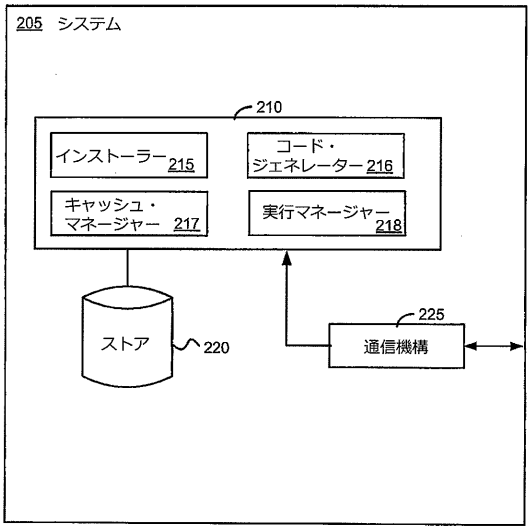
【００９３】

[00104] 上記の詳細な説明から理解できるように、ソフトウェアの生成およびキャッシュと関連する構成が説明された。ここで説明する主題事項の構成は、様々な変更や代替の構成が可能であるが、その特定の例示の実施形態が、図面に示され、上記で詳細に説明されている。しかし、特許請求される主題事項の構成を、開示した特定の形態に限定することは意図しておらず、むしろ、ここで説明した主題事項の様々な構成の精神内および範囲内にある全ての変更、代替の構成、および等価物をカバーすることを意図していることを、理解すべきである。

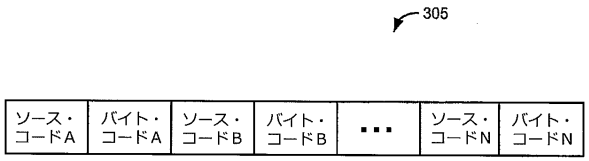
【図１】


モニター
プリンター
スピーカー
ローカル・エリア・ネットワーク
リモート・コンピューター
ワイド・エリア・ネットワーク
RAP
ネットワーク・インターフェース
モデム
出力周辺インターフェース
ユーザー入力インターフェース
キーボード
マウス
ビデオ・インターフェース
システム・バス
取り外し可能不揮発性メモリ・インターフェース
取り外し不可能不揮発性メモリ・インターフェース
処理ユニット
SD
プログラム・データ
他のプログラム・モジュール
アプリケーション・プログラム
オペレーティング・システム
システム・メモリ
(ROM)
(RAM)
Bios
オペレーティング・システム
アプリケーション・プログラム
他のプログラム・モジュール
プログラム・データ

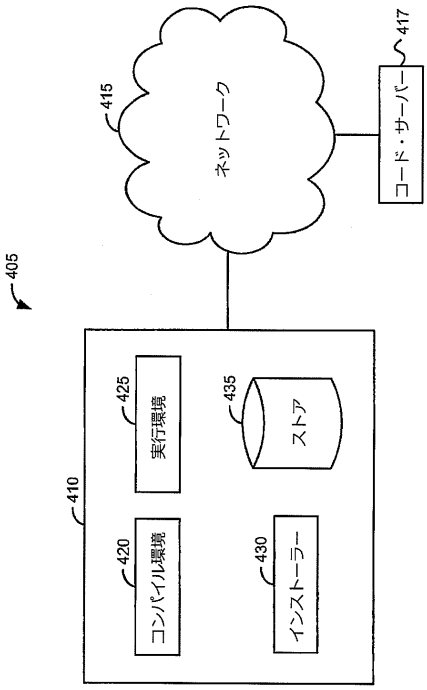
【図２】


205 システム
210
インストーラー 215
コード・ジェネレーター 216
キャッシュ・マネージャー 217
実行マネージャー 218
ストア 220
225
通信機構

