(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4717426号
(P4717426)
(45)発行日　平成23年7月6日(2011.7.6)　(24)登録日　平成23年4月8日(2011.4.8)

(51)Int.Cl.　F I
G06F 9/48 (2006.01)　G06F 9/46 452H
G06F 9/46 457

請求項の数 8　(全 16 頁)

(21)出願番号　特願2004-354696(P2004-354696)
(22)出願日　平成16年12月7日(2004.12.7)
(65)公開番号　特開2006-163840(P2006-163840A)
(43)公開日　平成18年6月22日(2006.6.22)
審査請求日　平成19年11月26日(2007.11.26)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(74)代理人　100130409
弁理士　下山　治
(74)代理人　100134175
弁理士　永川　行光

最終頁に続く

(54)【発明の名称】情報処理装置及びその方法

(57)【特許請求の範囲】
【請求項１】
プラットフォームがプロセスごとに生成される情報処理装置であって、
ソフトウェアモジュールのインストールを行うプラットフォームによりインストールされたソフトウェアモジュールに設定された属性を、前記ソフトウェアモジュールを起動する際に参照するモジュール属性参照手段と、
前記モジュール属性参照手段により参照された前記属性が、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは異なるプロセス上で実行することを示す別プロセス生成属性を含む場合、前記ソフトウェアモジュールを、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは別のプロセスとして生成された、当該ソフトウェアモジュールを実行するためのプラットフォーム上で実行するように起動する制御手段とを有し、
前記ソフトウェアモジュールは、前記ソフトウェアモジュールの持つ一部の機能を他のソフトウェアモジュールに提供する機能を有することを特徴とする情報処理装置。
【請求項２】
前記制御手段は、前記モジュール属性参照手段により参照された前記属性が、当該ソフトウェアモジュールをインストールしたプラットフォームと、前記別のプロセスとして生成された、当該ソフトウェアモジュールを実行するためのプラットフォーム上で実行される前記ソフトウェアモジュールとが連携することを示すプロセス間連携属性を含む場合に、前記ソフトウェアモジュールをインストールしたプラットフォームと、別のプロセスと

して生成されたプラットフォーム上で実行される前記ソフトウェアモジュールとを連携させることを特徴とする請求項１に記載の情報処理装置。
【請求項３】
オペレーティングシステム上で直接実行されるようにインストールされるアプリケーションに、オペレーティングシステム上で直接実行されるソフトウェアモジュールとしての属性を付与する手段と、
前記属性が付与されたアプリケーションを、前記ソフトウェアモジュールをインストールしたプラットフォームによる管理対象として取り込む手段とを更に有することを特徴とする請求項１又は２に記載の情報処理装置。
【請求項４】
プラットフォームがプロセスごとに生成される情報処理方法であって、
ソフトウェアモジュールのインストールを行うプラットフォームによりインストールされたソフトウェアモジュールに設定された属性を、前記ソフトウェアモジュールを起動する際に参照するモジュール属性参照工程と、
前記モジュール属性参照工程で参照された前記属性が、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは異なるプロセス上で実行することを示す別プロセス生成属性を含む場合、前記ソフトウェアモジュールを、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは別のプロセスとして生成された、プラットフォーム上で実行するように起動する制御工程と、
前記ソフトウェアモジュールの持つ一部の機能を他のソフトウェアモジュールに提供する提供工程と、
を有することを特徴とする情報処理方法。
【請求項５】
前記制御工程では、前記モジュール属性参照工程により参照された前記属性が、当該ソフトウェアモジュールをインストールしたプラットフォームと、前記別のプロセスとして生成された、当該ソフトウェアモジュールを実行するためのプラットフォーム上で実行される前記ソフトウェアモジュールが連携することを示すプロセス間連携属性を含む場合に、前記ソフトウェアモジュールをインストールしたプラットフォームと、別のプロセスとして生成されたプラットフォーム上で実行される前記ソフトウェアモジュールとを連携させることを特徴とする請求項４に記載の情報処理方法。
【請求項６】
オペレーティングシステム上で直接実行されるようにインストールされるアプリケーションに、オペレーティングシステム上で直接実行されるソフトウェアモジュールとしての属性を付与する工程と、
前記属性が付与されたアプリケーションを、前記ソフトウェアモジュールをインストールしたプラットフォームによる管理対象として取り込む工程とを更に有することを特徴とする請求項４又は５に記載の情報処理方法。
【請求項７】
コンピュータに、請求項４乃至６のいずれか１項に記載の情報処理方法を実行させるためのプログラム。
【請求項８】
請求項７に記載のプログラムを記録した、コンピュータにより読み取り可能な記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プラットフォームがプロセスごとに生成される情報処理装置とその方法に関するものである。
【背景技術】
【０００２】

マイクロコンピュータを組み込んで機器制御を行ういわゆる組み込み機器では、マイクロコンピュータで実行されるプログラムは、マスクＲＯＭなどに書き込まれており書き換えられないことが多かった。このためプログラムにバグがあった場合にも、そのプログラムを変更できず、バグを抱えたまま稼動しつづけなければならない場合が多かった。そのため、書き換え可能なフラッシュＲＯＭなどにプログラムを格納し、プログラムの書き換えに対応できる機器が出てきている。しかしこの場合もプログラム全体を変更することが多く、パーソナルコンピュータのように一部を変更したり、新たな機能を追加したりといったことが気軽に行なえるような構成にはなっていなかった。

【０００３】

記憶容量や処理能力に乏しい組み込み機器においても、パーソナルコンピュータのようにプログラムの更新や新機能の追加を気軽に行ない、組み込み機器でも変化の激しい業務処理に対応し得るものでなければならないようになってきている。そのための技術としてＪａｖａ技術（Ｊａｖａはプログラミング言語を含むソフトウェア技術であり、米国Sun Microsystems Inc.の登録商標）を使用し、Ｊａｖａの持つプログラムの可搬性を利用して、拡張可能なアプリケーションを提供する技術が提案されている（特許文献１及び非特許文献１）。この技術によれば、１つのアプリケーションの上で実行されるため、アプリケーション単位で機能を拡張するに比べてハードウェア資源の要求量が少なくて済む。

