

(19) 日本国特許庁(JP)

(12) 特許公報(B2)

(11) 特許番号

特許第4796966号
(P4796966)

(45) 発行日 平成23年10月19日(2011.10.19)

(24) 登録日 平成23年8月5日(2011.8.5)

(51) Int.Cl. F I
G06F 9/445 (2006.01) G06F 9/06 610L
 G06F 9/06 640A

請求項の数 18 (全 24 頁)

(21) 出願番号	特願2006-536561 (P2006-536561)	(73) 特許権者	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 9805 2-6399 レッドモンド ワン マイ クロソフト ウェイ
(86) (22) 出願日	平成16年7月21日(2004.7.21)	(74) 代理人	100077481 弁理士 谷 義一
(65) 公表番号	特表2007-519071 (P2007-519071A)	(74) 代理人	100088915 弁理士 阿部 和夫
(43) 公表日	平成19年7月12日(2007.7.12)	(72) 発明者	マーク エイ. アルカザール アメリカ合衆国 98052 ワシントン 州 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 ョン内
(86) 国際出願番号	PCT/US2004/023546		
(87) 国際公開番号	W02005/045562		
(87) 国際公開日	平成17年5月19日(2005.5.19)		
審査請求日	平成19年7月18日(2007.7.18)		
(31) 優先権主張番号	10/692, 323		
(32) 優先日	平成15年10月23日(2003.10.23)		
(33) 優先権主張国	米国 (US)		

最終頁に続く

(54) 【発明の名称】 ソフトウェアアプリケーションをプログレッシブインストールするためのシステムおよび方法およびAPI

(57) 【特許請求の範囲】

【請求項1】

アプリケーションに関連するアプリケーションパッケージであって、ユーザによる前記アプリケーションとの対話的操作を可能にするために必要な最小限のコードを含むアプリケーションパッケージを識別するステップと、

前記識別されたアプリケーションパッケージを、ローカルのストアにダウンロードするステップと、

前記ローカルのストアを含むクライアントから前記アプリケーションに関連する複数の追加のリソースのうちの1つに対する要求を受け取ると、前記追加のリソースをダウンロードするステップであって、前記複数の追加のリソースのうちの1つに対する要求は、前記アプリケーションパッケージに含まれたコードを実行する前記クライアントが、対応するユーザの操作に応じて発生させるステップと、

前記アプリケーションをクライアントサイドのアプリケーションに遷移させるための要求を受け取ると、残りのリソースを前記ローカルのストアにダウンロードするステップであって、前記アプリケーションをクライアントサイドのアプリケーションに遷移させるための要求は、対応するユーザの操作なしに作成されるステップと

をコンピュータに実行させることを特徴とするプログラム。

【請求項2】

前記アプリケーションパッケージの識別は、前記アプリケーションに関連するハイパーリンクをアクティブにすることを含むことを特徴とする請求項1に記載のプログラム。

【請求項 3】

前記アプリケーションパッケージの識別は、前記アプリケーションパッケージを識別するマニフェストの検索を開始することを含むことを特徴とする請求項 1 に記載のプログラム。

【請求項 4】

前記アプリケーションパッケージはアプリケーションのマニフェストを含むことを特徴とする請求項 1 に記載のプログラム。

【請求項 5】

前記アプリケーションパッケージに含まれたコードは実行可能形式であることを特徴とする請求項 1 に記載のプログラム。

10

【請求項 6】

前記アプリケーションに関連するウェブページ上のハイパーリンクをアクティブにすると、前記アプリケーションに関連する追加のリソースをダウンロードするステップをさらにコンピュータに実行させることを特徴とする請求項 1 に記載のプログラム。

【請求項 7】

前記追加のリソースはオンデマンドのリソースを含むことを特徴とする請求項 6 に記載のプログラム。

【請求項 8】

前記オンデマンドのリソースを前記ローカルのストアに格納するステップをさらにコンピュータに実行させることを特徴とする請求項 7 に記載のプログラム。

20

【請求項 9】

前記アプリケーションを前記クライアントサイドのアプリケーションに遷移させることは、複数の残りのリソースを前記ローカルのストアに格納することを含み、前記複数の残りのリソースは、前記ローカルのストアに現在格納されていないリソースであることを特徴とする請求項 1 に記載のプログラム。

【請求項 10】

前記アプリケーションを前記クライアントサイドのアプリケーションに遷移させることは、前記クライアントサイドのアプリケーションがオフラインで起動されたとき、前記クライアントサイドのアプリケーションの実行が前記アプリケーションに関連付けられた状態で再開するように、前記アプリケーションに関連付けられた状態を前記クライアントサイドのアプリケーションに遷移させることを含むことを特徴とする請求項 1 に記載のプログラム。

30

【請求項 11】

前記状態の情報は前記アプリケーションに関連するアプリケーションオブジェクトに格納されることを特徴とする請求項 10 に記載のプログラム。

【請求項 12】

ウェブアプリケーションをクライアントサイドのアプリケーションに遷移させる方法であって、コンピュータのプロセッサが、

アプリケーションに関連するアプリケーションパッケージであって、ユーザのアプリケーションに対する対話的操作を可能にするために必要な最小限のコードを含むアプリケーションパッケージを識別するステップと、

40

前記識別されたアプリケーションパッケージをローカルのストアにダウンロードするステップと、

前記ローカルのストアを含むクライアントから前記アプリケーションに関連する複数の追加のリソースのうちの 1 つに対する要求を受け取ると、前記追加のリソースをダウンロードするステップであって、前記複数の追加のリソースのうちの 1 つに対する要求は、前記アプリケーションパッケージに含まれたコードを実行する前記クライアントが、対応するユーザの操作に応じて発生させるステップと、

前記アプリケーションを前記クライアントサイドのアプリケーションに遷移させるための要求を受け取ると、残りのリソースを前記ローカルのストアにダウンロードするステッ

50

プであって、前記アプリケーションを前記クライアントサイドのアプリケーションに遷移させるための要求は、対応するユーザの操作なしに作成されるステップとを含むことを特徴とする方法。

【請求項 1 3】

前記アプリケーションを前記クライアントサイドのアプリケーションに遷移させるための要求は、前記ローカルのストアに現在格納されている追加のリソースの数に基づくことを特徴とする請求項 1 2 に記載の方法。

【請求項 1 4】

前記アプリケーションの遷移に対する要求は自律的に発生することを特徴とする請求項 1 3 に記載の方法。

【請求項 1 5】

前記アプリケーションパッケージはアプリケーションのマニフェストおよびコードを含み、前記アプリケーションのマニフェストは前記アプリケーションに関するリソースのサブセットを一意に識別することを特徴とする請求項 1 2 に記載の方法。

【請求項 1 6】

前記プロセッサが、前記追加のリソースがローカルで利用可能であるかどうかを判断するステップと、利用可能である場合、前記追加のリソースをダウンロードせずに、前記追加のリソースを利用するステップとをさらに含むことを特徴とする請求項 1 2 に記載の方法。

【請求項 1 7】

前記残りのリソースは、前記ローカルのストアに現在格納されていない複数の追加のリソースからの追加のサブセットであることを特徴とする請求項 1 2 に記載の方法。

【請求項 1 8】

前記アプリケーションを前記クライアントサイドのアプリケーションに遷移させることは、前記クライアントサイドのアプリケーションがオフラインで起動されたとき、前記クライアントサイドのアプリケーションの実行が、遷移したときの前記アプリケーションに関連付けられた状態で再開するように、前記アプリケーションに関する状態を前記クライアントサイドのアプリケーションに遷移させることを含むことを特徴とする請求項 1 2 に記載の方法。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、ソフトウェアアプリケーションのインストールに関する。

【背景技術】

【0002】

本出願は、参照により本明細書に組み込まれている、2003年5月22日に出願した Mark Alcazar、Michael Dunn、Adriaan Carter、および Prasad Tamma による「SYSTEM AND METHOD FOR PROGRESSIVELY INSTALLING A SOFTWARE APPLICATION」という表題の米国係属特許出願第 10 / 444 699 号の一部継続出願である。

【0003】

今日利用可能なアプリケーションは大きく分けて2種類ある。第1の種類のアプリケーションは、クライアントサイドアプリケーションである。クライアントサイドアプリケーションは、クライアントコンピュータに配置され、クライアントコンピュータが動作している限り利用することができる。このクライアントサイドアプリケーションは、利用できるようになるまでの間、異なるインストール状態をとる。通常、インストール状態に関して、インストール時に温度計などの何らかの形式の進行状況指示ユーザインターフェースを表示する。インストール状態では、クライアントサイドアプリケーションは利用できない。クライアントサイドアプリケーションは、ユーザがアプリケーションを使用する前に完全にインストールされていなければならない。