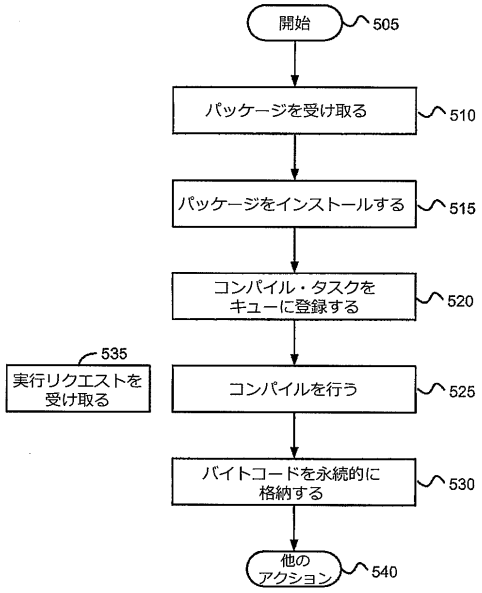
【図３】


305
ソース・コードA
バイト・コードA
ソース・コードB
バイト・コードB
...
ソース・コードN
バイト・コードN

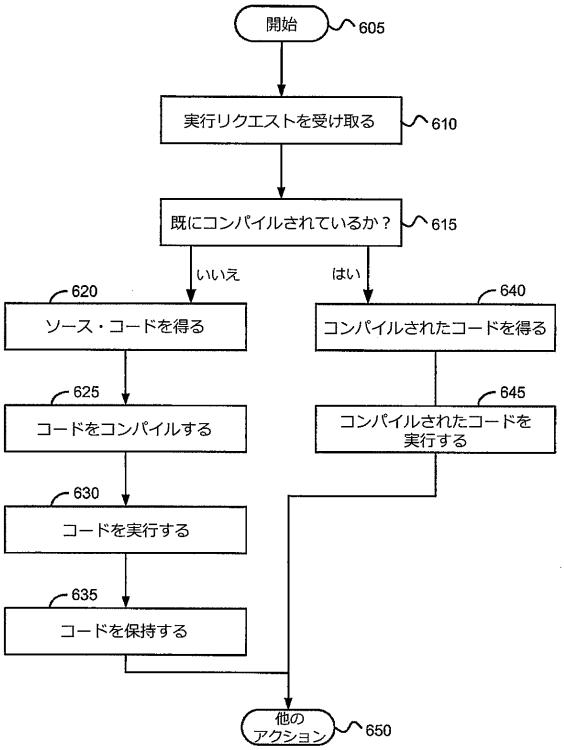
【図４】


405
ネットワーク
415
コード・サーバー
417
410
実行環境
425
ストア
435
コンパイル環境
420
インストーラー
430

【図５】


開始
505
パッケージを受け取る
510
パッケージをインストールする
515
コンパイル・タスクをキューに登録する
520
実行リクエストを受け取る
535
コンパイルを行う
525
バイトコードを永続的に格納する
530
他のアクション
540

【図６】


開始
605
実行リクエストを受け取る
610
既にコンパイルされているか？
615
いいえ
はい
620
ソース・コードを得る
640
コンパイルされたコードを得る
625
コードをコンパイルする
645
コンパイルされたコードを実行する
630
コードを実行する
635
コードを保持する
他のアクション
650

フロントページの続き

(74)代理人 100101373
弁理士 竹内 茂雄
(74)代理人 100118902
弁理士 山本 修
(74)代理人 100162846
弁理士 大牧 綾子
(72)発明者 フィッシャー, ジョモ
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, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ
(72)発明者 ジャクソン, マイケル・ウェイン
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, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ
(72)発明者 キリック, ユーナス
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, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ
(72)発明者 ルッコ, スティーヴン・エドワード
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, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ
(72)発明者 マクギャサ, ジェシー・ディー
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, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ
(72)発明者 ミアドヴィッチ, イェルゼイ・ズィー
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, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ
(72)発明者 スタイナー, スティーヴン・ジェイ
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, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ

審査官 石川 亮

(56)参考文献 特表２００４－５２８６２６（ＪＰ, Ａ）

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ ９／４４
Ｇ０６Ｆ ９／４４５
Ｇ０６Ｆ １３／００