【０００４】

非特許文献１のOSGi Service Platform（以下、ＯＳＧｉと略）は、アプリケーションに追加されるソフトウェアモジュールをバンドルと呼び、バンドルのインストール／起動／終了／更新／アンインストールといったソフトウェアモジュールのライフサイクルを管理する仕様を定義している。またＯＳＧｉでは、アプリケーションが停止したときの各バンドルの実行状況を記憶し、アプリケーションの再起動時に各バンドルの状態が停止時の状態に戻される機能も備えている。

【特許文献１】特開２０００－２９７１３号公報

【非特許文献１】OSGi Service Platform Specification Release 2, OSGi, 2001

【発明の開示】

【発明が解決しようとする課題】

【０００５】

以前は豊富なハードウェア資源を持ち得なかった機器でも、ハードウェア技術の進歩や製造技術の進歩などにより、より豊富な記憶容量、処理能力を得られるようになってきている。それに伴って、機能拡張として後から追加されるソフトウェアモジュールも大規模で複雑な処理を行なうようになってきている。

【０００６】

従来は、ソフトウェアを管理しているオペレーティングシステムからは、ソフトウェアモジュールは一つのアプリケーションとしてしか扱われない。このため、一つのソフトウェアモジュールの不具合による影響を他のソフトウェアモジュールに波及させないために、オペレーティングシステムが元々備えているソフトウェア分離実行機能（ソフトウェアをプロセスとして異なる空間で実行する機能）を使用することができない。そのため、高度な処理を実行することにより不具合を発生させる可能性が高まったソフトウェアモジュールが不具合を起こして、アプリケーション全体を機能不全に陥らせる可能性が高くなっている。

【０００７】

機器のハードウェア資源が豊富になったことにより、より高機能なオペレーティングシステムを機器に導入して複数のアプリケーションを同時に実行できるようになったため、追加機能を別アプリケーション（プロセス）として実行させることにより、問題の解決を図ることができるようになってきた。

【０００８】

しかしながら、一つのアプリケーションに対して機能を拡張するときの操作手順と、別のアプリケーションとして機能を拡張する場合の操作手順とは異なったものになる。従っ

て、同じ拡張機能のライフサイクル管理という目的をもった操作手順が複数存在することになる。また、アプリケーションの追加でのみ機能を追加する場合には、操作方法が変更されるためユーザを混乱させるといった問題や、それまで購入したり開発したりしていた機能拡張用のソフトウェアモジュールが全く使用できなくなり、ソフトウェア資産の継承性が犠牲になってしまうといった問題が存在する。

【０００９】

本発明は上記問題点に鑑みてなされたもので、本願発明の特徴は、プロセスごとに生成されるフレームワークにおいて、ソフトウェアモジュールを別プロセスとして実行するように設定できる機能を付加できる情報処理装置及びその方法を提供することにある。

【課題を解決するための手段】

【００１０】

本発明の一態様に係る情報処理装置は以下のような構成を備える。即ち、

プラットフォームがプロセスごとに生成される情報処理装置であって、

ソフトウェアモジュールのインストールを行うプラットフォームによりインストールされたソフトウェアモジュールに設定された属性を、前記ソフトウェアモジュールを起動する際に参照するモジュール属性参照手段と、

前記モジュール属性参照手段により参照された前記属性が、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは異なるプロセス上で実行することを示す別プロセス生成属性を含む場合、前記ソフトウェアモジュールを、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは別のプロセスとして生成された、当該ソフトウェアモジュールを実行するためのプラットフォーム上で実行するように起動する制御手段とを有し、

前記ソフトウェアモジュールは、前記ソフトウェアモジュールの持つ一部の機能を他のソフトウェアモジュールに提供する機能を有することを特徴とする。

【００１２】

本発明の一態様に係る情報処理方法は以下のような工程を備える。即ち、

プラットフォームがプロセスごとに生成される情報処理方法であって、

ソフトウェアモジュールのインストールを行うプラットフォームによりインストールされたソフトウェアモジュールに設定された属性を、前記ソフトウェアモジュールを起動する際に参照するモジュール属性参照工程と、

前記モジュール属性参照工程で参照された前記属性が、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは異なるプロセス上で実行することを示す別プロセス生成属性を含む場合、前記ソフトウェアモジュールを、当該ソフトウェアモジュールをインストールしたプラットフォームのプロセスとは別のプロセスとして生成された、プラットフォーム上で実行するように起動する制御工程と、

前記ソフトウェアモジュールの持つ一部の機能を他のソフトウェアモジュールに提供する提供工程と、を有することを特徴とする。

【発明の効果】

【００１４】

本発明によれば、プロセスごとに生成されるフレームワークにおいて、ソフトウェアモジュールを、そのソフトウェアモジュールをインストールしたプラットフォームのプロセスとは別のプロセスとして生成されたプラットフォーム上で実行でき、ソフトウェアモジュールの一部の機能を他のソフトウェアモジュールに提供して高度な処理を実行するようにしても、ソフトウェアモジュールでの不具合の発生によりアプリケーション全体を機能不全に陥らせる可能性が高くならないようにできるという効果がある。

【発明を実施するための最良の形態】

【００１５】

以下、添付図面を参照して本発明の好適な実施の形態を詳しく説明する。

【００１６】

［実施の形態１］

図１は、本発明の実施の形態に係るコンピュータ機器の構成を示すブロック図で、このコンピュータ機器は、例えばＰＣやワークステーションなどの情報処理装置である。

【００１７】

図において、制御部１０１０はＣＰＵを有し、この機器全体の動作を制御している。一次記憶部１０２０は、各種データや制御部１０１０で実行されるソフトウェアを一時的に記憶するための記憶部で、ＲＡＭなどで構成されている。二次記憶部１０３０は、各種データや制御部１０１０で実行されるソフトウェアを恒久的に記憶している記憶部で、例えばハードディスクなどで構成されている。入力部１０４０は、ユーザの指示や文字データや画像データを入力するためキーボード、マウス（ポインティングデバイス）、タッチスクリーン、スキャナなどを有する。表示部１０５０は、オペレータへの指示やデータの内容等を表示する液晶などの表示パネルを有する。ネットワークインターフェース部１０６０は、ＬＡＮ等のネットワークを介して他の機器と通信を行なう。バス１０７０は、上述した各部を接続し、それらの間でのデータや制御信号をやり取りするシステムバスである。