10

20

30

40

50

【0004】

他の種類のアプリケーションは、一般に、WebアプリケーションまたはWeb appと呼ばれる。Web appは、Webサーバに格納される。Web appは、一般に、インターネット上でアクセス可能な複数のWebページとして展開される。従来のWeb appは、マークアップ言語ベースのドキュメントを表す複数のWebページを含む。Web appは、Webページを通じてアクセスされるスクリプトまたはその他のリソースを含むこともできる。ほとんどのWeb appでは、複数のWebページおよびリソースは、Web appの「ビジネスロジック」が複数のリソース上に分散される形でハイパーリンクでつながっている。それぞれのページは、ビジネスロジック全体の一部を受け持ち、ページからページへナビゲートすることにより、ユーザはWeb app全体を体験することができる。このドキュメントの目的のために、「ナビゲート(する)」という用語は、ハイパーリンクをアクティブ化することなどによりWeb appに関連付けられているリソースをホスティング環境に読み出させることを意味する。リソースへのナビゲートでは、通常、ナビゲート先リソースがホスティング環境により読み出されるリソースである他のリソースからナビゲートすることを伴う。Web appは、インストールフェーズを必要とせず、クライアントコンピュータがWebサーバから接続を断たれると利用できなくなる。

10

【発明の開示】

【発明が解決しようとする課題】

【0005】

ソフトウェアアプリケーションと対話するこれらの方法は両方とも、利点と欠点を持ち、いずれも理想的とはいえない。

20

【課題を解決するための手段】

【0006】

本発明は、ユーザがアプリケーションとの対話を即座に開始できるようにソフトウェアアプリケーションのプログレッシブインストールを行うためのシステムおよび方法を実現する。その後、アプリケーションと対話をしている間に、アプリケーションは、ユーザのコンピュータにプログレッシブインストールされ、必要ならば、後からオフラインで使用可能にできる。プログレッシブインストール(progressive installation)は、スタートアップ状態、デマンド状態、および最終(ファイナル)状態という3つの状態を含む。これらの状態はいずれも、アプリケーションが使用不可になる専用インストールフェーズを必要としない。その代わりに、本発明のプログレッシブインストールでは、Web appが従来のWeb appとして対話できるように、またその後、ユーザとアプリケーションとの対話に影響を及ぼすことなくクライアントサイドアプリケーションに円滑に遷移できるように、2つの形態のアプリケーションインストールを混合する。

30

【0007】

本発明は、アプリケーションをプログレッシブインストールするためのメカニズムを提供する。プログレッシブインストールは、スタートアップ状態、デマンド状態、およびインストール状態という3つの状態を遷移する。スタートアップ状態では、アプリケーションに関連するコンポーネントのサブセットがダウンロードされ、ローカルデータストアに格納される。サブセットは、Webアプリケーションと類似の方法でアプリケーションを実行するのに十分なものである。デマンド状態では、アプリケーションに関連付けられたWebページ上のハイパーリンクをアクティブにした後アプリケーションに関連する追加リソースがダウンロードされる。オンデマンドリソースである追加リソースは、ローカルデータストアに格納される。オンラインリソースである追加リソースは、一時(transient)キャッシュに格納される。インストール状態では、アプリケーションは、クライアントサイドアプリケーションと似た方法で実行される。デマンド状態からインストール状態への遷移は、ユーザとアプリケーションとの対話に影響を及ぼすことなく実行される。この遷移は、ローカルデータストアに格納されている追加リソースの数に基づいて

40

50

自律的に、または外部からトリガがかかったときに、実行される。遷移の間、まだダウンロードされていない追加リソースがローカルデータストアにダウンロードされる。さらに、デマンド状態で派生する状態は、アプリケーションとともに保存され、これにより、アプリケーションはオフラインで実行されたときに同じ状態から再開することができる。

【発明を実施するための最良の形態】

【0008】

手短かに言うと、本発明は、ユーザがアプリケーションとの対話を即座に開始できるようにソフトウェアアプリケーションのプログレッシブインストールを行うためのシステムおよび方法を実現する。その後、アプリケーションと対話している間に、アプリケーションは、ユーザのコンピュータにプログレッシブインストールされ、必要ならば、後からオフラインで使用できるようにインストールされる。プログレッシブインストールは、スタートアップ状態、デマンド状態、および最終状態という3つの状態を含む。これらの状態はいずれも、アプリケーションが使用不可になる専用インストールフェーズを必要としない。その代わりに、本発明のプログレッシブインストールでは、Web appが従来のWeb appとして対話できるように2つの形態のアプリケーションインストールを併用し、その後、ユーザとアプリケーションとの対話に影響を及ぼすことなくアプリケーションをオフラインアプリケーションに円滑に遷移させるメカニズムを提供する。

【0009】

動作環境の例

図1は、本発明の一実施形態で使用できるコンピューティングデバイス例の図である。図1は、本発明の一実施例で使用できるコンピューティングデバイス例の図である。非常に基本的な構成では、コンピューティングデバイス100は、少なくとも1つの処理ユニット102およびシステムメモリ104を備えるのがふつうである。コンピューティングデバイスの正確な構成と種類に応じて、システムメモリ104は揮発性(RAMなど)、不揮発性(ROM、フラッシュメモリなど)、またはこれら2つの何らかの組合せとすることができる。システムメモリ104は、通常、オペレーティングシステム105、1つまたは複数のプログラムモジュール106を格納し、プログラムデータ107を含むこともできる。基本構成は、図1において点線108内のコンポーネントにより示されている。

【0010】

コンピューティングデバイス100は、さらに特徴または機能性を追加することもできる。例えば、コンピューティングデバイス100は、磁気ディスク、光ディスク、またはテープなどの追加データ記憶デバイス(取り外し可能および/または固定)を含むこともできる。このような追加記憶装置は、図1では、取り外し可能記憶装置109および固定記憶装置110により例示されている。コンピュータ記憶媒体は、コンピュータ可読命令、データ構造、プログラムモジュール、またはその他のデータなどの情報を格納する方法または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媒体を含むことができる。システムメモリ104、取り外し可能記憶装置109、および取り外し不可能記憶装置110は、すべてコンピュータ記憶媒体の実施例である。コンピュータ記憶媒体は、RAM、ROM、EEPROM、フラッシュメモリもしくはその他のメモリ技術、CD-ROM、デジタル多目的ディスク(DVD)もしくはその他の光ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置もしくはその他の磁気記憶デバイス、または所望の情報を格納するために使用することができコンピューティングデバイス100によりアクセスできるその他の媒体があるが、これらに限定されない。このような任意のコンピュータ記憶媒体をデバイス100の一部とすることができる。さらにコンピューティングデバイス100は、キーボード、マウス、ペン、音声入力デバイス、タッチ入力デバイスなどの入力デバイス112を含むこともできる。ディスプレイ、スピーカ、プリンタなどの出力デバイス114を含むこともできる。これらのデバイスは、当業ではよく知られているため、本明細書でさらに詳しい説明をする必要はない。

【0011】

また、コンピューティングデバイス 100 は、デバイスがネットワークなどを經由して他のコンピューティングデバイス 118 と通信するために使用する通信接続 116 も含むことができる。通信接続 116 は、通信媒体の一実施例である。通信媒体は、通常、搬送波もしくはその他のトランスポートメカニズムなどの変調されたデータ信号を介して、コンピュータ可読命令、データ構造、プログラムモジュール、またはその他のデータによって実現されることができ、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、信号内の情報を符号化する方法によりその特性のうち1つまたは複数が設定または変更された信号を意味する。例えば、通信媒体としては、有線ネットワークまたは直接配線接続などの有線媒体、ならびに、音響、RF、赤外線、およびその他の無線媒体などの無線媒体があるが、これらに限定されない。本明細書で使用されているコンピュータ可読媒体という用語は、記憶媒体および通信媒体の両方を含む。

10

【0012】

ネットワーク接続環境の例

図2は、本発明の実装を具体化できる分散ネットワーク環境の機能ブロック図概要である。図2に例示されているように、サーバ202およびクライアントコンピュータ220などの2台以上のコンピュータがネットワーク205を介して接続される。サーバ202およびクライアントコンピュータ220は、図1とともに上で説明されているデバイスなどのコンピューティングデバイスである。コンピュータは、企業環境内で接続することができ、その場合、ネットワーク205はローカルエリアネットワークまたはワイドエリアネットワークとすることができる。同様に、コンピュータは、インターネットなどのワイドエリアネットワークを介して任意に接続することができる。

20

【0013】

サーバ202は、リソースをネットワーク205に接続されている他のコンピューティングデバイスで利用できるようにする構成を採用するコンピューティングデバイスである。サーバ202は、ハイパーテキストマークアップ言語(HTML)ドキュメントなどのインターネット関連リソースに使用されるWebサービングソフトウェアを備えることができる。サーバ202は、サーバデータストア210の形態でローカル記憶装置を備える。サーバデータストア210上には、ネットワーク205上でサーバ202により利用できるようにされたリソースの少なくとも一部がある。具体的には、デプロイメントマニフェスト212は、サーバデータストア210だけでなく、アプリケーションパッケージ214および追加アプリケーションリソース216にも格納されるが、これらについては、図8~10を参照しながら以下で詳述する。サーバ202は、さらに、デプロイメントマニフェスト212だけでなく他の関連するドキュメントおよびリソースも構築し、保持するための他のアプリケーションも備える。この実装では、サーバ202は、アプリケーションパッケージ214および追加アプリケーションリソース216を他のコンピューティングデバイスからネットワーク205上で利用可能にする。

