(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4809772号
(P4809772)
(45)発行日　平成23年11月9日(2011.11.9)　(24)登録日　平成23年8月26日(2011.8.26)

(51)Int.Cl.　F I
G06F 9/44 (2006.01)　G06F 9/06 620K

請求項の数 31 (全 27 頁)

(21)出願番号	特願2006-536552 (P2006-536552)
(86)(22)出願日	平成16年7月12日 (2004.7.12)
(65)公表番号	特表2007-509404 (P2007-509404A)
(43)公表日	平成19年4月12日 (2007.4.12)
(86)国際出願番号	PCT/US2004/022378
(87)国際公開番号	WO2005/045559
(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12日 (2007.7.12)
(31)優先権主張番号	10/692,216
(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)
(31)優先権主張番号	10/692,432
(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)

(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　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　レイモンド　ダブリュ．マッカラム
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

最終頁に続く

(54)【発明の名称】コンピュータシステムおよび分散アプリケーションのモデルに基づく管理

(57)【特許請求の範囲】
【請求項１】
アプリケーションまたはサービスを管理する、モデルに基づく管理システムであって、
前記アプリケーションまたはサービスにおける、機能、構成、システムリソースの利用率、セキュリティ、およびパフォーマンスの少なくとも１つに関する所望の状態を記述した少なくとも１つのモデルを、ソースコードを用いて生成する記述コンポーネントであって、
前記モデルのソースコードに、健全性の判定および／または是正を行うため、及び監視ルールをいつ実行するかを指定するために用いられるソースコード部分を示すためのアトリビューションを挿入するアトリビューションコンポーネントと、
前記モデル及び前記アトリビューションを機械可読形式で含むマニフェストを生成するマニフェストコンポーネントと
を含む、記述コンポーネントと、
前記アプリケーションまたはサービスのインストール時に、前記マニフェストを使用して、前記アプリケーションまたはサービス自体を構成する管理サービスコンポーネントと
を含むことを特徴とするシステム。
【請求項２】
前記管理サービスコンポーネントは、構成の管理、問題の検出、診断、および復旧の少なくとも１つを含む管理動作を介して前記アプリケーションの可用性を確保することを特徴とする請求項１に記載のシステム。

【請求項３】
前記記述コンポーネントは、健全性の状態および復旧、構成設定、および管理タスクの１つまたは複数をモデル化するモデルコンポーネントを含むことを特徴とする請求項１に記載のシステム。
【請求項４】
前記記述コンポーネントは、前記マニフェストから受け取った情報から構成された複数のサービスを含む管理システムコンポーネントを含むことを特徴とする請求項１に記載のシステム。
【請求項５】
前記記述コンポーネントは、前記マニフェスト内で定義された管理タスクを含む管理タスクコンポーネントを含むことを特徴とする請求項１に記載のシステム。
【請求項６】
前記記述コンポーネントは、前記マニフェスト内で表現された前記所望の状態を用いる管理システムコンポーネントを含むことを特徴とする請求項１に記載のシステム。
【請求項７】
前記管理システムコンポーネントは、管理者によって改変された前記所望の状態を用いることを特徴とする請求項６に記載のシステム。
【請求項８】
前記所望の状態は、依存性を検証し、必要なファイル、設定、およびセキュリティデータのみをインストールすることを特徴とする請求項７に記載のシステム。
【請求項９】
前記所望の状態の１つまたは複数は、所定の仕様に従って、イベントをサブスクライブし、前記イベントを転送することを特徴とする請求項７に記載のシステム。
【請求項１０】
前記所望の状態の１つまたは複数は、機器編成データおよびカウンタデータの少なくとも１つを周期的に収集することを特徴とする請求項７に記載のシステム。
【請求項１１】
前記所望の状態の１つまたは複数は、自動管理タスクを実施することを特徴とする請求項７に記載のシステム。
【請求項１２】
前記所望の状態の１つまたは複数は、プログラム機能へのアクセスを制限することを特徴とする請求項７に記載のシステム。
【請求項１３】
前記所望の状態の１つまたは複数は、問題を検出し、根本的原因を診断し、是正措置を講じ、介入が必要なときには管理者に通知することの少なくとも１つを実施することを特徴とする請求項７に記載のシステム。
【請求項１４】
前記所望の状態の１つまたは複数は、複数の異なるコンピュータとともに用いるポリシーをカスタマイズすることを特徴とする請求項７に記載のシステム。
【請求項１５】
ソフトウエアおよびハードウエアのコンポーネントの可用性を監視するためのルールを定義し得るＲＤＬ（ルール定義言語）をさらに含み、前記ＲＤＬは、問題のテスト、診断、解決、検証、および通知の少なくとも１つを進めることを特徴とする請求項１に記載のシステム。
【請求項１６】
抽象リソース、物理リソース、またはリソースコレクションの少なくとも１つを一意に識別するのに使用するＵＲＩ（ユニフォームリソース識別子）をさらに含むことを特徴とする請求項１に記載のシステム。
【請求項１７】
ＵＲＩテンプレートをさらに含み、前記ＵＲＩテンプレートにより、プローブを識別し