【００１８】

図２は、本実施の形態に係るコンピュータ機器の制御部１０１０で実行されるソフトウェアの構成を階層を用いて表した図である。

【００１９】

図２において、オペレーティングシステム（ＯＳ）２０１０は、アプリケーションソフトウェアやハードウェア資源を管理し、このＯＳ２０１０上で各種ソフトウェアが実行される。２０２０は、ソフトウェアモジュールを拡張可能なＪａｖａ言語で記述されたアプリケーションであるOSGi Service Platformの仕様を実装したプラットフォームである。Java VM（Ｊａｖａ言語を解釈して実行する仮想マシン）２０３０は、プラットフォーム２０２０を実行するためのＪａｖａプログラム専用の仮想マシンである。２０４０はプラットフォーム２０２０上で実行されるＪａｖａ言語で記述されたソフトウェアモジュール（以下、バンドル(Bundle)と呼ぶ）であるバンドル群である。２０５０は、オペレーティングシステム２０１０上で稼動するその他のアプリケーション群を示している。尚、Java VM２０３０は、オペレーティングシステム２０１０からは、その他のアプリケーション群２０５０に属するアプリケーションと全く同等に扱われる。

【００２０】

図３は、本実施の形態に係るコンピュータ機器においてバンドル２０４０をインストールする際の処理を説明するフローチャートで、この処理を実行するプログラムは一次記憶部１０２０に記憶され、制御部１０１０の制御の下に実行される。

【００２１】

図４（Ａ）は、インストールされるバンドルファイルが暗号化されていないときの形式を説明する図、図４（Ｂ）は、Ｊａｒファイルの属性を記述しているマニフェストの一例を示す図である。

【００２２】

バンドルファイル４０１０は、Ｊａｖａ言語のＪａｒファイル形式である。このＪａｒファイルは、ＺＩＰ形式のアーカイブファイルであり、複数のファイルエントリ４０２０の列で構成されている。各ファイルエントリ４０２０は、そのファイルエントリの始まりを示すマジックナンバー４０３０と、ファイルエントリサイズ４０４０、ファイル名サイズ４０５０、ファイル名４０６０及びファイルコンテンツ４０７０で表される各フィールドを有している。尚、図４（Ａ）では、説明を簡単にするためにＺＩＰファイルの仕様を簡略化して示している。

【００２３】

このＪａｒファイルの各種属性はマニフェストと呼ばれ、ファイルエントリ内に格納されている。このマニフェストのファイル名は、図４（Ｂ）に示すように「META-INF/MANIFEST.MF」であり、このマニフェスト内には属性の名前と、その属性値とのペア形式で属性値が記述されている。

【００２４】
図４（Ｂ）において、「Bundle-Activator」は、インストールした後、一番最初に起動するJavaのクラスの名称を定義する。「Bundle-Classpath」は、クラスを探索する際のJarファイルの検索順序を示している。「NeedProcess」は、このバンドルの実行が別のプロセスで実行されることを示している。「Import-Package」は、他のバンドルからExportされているパッケージを使用している際に、それを定義する。「Import-Service」は他のバンドルの一部の機能を他に提供することを示す。
【００２５】
次に図３のフローチャートを参照して、本実施の形態に係るコンピュータ機器におけるバンドル２０４０のインストール処理を説明する。
【００２６】
まずステップＳ１で、インストールするように指定されたファイルの格納場所と同じ場所からインストールされたバンドルが既に存在しているか、即ち、そのバンドルファイルが既にインストールされているかを調べる。存在している場合は処理を終了し、存在していない場合はステップＳ２に進む。ステップＳ２では、そのバンドルファイルをプラットフォーム２０２０にインストールし、そのインストールされた、暗号化されているバンドルファイル４０１０を復号する。この復号に用いる暗号キーは、秘密鍵としてプラットフォーム２０２０が管理している。
【００２７】
次にステップＳ３で、ステップＳ２で復号したバンドルファイル４０１０を、バンドルを管理するためのディレクトリに格納する。次にステップＳ４で、プラットフォーム２０２０上のバンドル管理のためのバンドル管理オブジェクトを生成する。
【００２８】
図５は、本実施の形態に係るバンドル管理オブジェクトの構造を示す図である。
【００２９】
図において、バンドル識別子５０１０は、個々のバンドルを識別するための識別子である。ロケーション５０２０は、このバンドルがどこからインストールされたかを示す情報でありＵＲＬを使用して表される。ステータス５０３０は、このバンドルの実行状態を記述している。例えば、インストールされている場合は「INSTALLED」、実行中は「ACTIVE」がセットされる。属性リスト５０４０は、このバンドルの属性を記述している。即ち、図４（Ｂ）に示すマニフェストファイルで記述された属性を記憶している。プロセス識別子５０５０は、このバンドルが別プロセスとして実行されたときに付与される識別子である。この別プロセスとは、ソフトウェアモジュールがインストールされたアプリケーションが実行されているプロセスとは異なるプロセス上で、そのソフトウェアモジュールが実行されることを意味している。
【００３０】
そして図３のステップＳ５に進み、ステップＳ４で生成したバンドル管理オブジェクトの属性リスト５０４０に、バンドルファイル４０１０内のマニフェストに記述されていた属性を属性名をキーとして検索できるように全て登録しているハッシュテーブルインスタンスとして設定する。またこのとき、バンドル管理オブジェクトのステータス５０３０に、このバンドルがインストールされている状態であることを示す状態値（「INSTALLED」）を設定する。
【００３１】
以上説明した処理により、バンドル２０４０がインストールされる。尚ここで、別プロセスとして実行されるバンドル２０５０（図２）と、プラットフォーム２０２０上で実行されるバンドル２０４０のインストール処理においては、バンドル管理オブジェクトの属性リスト５０４０に、実行時に別プロセスで実行することを指示するための「NeedProcess」属性を持っているかどうかだけが異なるだけで、インストールの処理としては同じである。
【００３２】

図６は、本実施の形態に係るコンピュータ機器におけるバンドルの起動処理を説明するフローチャートで、この処理を実行するプログラムは一次記憶部１０２０に記憶され、制御部１０１０の制御の下で実行される。