30

【0014】

クライアントコンピュータ220は、ローカルで実行されるアプリケーションを実行するとともに、ネットワーク205上で他のコンピュータに接続するように構成されたコンピューティングデバイスである。クライアントコンピュータ220は、クライアントデータストア228の形態でローカル記憶装置も備える。クライアントデータストア228には、アプリケーションストア230および一時キャッシュ232が配置される。一実施形態では、クライアントコンピュータ220上で実行されるそれぞれのアプリケーションは、関連するアプリケーションストア230を有する。クライアントコンピュータ220は、さらに、ネットワーク上で他のコンピュータと対話するため他のアプリケーションも備える。このようなアプリケーションの1つに、インターネットブラウジングソフトウェア(これ以降、ブラウザ222と呼ばれる)などのホストソフトウェア222がある。ブラウザ222は、アプリケーションパッケージハンドラ224およびリソースローダ226と通信する。アプリケーションパッケージハンドラ224は、ブラウザ222がアプリケーションパッケージ214などのアプリケーションパッケージを見つけたときに、ブラウ

40

50

ザ側でアプリケーションパッケージハンドラ 224 を呼び出すことがわかるように登録される。その後、アプリケーションパッケージハンドラ 224 は、アプリケーションパッケージ 214 を処理する。アプリケーションパッケージ 214 は、クライアントコンピュータ 220 上で関連するアプリケーションの実行を開始するための情報を含む。アプリケーションパッケージ 214 の形式は、実行可能ファイルまたはその他のパッケージングタイプ形式でよい。一実施形態では、アプリケーションパッケージハンドラ 224 は、関連する情報を取得するためにアプリケーションパッケージの形式を解読するように構成することができる。ブラウザ 222 は、さらに、クライアントコンピュータが追加アプリケーションリソース 216 などの追加リソースをサーバ 202 に要求するときリソースローダ 226 と通信する。ブラウザ 222、アプリケーションパッケージハンドラ 224、およびリソースローダ 226 により実行される処理については、図 5 ~ 7 の流れ図とともに以下で詳述する。

10

【0015】

手短に言うと、クライアントコンピュータ 220 のユーザは、従来の任意の方法でサーバ 202 に接続することができる。サーバ 202 は、サーバデータストア 210 に配置されたファイルを利用可能にする Web ページまたは他の何らかのリソースを表示する。ユーザがリンクなどを選択したことに対する応答として、サーバ 202 はデプロイメントマニフェスト 212 へナビゲートし、要求されたアプリケーションに関連するアプリケーションパッケージ 214 を識別する。以下で詳述するが、アプリケーションパッケージ 214 は、アプリケーションを起動するのに必要な最低限の量のコードを含む。アプリケーションパッケージ 214 は、サーバ 202 からクライアントコンピュータ 220 にダウンロードされる。

20

【0016】

図 3 は、本発明の一実装によりリモートアプリケーションのプログレッシブダウンロードを使用可能にする Web ブラウジングソフトウェアにより表示できる画面表示である。図 3 を手短に参照すると、上述のサーバ 202 が使用される Web ページ 310 を含むブラウザ 222 の表示例 300 が示されている。Web ページ 310 は、特定の Web app と関連付けられたリソースでもよいし、またはリモートコンピューティングシステムがソフトウェアアプリケーションをダウンロードに利用できるようにするためのリソースでもよい。Web ページ 310 は、上述のデプロイメントマニフェスト 212 を指しているハイパーリンク 360 を含む。デプロイメントマニフェスト 212 は、アプリケーションを起動するのに必要な少なくとも最小のコードを含む、アプリケーションパッケージ 214 を指している。また、Web ページ 310 は、上述のデプロイメントマニフェスト 212 を指しているハイパーリンク 380 を含む。ハイパーリンク 380 を選択することは、ユーザが現在アプリケーションを明示的に「インストール」することに関心を持っていることを示す。図 7 について以下で詳述するように、ハイパーリンク 380 を選択した後、ユーザは、アプリケーションのダウンロードまたはコンピュータへのインストールを待たずに引き続きアプリケーションと対話することができる。

30

【0017】

Web ページ 310 はインターネット、企業イントラネット、または他のネットワークアクセス可能なロケーション上で供給することができることは理解されるであろう。ハイパーリンク 360 をアクティブにすると、アプリケーションパッケージ 214 はサーバからプルダウンされる。Web ページ 310 は、ユーザがアプリケーションを呼び出せる唯一の手段であることは理解されるであろう。例えば、アプリケーションパッケージ 214 へのリンクは、電子メールメッセージなどに入れて送ることができる。

40

【0018】

手法の実例

図 4 は、本発明の一実装による、アプリケーションのプログレッシブインストールのさまざまな状態を例示する状態図 400 である。プログレッシブインストールは、呼び出し状態 402、スタートアップ状態 404、デマンド状態 406、およびインストール状態

50

410を含む。呼び出し状態402で、ユーザはアプリケーションを呼び出す。ユーザがサーバ202から供給されるリンクをクリックすることによりアプリケーションを呼び出すと、プログレッシブインストールはスタートアップ状態404に進む。しかし、後で詳述するように、アプリケーションは、クライアントコンピュータ上にすでにインストールされている可能性がある。ローカルにインストールされているアプリケーションは、ローカルアプリケーションへのリンクを選択する、スタートメニュー内のローカルアプリケーションへのショートカットを選択するなどの方法により呼び出すことができる。ローカルにインストールされているアプリケーションが呼び出されると、プロセスはインストール状態410に遷移する。他の実施形態では、呼び出し状態402からインストール状態410への遷移は、サブスクリプション更新状態412を介して進行することができる。手短に言うと、サブスクリプション更新状態412により、サーバ202上でアプリケーションの更新が利用可能かどうかを判別するということである。更新があれば、アプリケーションの更新されたコンポーネントがダウンロードされる。

10

【0019】

次に、アプリケーションがローカルにインストールされていないと仮定して、プログレッシブインストールがスタートアップ状態404に進む。図5を参照し手短に言うと、スタートアップ状態404は、アプリケーションをクライアントコンピュータ上で実行するのに必要な最小のコードをダウンロードする。最小のコードはアプリケーション丸ごとと比べてかなり小さいので、ユーザはアプリケーションとの対話をすぐに開始することができる。これは、今日従来のWeb appと対話している場合にユーザが経験することとあまり変わらない。プログレッシブインストールは、スタートアップ状態からデマンド状態406に進む。図6を参照して手短に説明すると、デマンド状態406は、必要に応じて、リソースをダウンロードする。このため、ユーザは、アプリケーションの購入またはアプリケーションとの継続関係を進展させる前にアプリケーションを試すことができる。

20

【0020】

デマンド状態406から、ユーザは、アプリケーションのローカルでのインストールに先じて、終了状態408に進むことができる。これは、ユーザがブラウザを閉じると実行される。ユーザがデマンド状態406から終了状態408に遷移すると、アプリケーションダウンロードされたコンポーネントを削除することができる。したがって、クライアントコンピュータは、ユーザがアプリケーションを呼び出す前と同じ状態となる。そこで、ユーザは、後でリモートアプリケーションを再び呼び出すことができる。プログレッシブインストールは、再び、スタートアップ状態とデマンド状態を通る。そのため、ユーザは、アプリケーションをインストールすることに踏み切らずに再びアプリケーションを使用することができる。プログレッシブインストールは、デマンド状態406からインストール状態410に進む。遷移は、購入決定、昇格許可(elevated permission)(例えば、信頼昇格(trust elevation))の要求に基づいてユーザが開始するか、または予め定められている数のリソースがすでにクライアントコンピュータ上にインストールされているときなどに、オペレーティングシステム側で自律的に実行することができる。デマンド状態からインストール状態への遷移は、ユーザとアプリケーションとの対話に影響を及ぼさない。この遷移は、図7に関して以下で説明される。

30

40

【0021】

そこで、本発明によるプログレッシブインストールを使用すると、ユーザは、アプリケーションを呼び出すと直ちにアプリケーションとの対話を開始することができる。ユーザに影響を及ぼすことなくユーザがアプリケーションと対話している間にアプリケーションの各部がダウンロードされる。ユーザが専用のインストールを待つ必要はまったくない。

【0022】

図5は、本発明の一実施形態によるプログレッシブインストールのスタートアップ状態のプロセスを一般的に例示する論理流れ図である。プロセスはブロック501から開始し、そのブロックで、ネットワークに配置されるアプリケーションはアプリケーションに関