、前記プローブを取り出すことなく前記プローブの特徴を理解することができることを特徴とする請求項１に記載のシステム。

【請求項１８】

ＵＲＩテンプレートを用いて、特定のインスタンスを参照せずに機器編成を記述する機器編成カタログをさらに含むことを特徴とする請求項１に記載のシステム。

【請求項１９】

モデルに基づく管理システムであって、

アプリケーション、サービス、および／またはシステムの所望の状態に関して記述したモデルを生成する記述コンポーネントであって、

コンポーネントモデル、健全性モデル、構成モデル、管理タスクモデル、アーキテクチャモデル、パフォーマンスモデル、およびセキュリティモデルの少なくとも１つをさらに含むモデルコンポーネントと、

前記モデルのソースコードに、健全性の判定および／または是正を行うため、及び監視ルールをいつ実行するかを指定するために用いられるソースコード部分を示すためのアトリビューションを挿入するアトリビューションコンポーネントと、

前記モデル及び前記アトリビューションを機械可読形式で含むマニフェストを生成するマニフェストコンポーネントと、

前記アプリケーション、サービス、またはシステムへのインターフェースとなる１つまたは複数のＡＰＩ（アプリケーションプログラムインターフェース）を含む管理システムコンポーネントと、

前記モデルに基づく管理システムによって実施される監視タスク、トラブルシューティングタスク、および管理タスクの少なくとも１つを定義するタスクコンポーネントとの少なくとも１つを含む、記述コンポーネントと、

前記マニフェストを使用して、前記アプリケーション、サービス、および／またはシステムをデプロイする管理サービスコンポーネントと

を含むことを特徴とするシステム。

【請求項２０】

前記ＡＰＩは、中央構成、ロールに基づくアクセス、システムの監視、マニフェストの記憶および編集、イベントの生成およびログ記録、機器編成、パフォーマンスカウンタの処理、ローカルな構成、インストール、自動化、およびタスクのスケジューリングの少なくとも１つを進めることを特徴とする請求項１９に記載のシステム。

【請求項２１】

アプリケーションを管理するためのモデルに基づく管理方法であって、

ソースコードを使用して、前記アプリケーションの所望の状態を記述した１つまたは複数のモデルを形成するステップと、

前記モデルのソースコードに、健全性の判定および／または是正を行うため、及び監視ルールをいつ実行するかを指定するために用いられるソースコード部分を示すためのアトリビューションを挿入するステップと、

前記モデル及び前記アトリビューションを機械可読形式で含むマニフェストを生成するステップと、

前記マニフェストに基づいて、複数の管理システムサービスを構成するステップと、

前記マニフェスト内で所望の状態を表現するステップと

を含むことを特徴とする方法。

【請求項２２】

前記所望の状態の１つまたは複数に基づいて、依存性を検証し、必要なファイル、設定、およびセキュリティの少なくとも１つのみをインストールするステップをさらに含むことを特徴とする請求項２１に記載の方法。