【００３３】

まずステップＳ１１で、指定されたバンドルのバンドル管理オブジェクトのステータス５０３０が「ACTIVE」（実行中）であるかどうかを判定し、「ACTIVE」である場合は既に実行されているので処理を終了する。「ACTIVE」でない場合はステップＳ１２に進み、バンドル管理オブジェクトの属性リスト５０４０に、別プロセスで実行することを指示する「NeedProcess」が設定されているかどうかを調べる。ここで属性リスト５０４０はハッシュテーブルインスタンスとして構成されているため、「NeedProcess」属性の名前をキーとして指定することにより、「NeedProcess」属性値を得ることができる。別プロセスで実行することを示す「NeedProcess」属性が設定されている場合はステップＳ１３に進み、そうでない場合はステップＳ１６に進む。ステップＳ１６では、起動しようとしているバンドルは通常のプラットフォーム２０２０上のバンドルであるため、プラットフォーム２０２０上のバンドルとしての起動処理を行なう。こうして起動処理が終了すると、バンドル管理オブジェクトのステータス５０３０に、実行中であることを示す「ACTIVE」状態値を設定する。また、バンドル管理オブジェクトのプロセス識別子５０５０には、プラットフォーム２０２０自身のプロセス識別子を設定する。

【００３４】

また別プロセスで実行する場合はステップＳ１３で、ステップＳ１２で得た「NeedProcess」属性値を調べ、起動するバンドルとプラットフォーム２０２０との間で協調動作が必要かどうかを調べる。協調動作が必要なバンドルには、「NeedProcess」属性値として「Interactive」を指定していなければならない。従って、「NeedProcess」属性として指定されたものが「Interactive」以外である場合は、協調動作が必要ではないと認識してステップＳ１５に進み、別プロセスとして起動する。

【００３５】

一方、ステップＳ１３で、協調動作が必要であると判断した場合はステップＳ１４に進み、連携用の通信路を形成し、別プロセスとして起動する。

【００３６】

図７は、本実施の形態で実施される協調動作を説明する図である。

【００３７】

図において、プラットフォームＡ７０１０は、前述のインストール処理において、インストール指示を受けたプラットフォームとする。７０２０は、このプラットフォームＡ７０１０を実行するためのJava VM A（Ｊａｖａ仮想マシーン Ａ）である。プラットフォームＢ７０３０は、プラットフォームＡ７０１０上のバンドルの「NeedProcess」属性値として「Interactive」が指定されている場合に、指定プラットフォームＡ７０１０によって協調動作のために生成されたプラットフォームである。７０４０はプラットフォームＢ７０３０を実行するためのJava VM B（Ｊａｖａ仮想マシーン Ｂ）である。バンドル７０５０は、プラットフォームＡ７０１０にインストールされているが、別プロセス（ここではプラットフォームＢ７０３０）で実行され、実行時にプラットフォームＡ７０１０と連携するように指定されているバンドルである。

【００３８】

またバンドル７０７０は、プラットフォームＡ７０１０とは別プロセスで実行され、実行時にプラットフォームＡ７０１０と連携する必要が無いと指定されてインストールされているバンドルである。７０６０は、バンドル７０７０を実行するためのJava VM C（Ｊａｖａ仮想マシーン Ｃ）である。

【００３９】

このような場合、前述のステップＳ１３では、バンドル７０５０が、プラットフォームＡ７０１０上のバンドルと別プロセスで実行され、かつプラットフォームＡ７０１０との協調動作が必要であると認識されることになる。

【００４０】
この場合、プラットフォームＢ７０３０が使用する実行環境がまだ構築されていなければ、その実行環境を構築する。こうして構築された実行環境は、プラットフォームＡ７０１０内に、バンドル７０５０のバンドル識別子５０１０をキーとして参照されるように記憶している。
【００４１】
またプラットフォームＢ７０３０が再起動される場合に、プラットフォームＢ７０３０の停止時の状態を再現するため、どのようなバンドルがインストールされているかを示す情報、及びプラットフォームＢ７０３０が停止した時の各バンドルの状態が記憶されている。この再起動のために記憶されている情報のみを複製し、その他は新規に作成することにより、プラットフォームＢ７０３０の実行環境を作成することができる。そして、その複製した再起動のための情報を変更し、バンドル７０５０の情報だけを残してその他のバンドルの情報を全て削除する。即ち、バンドル７０５０のみがインストールされて実行されている状態になるような情報が生成される。その情報を用いて、プラットフォームＡ７０１０から見て別のプラットフォームであるプラットフォームＢ７０３０をJava VM Ｂ７０４０とともに起動する。これにより、バンドル７０５０のみがインストールされて実装されている別のプラットフォームＢ７０３０が起動される。
【００４２】
ＯＳＧｉでは、バンドルからプラットフォームにアクセスする際のインタフェースを「BundleContext」として、またバンドルを、他のバンドルやプラットフォームからアクセスするためのインタフェースを「Bundle」として定義している。
【００４３】
プラットフォームＡ７０１０とプラットフォームＢ７０３０との間の連携は、これら２つのインタフェースを基に、Ｊａｖａのプロセス間通信のための技術であるＲＭＩ（Remote Method Invocation）を使用できるように変更されたインタフェースを介して行なわれる。これによりプラットフォームＢ７０３０上で実行されるバンドル７０５０は、あたかもプラットフォームＡ７０１０上で実行されるかのように管理される。
【００４４】
ＯＳＧｉの仕様では、更に、バンドルの持つ一部の機能を他のバンドルに提供するためのサービス機能が定義されている。このサービス機能は、サービスのインタフェースをＪａｖａのインタフェースとして定義し、そのＪａｖａインタフェースを実装したサービスオブジェクトをレジストリに登録し、Ｊａｖａインタフェースをキーとしてレジストリ内で検索してサービスオブジェクトを受け取り、そのサービスを利用する形態をとっている。
【００４５】
本実施の形態では、別プロセスで実行するバンドルが提供するサービス、又は別プロセスで実行するバンドルが利用するサービスは、プロセス間で通信を行なうため、Remoteインタフェースを実装し、全てのメソッドがRemoteException例外を発生し得るように定義されたインタフェースのみを対象とするという制限を加える。
【００４６】
バンドル管理オブジェクトのプロセス識別子５０５０には、起動されたプラットフォームＢ７０３０のプロセスを識別する識別子が設定される。
【００４７】
図７において、バンドル７０７０は、図６のステップＳ１２で別プロセスで実行すると判定されるが、ステップＳ１３で、プラットフォームＡ７０１０とは連携を取る必要が無いと判断されている。このバンドル７０７０を起動する場合、プラットフォームＡ７０１０を起動する際のクラスパスに、バンドル７０７０を構成するＪａｒファイルを追加したものをクラスパスとし、「Bundle-Activator」属性として指定されたクラスを、バンドル管理オブジェクトの属性リスト５０４０から取得する。そして、この取得したクラスをエントリクラスとして指定し、Java VM Ｃ７０６０を起動することによりバンドル７０７０