50

係するハイパーリンクを選択するなどにより呼び出される。図5に進む前に、ネットワークに配置されるアプリケーションのコンポーネントについて図8と併せて説明する。

【0023】

図8は、サーバ202上のネットワークに配置されたアプリケーションのコンポーネントを図形的に表したものである。コンポーネントは、デプロイメントマニフェスト212、アプリケーションパッケージ214、および追加アプリケーションリソース216を含む。アプリケーションパッケージ214は、アプリケーションマニフェスト802およびコード804を含む。アプリケーションマニフェスト802には、各コンポーネント、そのバージョン、および依存関係を含むアプリケーションコードの詳細が記述される。そのような性質のサンプルアプリケーションマニフェストが、本明細書に「付録A - サンプルアプリケーションマニフェスト」として収録されている。本明細書では特定のファイルとして説明されているが、本発明のアプリケーションマニフェスト212は何らかの形のアプリケーションのコンポーネントを説明する情報を意味するものとして解釈すべきであり、本明細書で説明されている場所以外の場所に存在していてもよい。本明細書で説明されているアプリケーションマニフェスト212は例示のために用意されているにすぎない。一実施形態では、コード804は、アプリケーションを実行するのに必要な最小のコードを含む。当業者であれば、本発明から逸脱することなく必要のない追加コードをアプリケーションパッケージ214に含めることができることは理解するであろう。しかし、遅延またはユーザへの影響を少なくするために、最低限の量のコードが必要である。追加アプリケーションリソース216は、マークアップA 810、追加コード812、マークアップB 814などのオンデマンドリソースおよびオンラインリソースを含む。図8は、5つの追加リソースのみを例示しているが、当業者であれば、通常、Web appのコンポーネントであるリソースがさらにいくつかあることを理解するであろう。

10

20

【0024】

図5を参照すると、本発明の一実施形態では、ブロック502でデプロイメントマニフェスト212へのナビゲーションが行われる。一実施形態では、デプロイメントマニフェストはリモートサーバ上に配置され、これによりアプリケーションのエントリーポイントが識別される。サンプルデプロイメントマニフェストが、本明細書に「付録B - サンプルデプロイメントマニフェスト」として収録されている。

【0025】

ブロック504で、このエントリーポイントへのナビゲーションが実行される。一実施形態では、エントリーポイントは、アプリケーションマニフェストおよびアプリケーションを実行するのに必要な最低限の量のコードを含むアプリケーションパッケージ(例えば、図8に示されているアプリケーションパッケージ214)とすることができる。

30

【0026】

ブロック506で、ホストが起動される。一実施形態では、ホストはWebブラウザである。アプリケーションがクライアントコンピュータからプログレッシブインストールされる他の実施形態では、ホストはスタンドアロンのホストとすることができる。そうすると、スタンドアロンのホストは、ホストがWebブラウザである実施形態を使用して以下で説明するのと同じ方法で機能する。アプリケーションパッケージに関連するファイルタイプに対しアプリケーションパッケージハンドラが登録される。したがって、ブラウザがアプリケーションパッケージを受け取ると、アプリケーションパッケージハンドラは本発明によるプログレッシブインストールを開始することができる。一実施形態では、アプリケーションパッケージハンドラは、アプリケーションパッケージ214に関連付けられたファイルタイプについて登録されているmimeハンドラとすることができる。

40

【0027】

ブロック508で、アプリケーションを実行するために必要が最低限度の量のコードがダウンロードされる。アプリケーションパッケージ214を使用する実施形態では、これはアプリケーションパッケージ214をダウンロードすることを含む。上述のように、ブラウザはアプリケーションパッケージハンドラを呼び出してアプリケーションパッケージ

50

214を処理する。

【0028】

ブロック510で、リソースローダが登録され、アプリケーションに関連付けられる。リソースローダは、必要なときにアプリケーションリソースをロードする役割を持つ。一実施形態では、リソースローダは、アプリケーションの「コンテキストで」リソースをロードする方法が組み込まれているプラグ可能プロトコルとすることができる。

【0029】

ブロック512で、アプリケーションのアプリケーションドメインが作成される。ブロック514で、アプリケーションを構成するユーザコードが実行される。一実施形態では、アプリケーションの実行によりアプリケーションオブジェクトが作成される。アプリケーションオブジェクトを作成するために、クラスを定義するアプリケーションパッケージ内のコードが実行される。アプリケーションオブジェクトは一意的IDを持つ。さらに、アプリケーションオブジェクトは状態を含む。この状態は、デマンド状態で連続的に更新される。この状態は、ユーザとアプリケーションとの対話に関連する情報を含む。この状態情報を使用することで、アプリケーションはWebアプリケーションからクライアントアプリケーションへ滑らかに遷移することができる。

【0030】

スタートアップ状態での処理が完了する。その後、プログレッシブインストールがデマンド状態に進む。図8に示されているように、スタートアップが完了した後、アプリケーションパッケージ214はクライアントコンピュータ220のアプリケーションストア230に格納される。ユーザは、アプリケーションとの対話を開始することができる。従来のWeb appでは、Web appのリソースは、アプリケーション毎のストア230の概念もなく、一時キャッシュ232にダウンロードされていた。これからわかるように、アプリケーション毎のストア230を有することにより、本発明はWeb appからクライアントサイドアプリケーションへユーザに影響を及ぼすことなく滑らかに遷移することができる。

【0031】

図6は、プログレッシブインストールのデマンド状態のプロセスを一般的に例示する論理流れ図である。デマンド状態はブロック601から始まるが、そのブロックでスタートアップ状態が完了し、ユーザはアプリケーションと対話する。プロセス600は、アプリケーションの一部が要求されたときに必ず実行される処理を示す。これは、アセンブリロード(コード)、リソースなどの要求を含むことができる。通常、デマンド状態ではリソースおよびコードに対する多くの要求が受け取られる。それぞれのそのような要求はプロセス600を実行する。以下の説明では、要求がリソースの場合のプロセス600を議論する。当業者であれば、要求がアセンブリロードについてのものである場合にプロセス600も実行されることを理解するであろう。

【0032】

ブロック602で、要求が受け取られる。通常、ユーザがアプリケーションに関連付けられたWebページのうちの1つにあるハイパーリンクを選択すると必ず要求が発生する。処理は、決定ブロック604に続く。

【0033】

決定ブロック604で、要求がリソースについてのものであるかどうかを判別される。要求がリソースについてのものではない場合、処理は終わりまで進む。その一方で、要求がリソースについてのものである場合、処理は決定ブロック606に進む。

【0034】

決定ブロック606で、要求されたリソースがローカルで使用可能であるかどうかを判別される。リソースがローカルで使用可能である場合、使用するリソースのローカルコピーがロードされ、プロセスは終わりまで進む。リソースのタイプに応じて、ローカルコピーはアプリケーションストア内にあるか、または一時キャッシュ内にある。リソースがローカルで使用できない場合、処理は決定ブロック610に進む。

10

20

30

40

50

【 0 0 3 5 】

決定ブロック 6 1 0 で、要求されたリソースがオンデマンドリソースであるかどうか判别される。要求されたリソースがオンデマンドリソースの場合、処理はブロック 6 1 2 に進み、そこで、リソースは h t t p を介してロードされ、ローカルアプリケーションストアにキャッシュされる。例えば、図 9 で、マークアップ A 8 1 0 およびコード 8 1 2 は、アプリケーションストア 2 3 0 に格納される。プロセスは終わりまで進む。決定ブロック 6 1 0 で、リソースがオンデマンドリソースでない場合、処理は決定ブロック 6 2 0 に続く。

【 0 0 3 6 】

決定ブロック 6 2 0 で、要求がオンラインリソースであるかどうか判别される。リソースがオンラインリソースである場合、処理はブロック 6 2 2 に続き、そこで、オンラインリソースが h t t p を介してロードされる。ブロック 6 2 4 で、オンラインリソースはトランスポートキャッシュ 2 3 2 にキャッシュされる。例えば、図 9 で、マークアップ b 8 1 4 は、オンラインリソースとして指定されており、一時キャッシュに格納される。その後処理は完了する。

【 0 0 3 7 】

一実施形態では、それぞれのリソースは、何らかの形で関連付けられているリソースのグループに属することができる。例えば、一般に併用されるリソースは、1つのグループに入れることができる。このような場合、ブロック 6 1 2 および 6 2 2 で、対象リソースを含むリソースのグループ全体を読み出すことができる。この手法により、後から必要になる他のリソースがローカルにすでに存在する可能性が高まる。