【請求項２３】

前記所望の状態の１つまたは複数に基づく所定のイベント仕様に従って、イベントをサブスクライブし、前記イベントを転送するステップをさらに含むことを特徴とする請求項

２１に記載の方法。
【請求項２４】
前記所望の状態の１つまたは複数に基づいて、機器編成情報、カウンタ情報、およびテストを周期的に収集することによって機器編成をポーリングするステップをさらに含むことを特徴とする請求項２１に記載の方法。
【請求項２５】
前記所望の状態の１つまたは複数に基づいて、自動管理タスクを実施するステップをさらに含むことを特徴とする請求項２１に記載の方法。
【請求項２６】
前記所望の状態の１つまたは複数に基づいて、プログラム機能へのアクセスを制限するステップをさらに含むことを特徴とする請求項２１に記載の方法。
【請求項２７】
問題を検出し、根本的原因を診断し、是正措置を講じ、介入が必要とされるときにはシステム管理者に通知することによって、前記所望の状態に基づいてシステムのプロセスを監視するステップをさらに含むことを特徴とする請求項２１に記載の方法。
【請求項２８】
問題のテスト、診断、解決、検証、および通知を進めるために、ポリシーをカスタマイズし、当該カスタマイズしたポリシーを異なるコンピュータに適用するステップをさらに含むことを特徴とする請求項２１に記載の方法。
【請求項２９】
サービスの１つまたは複数の健全性状態を判定するステップと、
前記アプリケーションの機器編成をパブリッシュするステップと、
当該パブリッシュした機器編成を分析するステップと、
前記パブリッシュした機器編成に基づいて、前記サービスの健全性モデルを形成するステップと
をさらに含み、
前記健全性モデルは、コンポーネント間の関係の情報を含むことを特徴とする請求項２１に記載の方法。
【請求項３０】
アプリケーションまたはサービスを管理する方法を実施するための、コンピュータ実行可能命令を格納したコンピュータ可読記録媒体であって、前記方法は、
ソースコードを使用して、前記アプリケーションの所望の状態を記述した１つまたは複数のモデルを形成するステップと、
前記モデルのソースコードに、健全性の判定および／または是正を行うため、及び監視ルールをいつ実行するかを指定するために用いられるソースコード部分を示すためのアトリビューションを挿入するステップと、
前記モデル及び前記アトリビューションを機械可読形式で含むマニフェストを生成するステップと、
前記マニフェストに基づいて、複数の管理システムサービスを構成するステップと、
前記マニフェスト内で所望の状態を表現するステップと
を含むことを特徴とするコンピュータ可読記録媒体。
【請求項３１】
前記所望の状態の１つまたは複数に基づいて、依存性を検証し、必要なファイル、設定、およびセキュリティの少なくとも１つのみをインストールするステップをさらに含むことを特徴とする請求項３０に記載のコンピュータ可読記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、コンピュータシステムに関し、より詳細には、コンピュータシステムおよびアプリケーションの管理に関する。