を起動する。この場合、「Bundle-Activator」として指定されたクラスには、以下のシグネチャを持つメソッドが定義されていなければならない。

【００４８】

public static void main(String [] args)

本実施の形態では、バンドル７０７０はＪａｖａ言語で記述されたものとしたが、オペレーティングシステム２０１０が実行可能な形式であれば、このようなＪａｖａ言語で記述されたものでなくても良い。この場合、実行可能なバンドルのファイルとしてＪａｒファイルが使用できない場合が多いため、バンドルとしてインストールされたＪａｒファイル内に実行可能ファイルを内包し、実行時にＪａｒファイルから取り出して実行するよう構成することができる。

【００４９】

こうして生成されたプロセスの識別子を、バンドル管理オブジェクトのプロセス識別子５０５０に設定する。

【００５０】

以上説明したようにして、プラットフォーム上にインストールされているバンドルを起動することができる。

【００５１】

次に、バンドルを停止する際の処理について説明する。

【００５２】

図８は、本実施の形態に係るバンドルの停止処理を説明するフローチャートで、この処理を実行するプログラムは一次記憶部１０２０に記憶され、制御部１０１０の制御の下で実行される。

【００５３】

まずステップＳ２１で、バンドル管理オブジェクトのステータス５０３０が「ACTIVE」（実行中）であるかを調べ、「ACTIVE」でない場合は既に停止しているものとして処理を終了する。「ACTIVE」である場合はステップＳ２２に進み、バンドル管理オブジェクトの属性リスト５０４０に「NeedProcess」属性が設定されているかどうかを調べる。「NeedProcess」属性が設定されている場合はステップＳ２３へ、そうでない場合はステップＳ２６に進む。ステップＳ２６では、停止しようとしているバンドルは通常のプラットフォーム２０２０上のバンドル２０４０であるため、プラットフォーム２０２０上のバンドルとしての停止処理を行なう。この停止処理が終了すると、バンドル管理オブジェクトのステータスフィールド５０３０に停止中であることを示す「INSTALLED」状態値を設定する。その後処理を終了する。

【００５４】

一方、ステップＳ２３では、ステップＳ２２で得た「NeedProcess」属性値を調べ、実行されているバンドルとプラットフォームとの間で協調動作が必要となっているどうかを調べる。協調動作が必要な場合はステップＳ２４へ、そうでない場合はステップＳ２５へ進む。ステップＳ２４では、指定されたバンドルが起動されているプラットフォーム（図７の例では、プラットフォームＢ７０３０）をバンドル管理オブジェクトのプロセス識別子５０５０にて識別し、この識別したプラットフォームに対して、プラットフォームのシャットダウン処理を指示する。このプラットフォームのシャットダウン処理では、プラットフォーム上で動いているバンドルの状態を二次記憶部１０３０に格納し、再起動時に、その状態を回復できるようにし、プラットフォームを停止してステップＳ２５へ進む。ステップＳ２５では、バンドル管理オブジェクトのプロセス識別子５０５０に記述されているプロセス識別子を使用して、オペレーティングシステム２０１０に依頼して、そのプロセス識別子で識別されているプロセスを終了してステップＳ２６に進み、プラットフォーム２０２０上のバンドルとしての停止処理を行なう。

【００５５】

以上のようにしてバンドルの停止処理を行なう。

【００５６】

最後に、バンドルをアンインストールする際の処理について説明する。

【００５７】

図９は、バンドルをアンインストールする際の処理を説明するフローチャートで、この処理を実行するプログラムは一次記憶部１０２０に記憶され、制御部１０１０の制御の下で実行される。

【００５８】

まずステップＳ３１で、指定されたバンドルが停止しているかをバンドル管理オブジェクトのステータス５０３０が「ACTIVE」かどうかにより調べる。ステータス５０３０に設定されている状態が「ACTIVE」でなければ停止中であると判断してステップＳ３３に進むが、実行中であればステップＳ３２に進み、図８を参照して説明したバンドルの停止処理を行なってステップＳ３３に進む。

【００５９】

ステップＳ３３では、バンドル管理オブジェクトの属性リスト５０４０に「NeedProcess」属性が設定されているかどうかを調べる。「NeedProcess」属性が設定されている場合はステップＳ３４へ進み、そうでない場合はステップＳ３６へ進む。ステップＳ３４では、ステップＳ３３で得た「NeedProcess」属性値を調べ、指定されたバンドルとプラットフォームとの間で協調動作が必要かどうかを調べる。協調動作が必要な場合はステップＳ３５へ進み、そうでない場合はステップＳ３６に進む。ステップＳ３５では、起動時に生成したプラットフォーム（図７の例では、プラットフォームＢ７０３０）用の実行環境を削除してステップＳ３６に進み、フレームワーク上で通常のバンドルのアンインストール処理を行なう。このとき、インストール時に生成したファイルなどバンドルに関連する資源を全て削除する。

【００６０】

以上のようにして、バンドルのアンインストール処理を行なう。

【００６１】

このように、バンドルの属性として別プロセスで実行することが指定されている場合には別プロセスでバンドルを実行させ、管理は拡張可能なアプリケーションで行なうことが可能となる。

【００６２】

［実施の形態２］

次に、本発明の実施の形態２について説明する。

【００６３】

本実施の形態２は、前述の実施の形態１のプラットフォームに対してインストールされておらず、直接オペレーティングシステム２０１０上で実行可能なアプリケーションとしてインストールされていたものを、プラットフォーム上でアプリケーション管理を可能にする方法を示すものである。従って、コンピュータ機器の構成、ソフトウェアの構成、データ形式等は前述の実施の形態１と同じであるため、その説明を省略する。

【００６４】

図１０は、本発明の実施の形態２に係るインストール処理を説明するフローチャートで、この処理を実行するプログラムは一次記憶部１０２０に記憶され、制御部１０１０の制御の下で実行される。