【 0 0 3 8 】

そのため、デマンド状態では、追加リソースがダウンロードされ、アプリケーション毎のストアに初期値として配置されることが分かるであろう。ダウンロードされるリソースはアプリケーションをオフラインで実行するのに必要なリソースと同じであるため、後述のように、アプリケーションストアをアプリケーションオブジェクトと併用することで、本発明は、ユーザとアプリケーションとの対話に影響を及ぼすことなく、Webアプリケーションからクライアントサイドアプリケーションへ滑らかに遷移することができる。したがって、2つの異なる種類のアプリケーション(つまり、クライアントサイドアプリケーションとWebアプリケーション)を用意せずに、1種類のアプリケーションを両方の目的に使用することができる。本発明を使用することで、この1種類のアプリケーションは必要なときに一方の目的から他方の目的に滑らかに遷移する。

【 0 0 3 9 】

図 7 は、プログレッシブインストールのデマンド状態とインストール状態との間で遷移するプロセスを一般的に例示する論理流れ図である。処理はブロック 7 0 1 から開始し、そこで、アプリケーションをインストールすべきであることを知らせるトリガが発生する。トリガは、ユーザ開始トリガ、またはアプリケーション毎のストアにすでにダウンロードされているリソースの数などの外部ベンチマークに基づいて自律的なトリガとすることもできる。処理は、ブロック 7 0 2 に続く。

【 0 0 4 0 】

ブロック 7 0 2 で、残りのオンデマンドリソースが h t t p を介してダウンロードされる。処理は、ブロック 7 0 4 に続く。ブロック 7 0 4 で、これら残りのオンデマンドリソースは、アプリケーションストアに格納される。これは、ユーザがまだアプリケーションと対話している間に実行される。例えば、図 1 0 は、アプリケーションストアに現在配置されているマークアップ C 8 1 6 を例示している。処理は、ブロック 7 0 6 に続く。

【 0 0 4 1 】

ブロック 7 0 6 で、オンラインリソースのコピーがアプリケーションストア 2 3 0 に格納される。したがって、一時キャッシュ内のコピーが取り除かれても、オンラインリソースのコピーはそのまま存在する。例えば、図 1 0 を参照すると、マークアップ B 8 1 4 はアプリケーションストアに格納されているものとして例示されている。処理は、ブロッ

10

20

30

40

50

ク 7 0 8 に 続 く。

【 0 0 4 2 】

ブ ロ ッ ク 7 0 8 で、ア ク テ ィ ベ ー シ ョ ン 情 報 が オ ペ レ ー テ ィ ン グ シ ス テ ム に 記 録 さ れ る。例 えば、シ ョ ー ト カ ッ ト が ス タ ー ト メ ニ ュ ー に 追 加 さ れ る 場 合 が あ る。ア ク テ ィ ベ ー シ ョ ン 情 報 に よ り、ア プ リ ケ ー シ ョ ン が 次 回 ロ ー カ ル で 呼 び 出 さ れ た と き に 従 来 の メ カ ニ ズ ム を 使 用 し て ア プ リ ケ ー シ ョ ン を 呼 び 出 す こ と が で き る。処 理 は、ブ ロ ッ ク 7 1 0 に 続 く。

【 0 0 4 3 】

ブ ロ ッ ク 7 1 0 で、イ ン プ レ ッ シ ョ ン 情 報 (i m p r e s s i o n i n f o r m a t i o n) が オ ペ レ ー テ ィ ン グ シ ス テ ム に 記 録 さ れ る。イ ン プ レ ッ シ ョ ン 情 報 は、フ ェ イ ル 関 連 付 け な ど の、ア プ リ ケ ー シ ョ ン と オ ペ レ ー テ ィ ン グ シ ス テ ム と の 相 互 作 用 を 記 述 す る も の で あ る。さ ら に、イ ン プ レ ッ シ ョ ン 情 報 で は、ア プ リ ケ ー シ ョ ン の 変 更 / プ ロ グ ラ ム エ ン ト リ か ら の 削 除 の 仕 方 に つ い て も 記 述 す る。図 1 0 を 参 照 す る と、ア ク テ ィ ベ ー シ ョ ン 情 報 8 3 2 お よ び イ ン プ レ ッ シ ョ ン 情 報 8 3 4 は、ク ラ イ ア ン ト コ ン プ ュ ー タ 2 2 0 の オ ペ レ ー テ ィ ン グ シ ス テ ム 情 報 8 3 0 内 に 示 さ れ て い る。そ の 後 処 理 は 完 了 す る。

10

【 0 0 4 4 】

上 述 の よ う に、こ の 時 点 で、ア プ リ ケ ー シ ョ ン は オ フ ラ イ ン で 利 用 可 能 に な っ て い る。上 述 の 説 明 を 読 む と わ か る よ う に、ユ ー ザ は ア プ リ ケ ー シ ョ ン の イ ン ス ト ー ル を 待 つ 必 要 が な か っ た。デ マ ン ド 状 態 で 生 成 さ れ る 情 報 は、イ ン ス ト ー ル 状 態 に 移 行 さ れ る。し た が っ て、ユ ー ザ が ア プ リ ケ ー シ ョ ン と 対 話 し て い る 間 に 格 納 さ れ た ア プ リ ケ ー シ ョ ン の I D お よ び 状 態 情 報 を 使 用 す る こ と に よ り、ア プ リ ケ ー シ ョ ン は イ ン ス ト ー ル 状 態 に 滑 ら か に 遷 移 す る こ と が で き る。

20

【 0 0 4 5 】

ア プ リ ケ ー シ ョ ン プ ロ グ ラ ミ ン グ イ ン タ ー フ ェ ー ス の 実 施 例

具 体 的 な 一 実 施 例 で は、上 述 の 手 法 は、ダ ウ ン ロ ー ド お よ び イ ン ス ト ー ル、サ ー ビ ス、信 頼 と プ ラ イ バ シ ー の 確 立、な ら び に ア プ リ ケ ー シ ョ ン の 最 終 的 実 行 を 管 理 す る た め の 機 能 性 を 公 開 す る ア プ リ ケ ー シ ョ ン プ ロ グ ラ ミ ン グ イ ン タ ー フ ェ ー ス (A P I) で 実 装 す る こ と が で き る。こ の よ う な A P I の 一 実 施 例 に つ い て、こ れ ら の 各 機 能 を 処 理 す る た め の D e p l o y m e n t M a n a g e r の 以 下 の ク ラ ス 例 で 説 明 す る。こ の D e p l o y m e n t M a n a g e r の ス ケ ル ト ン タ イ プ の 定 義 を 以 下 に 示 す。

30

【 0 0 4 6 】

* * * * *

```
public class DeploymentManager
{
    public DeploymentManager(string identity, string codebase)
    { ... }

    //呼び出すことができる非同期オペレーションに対するイベント。
    public event BindCompletedEventHandler
BindCompleted;
    public event
DeterminePlatformRequirementsCompletedEventHandler
DeterminePlatformRequirementsCompleted;
    public event
DetermineAuthorizationCompletedEventHandler
DetermineAuthorizationCompleted;
    public event SynchronizeCompletedEventHandler
SynchronizeCompleted;
    public event ExecuteCompletedEventHandler
ExecuteCompleted;
```

40

50

//すべての非同期オペレーションに対するProgressChangedイベント

```
public event DeploymentProgressChangedEventHandler
DeploymentProgressChanged;
```

```
//BindAsync
public void BindAsync(object userToken)
{ ... }
```

10

```
//DetermineRequirementsAsync
public void
DeterminePlatformRequirementsAsync(object userToken)
{ ... }
```

```
// DetermineAuthorizationAsync
public void DetermineTrustAsync(object userToken)
{ ... }
```

```
// SynchronizeAsync
public void SynchronizeAsync(object userToken)
{ ... }
```

20

```
// ExecuteAsync
public void ExecuteAsync(object userToken)
{ ... }
```

```
// AsyncCancel
public void AsyncCancel()
{ ... }
```

30

```
}
```

```
* * * * *
```

DeploymentManagerは、(1) マニフェストバインディング、(2) プラットフォーム要件の判別、(3) 権限の判別、(4) 同期処理、および(5) 実行の5つの主要オペレーションのためのメソッドを公開することに留意されたい。これらの関数について、それぞれ、簡単にここで説明することにする。

【0047】

マニフェストバインディング

マニフェストバインディングは、展開されたアプリケーションに関する必要なマニフェストメタデータをインストール、サービス、およびアクティベーションのために最初に取得するプロセスである。バインディングは、一般に、デプロイメントマニフェストへのコードベース、または展開されたアプリケーションのID、または場合によってはそれらの両方のから始まる。バインディングでは、展開されたアプリケーションに関して後から決定するためのマニフェスト情報の最小セットを取り出す。マニフェストバインドでは、バインド、アプリケーションストアの状態、およびバインド先のアプリケーションのコンテキストに応じてネットワーク接続を必要とする場合もあれば必要としない場合もある。アプリケーションがすでにマシンに展開されている場合、バインドは、ネットワークI/Oなしの完全オフライン状態で成功することができる。

40

【0048】

プラットフォーム要件の判別

50

バイディングが完了した後、クライアントマシンのインストール状態を問い合わせアプリケーションを実行するために必要なプラットフォームが存在しているかどうかを判別できるようになる。プラットフォームは、アプリケーションが依存しているが、展開済みアプリケーションインストールの一部としてインストールすることができないソフトウェアとして識別することができる。展開されたアプリケーションは、アプリケーションマニフェストに記述されているこれらの依存関係を参照する。例えば、オペレーティングシステムの最小バージョン、ランタイム環境の最小バージョン、GAC常駐アセンブリの特定のバージョンについてサポートを提供することもできる。バージョンが最小として処理されるかどうかは、アセンブリがプラットフォームまたはライブラリとしてマーク付けされるかどうかによって依存する場合がある。

10

【0049】

プラットフォーム要件が満たされている場合、アプリケーションインストールは続行する。プラットフォーム要件が満たされていない場合、どのプラットフォーム依存関係が満たされていなかったかに関する特定の情報とともに失敗が返される。アプリケーションインストールプロセスは、その後、続行できないが、マシンに必要なプラットフォームをインストールするアクションが実行された後そのプロセスを繰り返すことができる。

【0050】

権限の判別

アプリケーションにどのような信頼、プライバシー、およびライセンスが受け入れられるかに関する決定を、このような情報を展開およびアプリケーションマニフェスト内に配置することもできるため、マニフェストバイディングが完了した後に行うことができる。これらの決定は、権限の一般的カテゴリの下にグループ化される。例えば、アプリケーションが既定のサンドボックスで実行できない場合に、権限がユーザプロンプティングを必要とすることがある。権限の付与が成功すると、クライアントマシン上でアプリケーションを実行するのに必要な一組の許可付与、プライバシーポリシー保証、ライセンスキーなどが生成される。この情報は、アプリケーションがアクティブにされているときに決定を後で繰り返す必要がないようにキャッシュしておくことができる。権限が失敗した場合、アプリケーションインストールは続行できない。

20

【0051】

同期処理

プラットフォームおよび権限の判別が首尾よく完了すると、アプリケーションペイロードをインストールする実際のタスクを開始できる。このプロセスは、同期処理と呼ばれる。同期処理は、展開済みアプリケーションの必要なバージョンがすでにマシン上に存在していることを単に確認するだけで成功する。それとは別に、同期処理では、リモートで展開されているアプリケーション（オンデマンドコンポーネント、言語パックとともに必要）またはアプリケーションを起動するために必要な最小限度必要サブセットのみ（例えば、必要なコンポーネントのみ）の完全ダウンロードを伴う場合がある。ダウンロードが発生した場合、同期処理はさらに、（1）アプリケーションペイロードがディスク上の一時記憶域に複製される純粋なトランスポートフェーズおよび（2）展開済みアプリケーションが格納処理され、バイディングに利用可能にされる、コミットオペレーションの2つのフェーズに細分される。同時処理は、既存のインストール済みアプリケーションのオプションコンポーネントまたはオンデマンドコンポーネントのダウンロードに使用することもできる。

30

40

【0052】

実行

アプリケーションがプラットフォームおよび権限の決定を正常に完了し、ペイロードの同期処理が正常に行われ、格納のコミットが行われた後、アプリケーションを実行（起動（`launch`）または実行（`run`））できる。実行は、別のスタンドアロンのプロセスで行うか、または既存の呼び出し側のプロセスを使用して行うことができる。両方の場合に、実行ホストは、場合によっては再プロンプティングを回避するために前回の権限呼

50

び出しですでにすでにキャッシュされている決定を利用して、アプリケーションの安全な実行環境を提供する。

【 0 0 5 3 】

クライアントサイドの実装

非同期完了結果などの `DeploymentManager` メソッドの呼び出しからの通知を受け取るために、クライアントは、関連する完了イベントハンドラ（例えば、`BindCompletedEventHandler` など）の実装を提供する。これらは、関連付けられた完了イベント引数（例えば、`BindCompletedEventArgs`）を受け渡される。これらのオペレーションに対する非同期の進行結果を受け取るために、クライアントは `DeploymentProgressChangedEventHandler` の実装を提供する。これは、引数（`DeploymentProgressChangedEventArgs`）を受け渡される。以下に、そのようなイベントハンドラ実装の実施例のスケルトンを示す。

10

【 0 0 5 4 】

```
* * * * *
// イベントハンドラデリゲートおよび引数をバインドする。
```

【 0 0 5 5 】

```
public delegate void BindCompletedEventHandler(object
sender, BindCompletedEventArgs e);
public class BindCompletedEventArgs :
AsyncCompletedEventArgs
{
    public BindCompletedEventArgs(Exception error, bool
cancelled, object userToken) : base(error, cancelled,
userToken)
    { ... }
}
```

20

```
// DeterminePlatformRequirements イベントハンドラデリゲートおよび引数。
```

【 0 0 5 6 】

```
public delegate void
DeterminePlatformRequirementsCompletedEventHandler(object
sender,
DeterminePlatformRequirementsCompletedEventArgs e);
public class
DeterminePlatformRequirementsCompletedEventArgs :
AsyncCompletedEventArgs
{
    public
DeterminePlatformRequirementsCompletedEventArgs(Exceptio
n error, bool cancelled, object userToken) : base(error,
cancelled, userToken)
    { ... }
}
```

30

40

```
// DetermineAuthorization イベントハンドラデリゲートおよび引数。
```

【 0 0 5 7 】

```
public delegate void
DetermineAuthorizationCompletedEventHandler(object
sender, DetermineAuthorizationCompletedEventArgs e);
public class DetermineAuthorizationCompletedEventArgs :
```

50

```
AsyncCompletedEventArgs
```

```
{
    public
    DetermineAuthorizationCompletedEventArgs(Exception
error, bool cancelled, object userToken) : base(error,
cancelled, userToken)
    { ... }
}
```

```
//イベントハンドラデリゲートおよび引数を同期処理する。
```

10

```
【 0 0 5 8 】
```

```
public delegate void
SynchronizeCompletedEventHandler(object sender,
SynchronizeCompletedEventArgs e);
public class SynchronizeCompletedEventArgs :
AsyncCompletedEventArgs
{
    public SynchronizeCompletedEventArgs(Exception
error, bool cancelled, object userToken) : base(error,
cancelled, userToken)
    { ... }
}
```

20

```
//イベントハンドラデリゲートおよび引数を実行する。
```

```
【 0 0 5 9 】
```

```
public delegate void ExecuteCompletedEventHandler(object
sender, ExecuteCompletedEventArgs e);
public class ExecuteCompletedEventArgs :
AsyncCompletedEventArgs
{
    public ExecuteCompletedEventArgs(Exception error,
bool cancelled, object userToken) : base(error,
cancelled, userToken)
    { ... }
}
```

30

```
//DeploymentProgressChangedイベントハンドラデリゲートおよび引数。
```

```
【 0 0 6 0 】
```

```
public delegate void
DeploymentProgressChangedEventHandler(object sender,
DeploymentProgressChangedEventArgs e);
public class DeploymentProgressChangedEventArgs :
ProgressChangedEventArgs
{
    public DeploymentProgressChangedEventArgs(object
userToken, int progressPercentage) : base(userToken,
progressPercentage)
    { ... }
}
```

40

```
* * * * * * * * * * * * * * * * * * * * * * * *
```

50

説明したばかりのメカニズムを呼び出すクライアント（例えば、アプリケーション）の一実装の一般的な実施例を以下に示す。以下の実施例は、上述の `DeploymentManager` を実装するクラス（クライアント）に基づくアプリケーションである。

【0061】

```

* * * * *
public class Client
{
    public static void Main()
    {
        DeploymentManager dep = new
DeploymentManager("name=bar, version=1.0.0.0",
"http://www.foo.com/bar.deploy");
        dep.DeploymentProgressChanged += new
DeploymentProgressChangedEventHandler(Client.ProgressCha
nged);
        dep.BindCompleted += new
BindCompletedEventHandler(Client.BindCompleted);

        dep.BindAsync(null);
    }
    public static void BindCompleted(object sender,
BindCompletedEventArgs e)
    {
        System.Console.WriteLine(e.ToString());
    }
    public static void ProgressChanged(object sender,
DeploymentProgressChangedEventArgs e)
    {
        System.Console.WriteLine(e.ProgressPercentage);
    }
}
* * * * *

```

上記の明細、実施例、およびデータは、本発明の実装の構造および使用に関する完全な説明となっている。本発明の多くの実施形態は、本発明の精神と範囲を逸脱することなく実装できるため、本発明は付属の請求項によって定められる。

【0062】

付録A．サンプルアプリケーションマニフェスト

【0063】

【数 1 - 1】

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0" xmlns:asmv2="urn:schemas-
microsoft-com:asm.v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1
assembly.adaptive.xsd">
```

```
<assemblyIdentity name="MyDocViewer" version="1.0.0.0"
processorArchitecture="x86" asmv2:culture="en-us"
publicKeyToken="0000000000000000" />
```

10