【００６５】

まずステップＳ４１で、インストールするように指定されたファイルの格納場所と同じ場所からインストールされているバンドルが既に存在しているかを調べる。存在している場合は処理を終了し、存在していない場合はステップＳ４２に進み、インストールすべきデータを復号する。この復号に使用する暗号化キーは、プラットフォームが記憶している。次にステップＳ４３で、インストールするように指定されたファイルが格納されている位置を調べる。この格納位置はＵＲＬで指定されるので、格納されている位置がネットワークで繋がった装置に存在するか否かは、ＵＲＬのホスト情報を調べることで判別できる。ファイル名ではなくストリームデータとしてインストールすべきバンドルのデータを渡

されることがあるが、このときには同一装置内に無いものと認識する。こうしてステップＳ４３で、同一装置内に存在すると判断した場合はステップＳ４４へ進み、そうでなければステップＳ４５に進む。

【００６６】

ステップＳ４４では、指定されたファイルがバンドルファイルであるかどうかを調べる。ここではＺＩＰファイルであり、マニフェストファイルエントリが存在し、バンドル特有のマニフェストヘッダが存在する場合にバンドルファイルであると認識する。そして、バンドルファイルであると判断した場合はステップＳ４５へ進み、そうでない場合はステップＳ４８へ進む。

【００６７】

ステップＳ４５では、復号されたバンドルファイルを、プラットフォームがバンドル管理を行なうためにバンドルファイルを格納しておくディレクトリに格納する。次にステップＳ４６で、通常のバンドルとしてバンドル管理オブジェクトを生成する。この場合ステータス５０３０は、「INSTALLED」に設定される。そしてステップＳ４７に進む。

【００６８】

一方、ステップＳ４４で、バンドルファイルでないときはステップＳ４８に進み、バンドル管理オブジェクトの属性リスト５０４０に「NeedProcess」属性を、オペレーティングシステム２０１０上で実行可能であることを示す「Native」を属性値として持つようにして設定する。こうすることにより、対象となるバンドルが別プロセスで実行され、且つプラットフォームとは連携を取らずに実行されるようになる。ステップＳ４７では、ステップＳ４６又はステップＳ４８で作成したバンドル管理オブジェクトから、バンドルオブジェクトを作成してバンドル管理に使用できるように記憶する。尚、バンドルの起動処理、終了処理、アンインストール処理は、前述の実施の形態１の場合と同等である。

【００６９】

こうすることにより、オペレーティングシステム２０１０用にインストールされているアプリケーションもフレームワークＡ７０１０上で管理することができる。

【００７０】

以上、本発明の実施の形態を詳述したが、本発明は、複数の機器から構成されるシステムに適用しても良いし、または一つの機器からなる装置に適用しても良い。

【００７１】

なお本発明は、前述した実施の形態の機能を実現するソフトウェアのプログラムを、システム或いは装置に直接或いは遠隔から供給し、そのシステム或いは装置のコンピュータが、その供給されたプログラムコードを読み出して実行することによっても達成される場合を含む。その場合、プログラムの機能を有していれば、その形態はプログラムである必要はない。従って、本発明の機能処理をコンピュータで実現するために、該コンピュータにインストールされるプログラムコード自体も本発明を実現するものである。つまり、本発明には、本発明の機能処理を実現するためのコンピュータプログラム自体も含まれる。その場合、プログラムの機能を有していれば、オブジェクトコード、インタプリタにより実行されるプログラム、ＯＳに供給するスクリプトデータ等、プログラムの形態を問わない。

【００７２】

プログラムを供給するための記憶媒体としては、例えば、フロッピー（登録商標）ディスク、ハードディスク、光ディスク、光磁気ディスク、ＭＯ、ＣＤ－ＲＯＭ、ＣＤ－Ｒ、ＣＤ－ＲＷ、磁気テープ、不揮発性のメモリカード、ＲＯＭ、ＤＶＤ（ＤＶＤ－ＲＯＭ，ＤＶＤ－Ｒ）などがある。その他のプログラムの供給方法としては、クライアントコンピュータのブラウザを用いてインターネットのホームページに接続し、該ホームページから本発明のコンピュータプログラムそのもの、もしくは圧縮され自動インストール機能を含むファイルをハードディスク等の記憶媒体にダウンロードすることによっても供給できる。また本発明のプログラムを構成するプログラムコードを複数のファイルに分割し、それぞれのファイルを異なるホームページからダウンロードすることによっても実現可能であ

る。つまり本発明の機能処理をコンピュータで実現するためのプログラムファイルを複数のユーザに対してダウンロードさせるＷＷＷサーバも、本発明のクレームに含まれるものである。

【００７３】

また、本発明のプログラムを暗号化してＣＤ－ＲＯＭ等の記憶媒体に格納してユーザに配布し、所定の条件を満足するユーザに対してインターネットを介してホームページから暗号化を解く鍵情報をダウンロードさせ、その鍵情報を使用することにより暗号化されたプログラムを実行してコンピュータにインストールさせて実現することも可能である。

【００７４】

またコンピュータが、読み出したプログラムを実行することによって、前述した実施形態の機能が実現される他、そのプログラムの指示に基づき、コンピュータ上で稼動しているＯＳなどが、実際の処理の一部または全部を行ない、その処理によっても前述した実施形態の機能が実現され得る。

【００７５】

さらに、記録媒体から読み出されたプログラムが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書き込まれた後、そのプログラムの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部または全部を行ない、その処理によっても前述した実施形態の機能が実現される。

【図面の簡単な説明】

【００７６】

【図１】本発明の実施の形態に係るコンピュータ機器の構成を示すブロック図である。

【図２】本実施の形態に係るコンピュータ機器の制御部で実行されるソフトウェアの構成を階層を用いて表した図である。

【図３】本実施の形態に係るコンピュータ機器においてバンドルをインストールする際の処理を説明するフローチャートである。

【図４】インストールされるバンドルファイルが暗号化されていないときの形式を説明する図（Ａ）、図４（Ｂ）は、Ｊａｒファイルの属性を記述しているマニフェストの一例を示す図である。

【図５】本実施の形態に係るバンドル管理オブジェクトの構造を示す図である。

【図６】本実施の形態に係るコンピュータ機器におけるバンドルの起動処理を説明するフローチャートである。

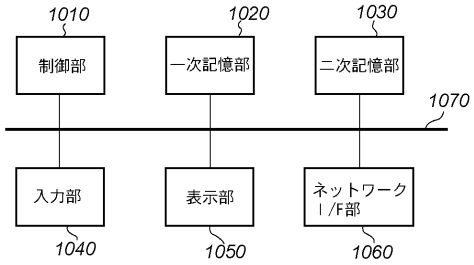
【図７】本実施の形態で実施される協調動作を説明する図である。

【図８】本実施の形態に係るバンドルの停止処理を説明するフローチャートである。

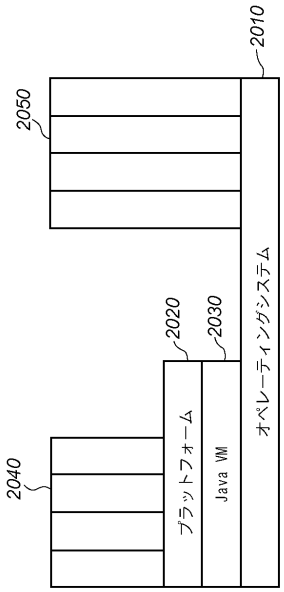
【図９】本発明の実施の形態１に係るバンドルをアンインストールする際の処理を説明するフローチャートである。

【図１０】本発明の実施の形態２に係るインストール処理を説明するフローチャートである。

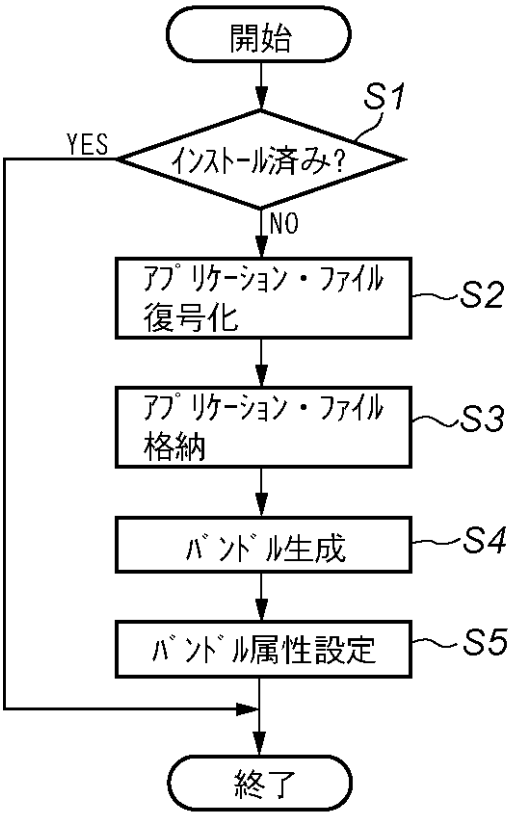
【図１】


1010
制御部
1020
一次記憶部
1030
二次記憶部
1070
入力部
1040
表示部
1050
ネットワーク
I/F部
1060

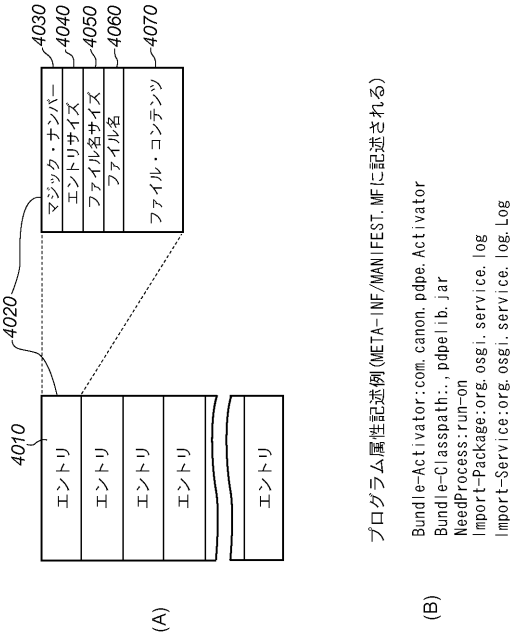
【図２】


2010
2050
オペレーティングシステム
2020
2030
プラットフォーム
Java VM
2040

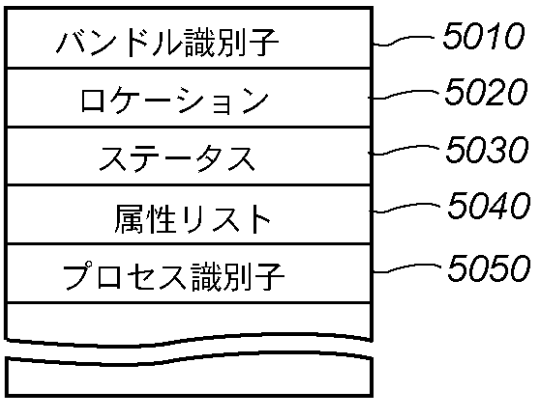
【図３】


開始
S1
インストール済み?
YES
NO
アプリケーション・ファイル
復号化
S2
アプリケーション・ファイル
格納
S3
バンドル生成
S4
バンドル属性設定
S5
終了

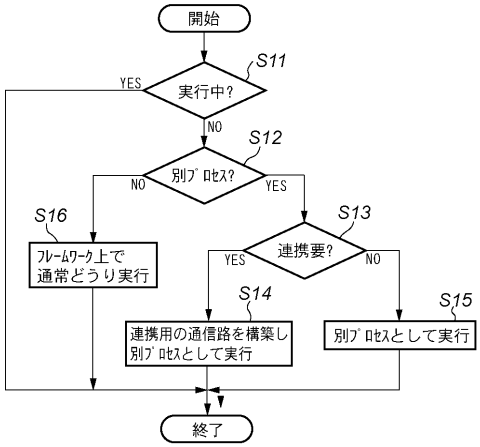
【図４】


4030
4040
4050
4060
4070
マジック・ナンバー
エントリサイズ
ファイル名サイズ
ファイル名
ファイル・コンテンツ
4020
4010
エントリ
エントリ
エントリ
エントリ
エントリ
(A)
プログラム属性記述例(META-INF/MANIFEST.MFに記述される)
Bundle-Activator:com.canon.pdpe.Activator
Bundle-Classpath:.,pdpelib.jar
NeedProcess:run-on
Import-Package:org.osgi.service.log
Import-Service:org.osgi.service.log.Log
.....
(B)