```
<entryPoint name="main" xmlns="urn:schemas-microsoft-
com:asm.v2" dependencyName="MyDocViewer">
  <commandLine file="MyDocViewer.exe" parameters="" />
</entryPoint>
```

```
<TrustInfo xmlns="urn:schemas-microsoft-com:asm.v2"
xmlns:temp="temporary">
```

20

```
<Security>
  <ApplicationRequestMinimum>
    <PermissionSet
class="System.Security.PermissionSet" ID="FullTrust"
version="1" Unrestricted="true" />
    <AssemblyRequest name="MyDocViewer"
PermissionSetReference="FullTrust" />
  </ApplicationRequestMinimum>
</Security>
</TrustInfo>
```

30

```
<file name="Sample4.xaml"
hash="75966580bf63a6f7d9818bcbf6c8c61343e61d9f"
hashalg="SHA1" asmv2:size="636" />
<file name="Sample5.xaml"
hash="9fe4e7312498c0b62ab455b289e27fc2fc8b3bb3"
hashalg="SHA1" asmv2:size="615" />
<file name="Sample6.xaml"
hash="760221281e78c621f45947b97b87e65a2bee6e14"
hashalg="SHA1" asmv2:size="2750" />
```

```
<dependency asmv2:name="MyDocViewer">
  <dependentAssembly>
```

40

【 0 0 6 4 】

【数 1 - 2】

```
<assemblyIdentity name="MyDocViewer"
version="0.0.0.0" publicKeyToken="f745653e2b97409d"
processorArchitecture="x86" />
  </dependentAssembly>
  <asmv2:installFrom codebase="MyDocViewer.exe"
hash="95f2246ac74b3f32938db0ebed313e38ee7b4b5b"
hashalg="SHA1" size="12288" />
  </dependency>
</assembly>
```

10

【 0 0 6 5】

付録 B . サンプルデプロイメントマニフェスト

【 0 0 6 6】

【数 2】

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0" xmlns:asmv2="urn:schemas-
microsoft-com:asm.v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1
assembly.adaptive.xsd">
  <assemblyIdentity name="MyDocViewer.deploy"
version="1.0.0.0" processorArchitecture="x86"
asmv2:culture="en-us" publicKeyToken="0000000000000000"
/>
  <description asmv2:publisher="MyCompany"
asmv2:product="WCP Application of MyDocViewer"
asmv2:supportUrl="http://www.mycompany.com/AppServer/MyD
ocViewer/support.asp" />
  <deployment xmlns="urn:schemas-microsoft-com:asm.v2"
isRequiredUpdate="false">
    <install shellVisible="true" />
    <subscription>
      <update>
        <beforeApplicationStartup />
        <periodic>
          <minElapsedTimeAllowed time="6" unit="hours"
/>
          <maxElapsedTimeAllowed time="1" unit="weeks"
/>
        </periodic>
      </update>
    </subscription>
  </deployment>
  <dependency>
    <dependentAssembly>
      <assemblyIdentity name="MyDocViewer"
version="1.0.0.0" processorArchitecture="x86"
asmv2:culture="en-us" publicKeyToken="0000000000000000"
/>
    </dependentAssembly>
    <asmv2:installFrom codebase="MyDocViewer.manifest"
/>
  </dependency>
</assembly>

```

【図面の簡単な説明】

【0067】

【図1】本発明の一実施形態で使用できるコンピューティングデバイス例の図である。

【図2】本発明の実装を具体化できる分散ネットワーク環境の機能ブロック図概要である。

【図3】本発明の一実装によりアプリケーションのプログレッシブダウンロードを使用可能にするためWebブラウジングソフトウェアにより表示できる画面表示である。

【図4】本発明の一実装による、アプリケーションのプログレッシブインストールのさまざまな状態を例示する状態図である。

【図5】プログレッシブインストールのスタートアップ状態のプロセスを一般的に例示する論理流れ図である。

【図6】プログレッシブインストールのデマンド状態のプロセスを一般的に例示する論理流れ図である。

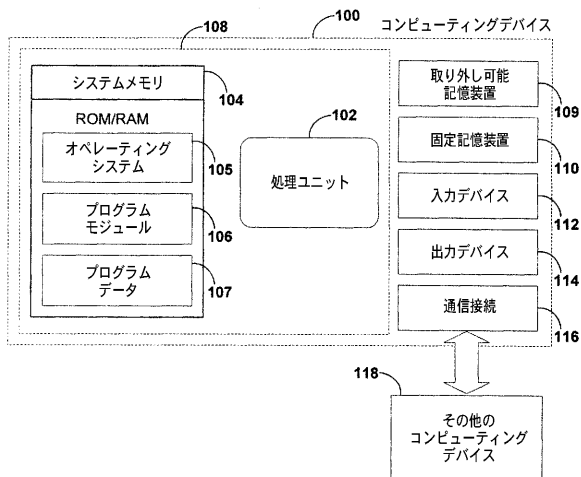
【図7】プログレッシブインストールのデマンド状態とインストール状態との間で遷移するプロセスを一般的に例示する論理流れ図である。

【図8】本発明の一実装による、プログレッシブインストールでロードされるファイルを一般的に例示する一連のブロック図である。

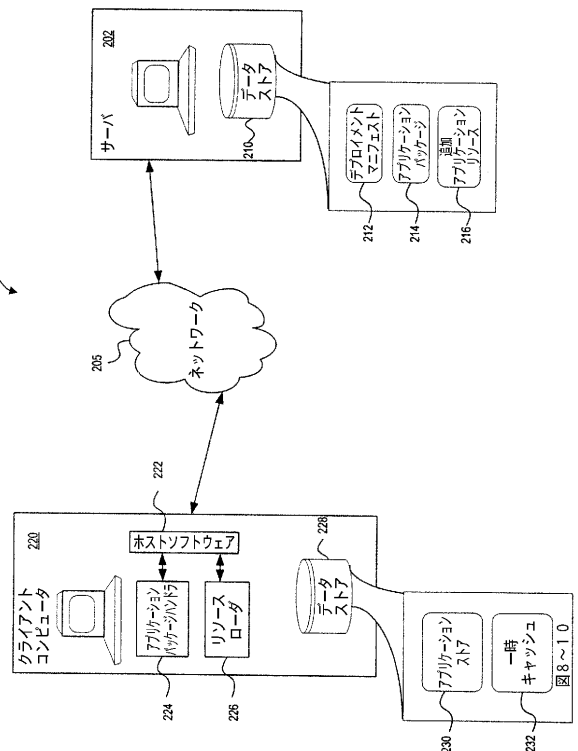
【図9】本発明の一実装による、プログレッシブインストールでロードされるファイルを一般的に例示する一連のブロック図である。