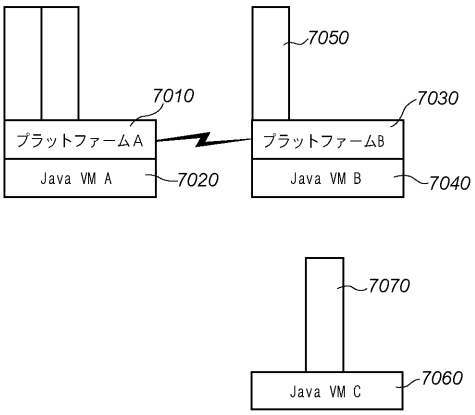
【図５】


バンドル識別子
ロケーション
ステータス
属性リスト
プロセス識別子
5010
5020
5030
5040
5050

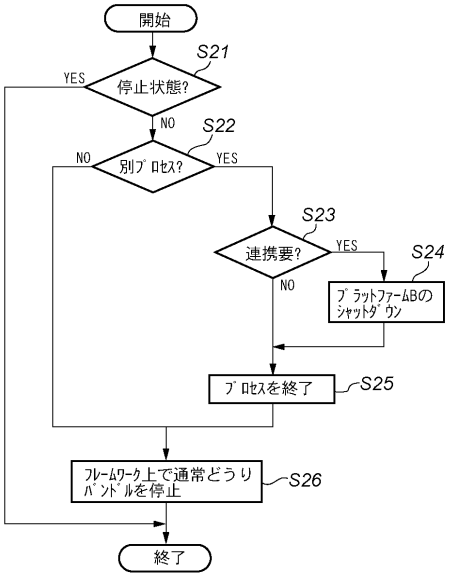
【図６】


開始
S11
実行中?
YES
NO
S12
別ﾌﾟﾛｾｽ?
NO
YES
S16
ﾌﾚｰﾑﾜｰｸ上で
通常どうり実行
S13
連携要?
YES
NO
S14
連携用の通信路を構築し
別ﾌﾟﾛｾｽとして実行
S15
別ﾌﾟﾛｾｽとして実行
終了

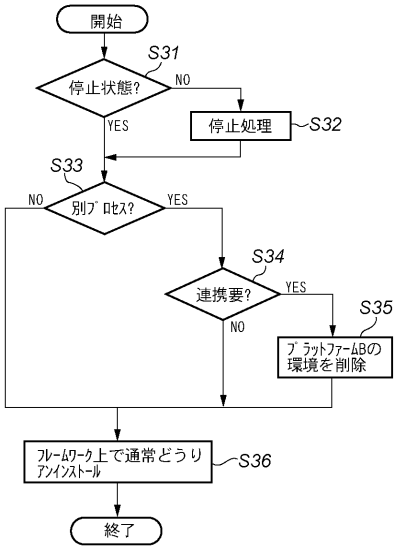
【図７】


7050
7010
7030
プラットファームA
プラットファームB
Java VM A
Java VM B
7020
7040
7070
Java VM C
7060

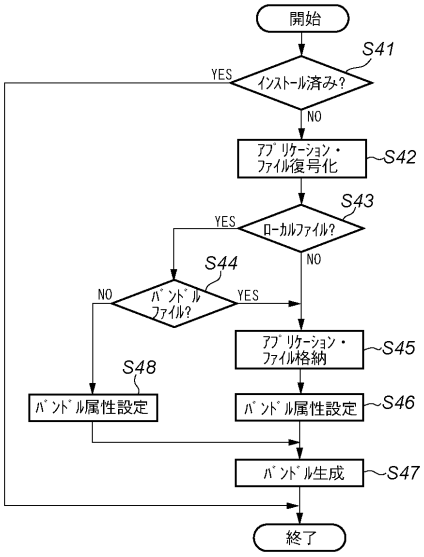
【図８】


開始
S21
停止状態?
YES
NO
S22
別ﾌﾟﾛｾｽ?
NO
YES
S23
連携要?
YES
NO
S24
ﾌﾟﾗｯﾄﾌｧｰﾑBの
ｼｬｯﾄﾀﾞｳﾝ
ﾌﾟﾛｾｽを終了
S25
ﾌﾚｰﾑﾜｰｸ上で通常どうり
ﾊﾞﾝﾄﾞﾙを停止
S26
終了

【図９】


開始
S31
停止状態?
NO
YES
停止処理
S32
S33
別ﾌﾟﾛｾｽ?
NO
YES
S34
連携要?
YES
NO
S35
ﾌﾟﾗｯﾄﾌｧｰﾑBの
環境を削除
ﾌﾚｰﾑﾜｰｸ上で通常どうり
ｱﾝｲﾝｽﾄｰﾙ
S36
終了

【図１０】


開始
S41
YES
ｲﾝｽﾄｰﾙ済み?
NO
ｱﾌﾟﾘｹｰｼｮﾝ・
ﾌｧｲﾙ復号化
S42
S43
YES
ﾛｰｶﾙﾌｧｲﾙ?
NO
S44
NO
ﾊﾞﾝﾄﾞﾙ
ﾌｧｲﾙ?
YES
ｱﾌﾟﾘｹｰｼｮﾝ・
ﾌｧｲﾙ格納
S45
S48
ﾊﾞﾝﾄﾞﾙ属性設定
ﾊﾞﾝﾄﾞﾙ属性設定
S46
ﾊﾞﾝﾄﾞﾙ生成
S47
終了

フロントページの続き

(72)発明者　牛久　豊彦
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

審査官　北元　健太

(56)参考文献　米国特許出願公開第２００３／０１１５１７８（ＵＳ，Ａ１）
特開２００４－３１８４５９（ＪＰ，Ａ）
国際公開第２００２／０７５５３８（WO，Ａ１）
特開平０９－２４５００３（ＪＰ，Ａ）
特開２００１－３２５１０６（ＪＰ，Ａ）
特開２００３－２８０９２６（ＪＰ，Ａ）
国際公開第０１／８４３０３（WO，Ａ１）
田中　克明　Katsuaki　TANAKA，ナビゲーションとの連携動作を可能とする車載端末向けテレマティクスプラットフォームの開発　Telematics Terminal Platform which Enables Cooperation between Telematics Applications and Navigation Functions，情報処理学会研究報告　Ｖｏｌ．２００３　Ｎｏ．８９　IPSJ SIG Technical Reports，日本，社団法人情報処理学会　Information Processing Society of Japan，２００３年　9月　9日，第2003巻，６５－７２頁

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４６－９／５４
Ｇ０６Ｆ　　９／４４５