【図10】本発明の一実装による、プログレッシブインストールでロードされるファイルを一般的に例示する一連のブロック図である。

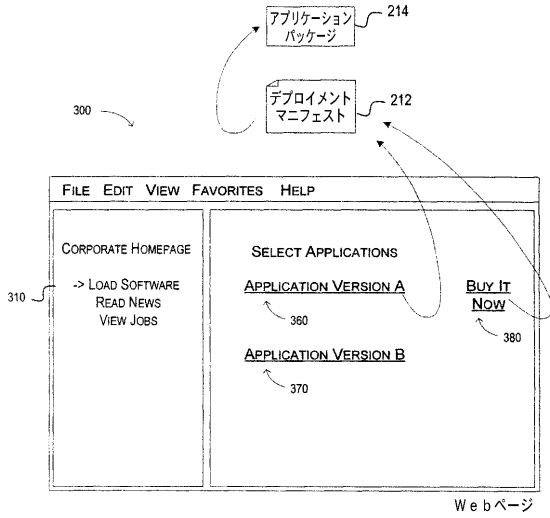
【図1】



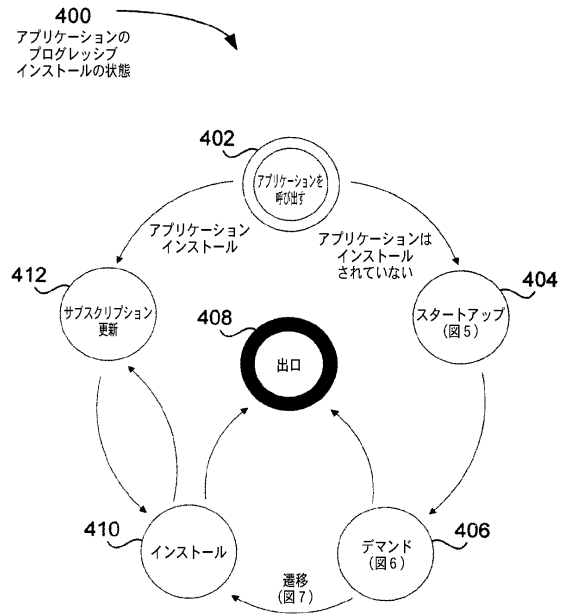
【図2】



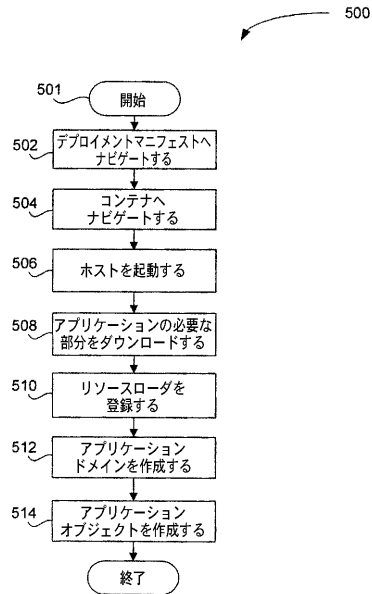
【図3】



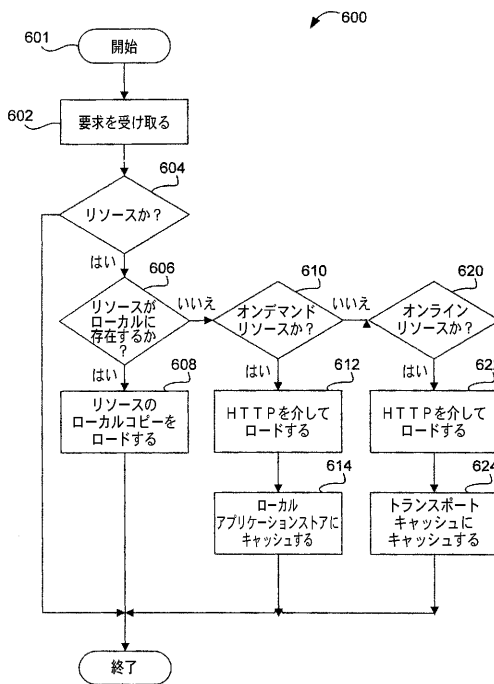
【図4】



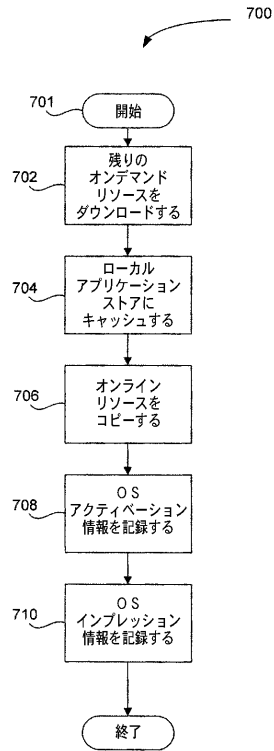
【図5】



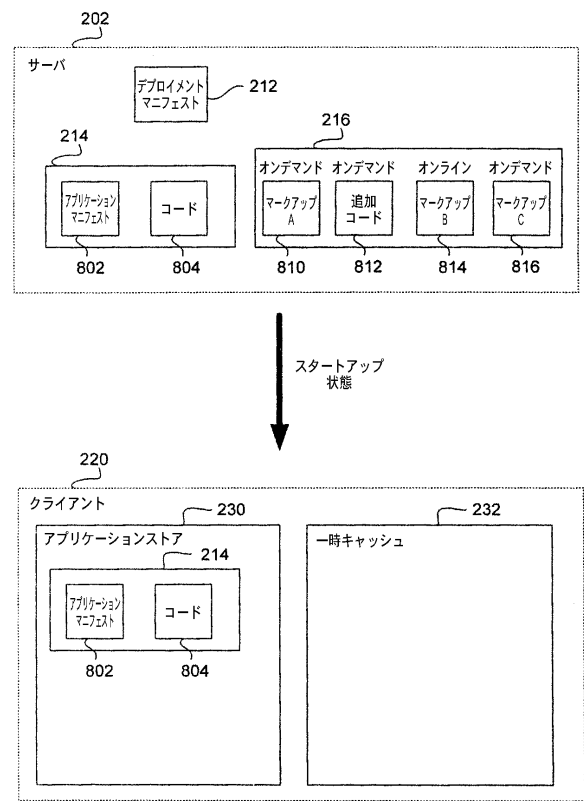
【図6】



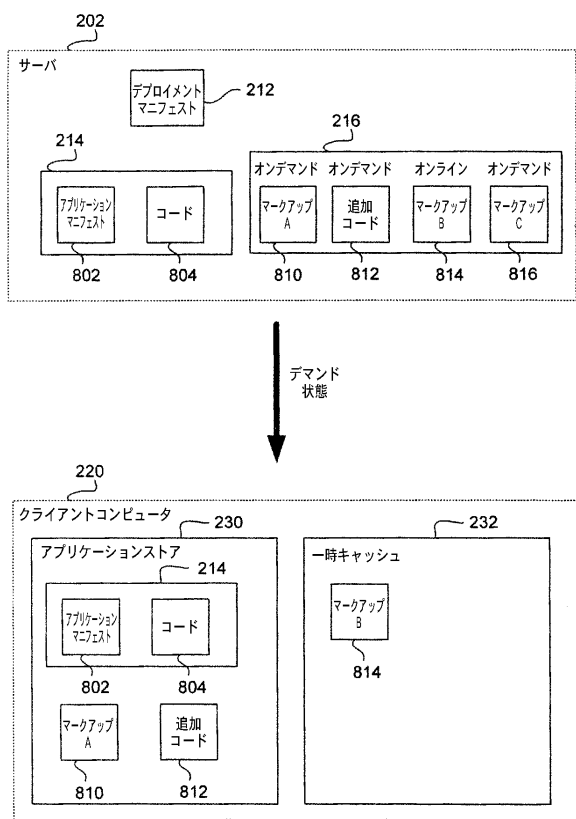
【図7】



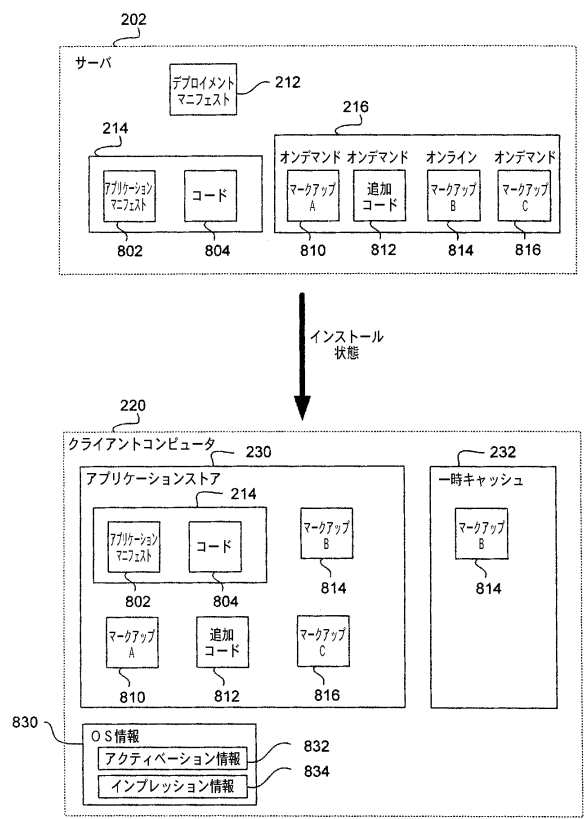
【図8】



【図9】



【図10】



フロントページの続き

- (72)発明者 マイケル ダン
アメリカ合衆国 98052 ワシントン州 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 コーポレーション内
- (72)発明者 アドリアン ダブリュ.カンター
アメリカ合衆国 98052 ワシントン州 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 コーポレーション内
- (72)発明者 ヴェンカタ ラマ プラサド タムマナ
アメリカ合衆国 98052 ワシントン州 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 コーポレーション内

審査官 久保 光宏

- (56)参考文献 特開2003-208316(JP,A)
特開2003-169372(JP,A)
国際公開第01/098936(WO,A1)
特開2003-177932(JP,A)
日経Windows 2000, 日本, 日経BP社, 2001年 1月 1日, 2000年1月号(通巻第4
6号), 第244頁, ISSN:1345-2835
「Oracle Application Server リリース4.0.8 EJB,ECO/JavaおよびCORBAアプリケーション開発
者ガイド J00088-01」, 日本, ORACLE, 1999年10月, 第1版, 第4-13~4-16頁
社団法人情報処理学会編, 「新版 情報処理ハンドブック」, 日本, 株式会社オーム社, 199
5年11月25日, 第1版, 第711頁, ISBN:4-274-07832-9
「ソリューションの軌跡 キリンビール医薬事業本部のNeWS(MR活動支援システム)」, 日経オ
ープンシステム, 日本, 日経BP社, 2000年 8月15日, 2000年8月号(第89号), 第1
46~151頁, ISSN:0918-581X

- (58)調査した分野(Int.Cl., DB名)
G06F9/445,
G06F13/00,
G06F15/00,
CSDB(日本国特許庁)