【背景技術】
【０００２】
従来方式のシステム管理は、概ね場当たり的である。アプリケーションの開発者は、自身のアプリケーションを管理し、高い信頼性を実現するための体系化されたフレームワークをもっていない。アプリケーションの予想される挙動は、多くはリバース・エンジニアリングにより解析される。アプリケーションの開発者は、自身のアプリケーションを管理し、高い信頼性を実現することについてどのように考えるかに関する体系化された手引きおよびフレームワークをもっていない。さらに、オペレーティングシステムは、開発者が活用し得る包括的なシステムを提供しない。
【０００３】
システムの複雑さは、操作者が理解し得るものを超えつつある。依存性および隠れたエラーを突き止めるために多大な時間が費やされている。機器編成が利用できない場合、操作者は、プロセスダンプ（ｐｒｏｃｅｓｓ　ｄｕｍｐ）を取得する必要がある。というのは、これが、実行中の要求の種類と現在の状態を決定する唯一の方法だからである。
【０００４】
現在のシステムがユーザに潜在的な問題および実際の問題を警告する能力は貧弱である。ユーザは、どんなアプリケーションがインストールされているかを容易に見分けることができず、システムおよびアプリケーションが、正しいファイルを有し、かつ正しいバージョンであり、これらのシステムおよびアプリケーションの使用方法に対して正しく構成されており、環境に対して安全に構成されているか、ならびにこれらのシステムおよびアプリケーションが最適に動作し、かつリソースが不足していないかがわからない。さらに、アプリケーションを複数の機械にわたってデバッグするのは容易ではない。すなわち、共通のアプリケーションおよびトランザクションのコンテキストが存在しない。
【０００５】
操作者は、アプリケーションの依存性が、ファイル、コンポーネント、構成設定、セキュリティ設定、またはストレージエリアネットワーク（ｓｔｏｒａｇｅ　ａｒｅａ　ｎｅｔｗｏｒｋ）およびルータなどの装置のいずれなのかを容易に見分けることもできない。システムは、変更により他のアプリケーションを壊す恐れがあることをユーザに警告することもできないし、この情報を用いて根本的な原因を特定する助けとすることもできない。
【発明の開示】
【発明が解決しようとする課題】
【０００６】
現在、最も一般的なのは事後対応型の監視であり、この場合、警告により、ユーザは障害があったことがわかるが、問題の原因はわからない。改善されたスクリプトおよびプロバイダ（ｐｒｏｖｉｄｅｒ）は、より有益かつ動作に結びつけ得る警告を提供し得るが、根本的な原因を分析する基盤が欠けている。トラブルシューティングを行うには、しばしば追加の診断が必要とされる。しかし、事後対応型の監視に伴う問題は、多くの場合警告は遅すぎ、すでにアプリケーションがユーザに利用可能な状態ではなくなっていることである。フェイルオーバをトリガするか、あるいは、負荷分散装置によりサーバをオフラインにすることによって監視は役に立ち得る。しかし、システムは、アプリケーションにおける潜在的な問題を、これらの潜在的な問題が障害になる前に検出するのに十分にインテリジェントであるべきである。
【０００７】
他の問題は、複数のマシンおよびクライアント全体を見渡すことによってしか検出されない。これらの例には、分散型浸入検出およびアプリケーションパフォーマンスの低下が含まれる。管理者が、その自由裁量で想定パフォーマンスからの逸脱を判断でき、スナップショット（ｓｎａｐｓｈｏｔ）をキャプチャしたときに根本的な原因が構成の変更であることを突き止めることができ、ユーザの不満が生じる前に問題を解決すれば、多くの大規模なネットワークパフォーマンスの問題およびダウンタイムを回避し得るはずである。

【０００８】
従来方式では、分散アプリケーションに伴う問題は、ユーザの観点から履歴のデータまたは傾向を見ることによってしか判定されない。管理者は、管理者による複製バックログが問題であるかどうかがわからないことが多く、まずサービスを実行し、運用メトリクスのログを取って警告閾値および臨界閾値によるベースラインを確立する必要がある。
【０００９】
管理基盤を実現するために、改善されたメカニズムが求められている。
【課題を解決するための手段】
【００１０】
本発明のいくつかの態様を基本的に理解するために、本発明の概要を簡略化して以下に示す。この概要は、本発明を広範に概説したものではなく、本発明の主要不可欠な要素を明らかにし、また、本発明の範囲を限定するためのものではない。この概要の唯一の目的は、本発明のいくつかの概念を簡略化した形態で、後で示すより詳細な説明の前置きとして示すことである。
【００１１】
本明細書で開示し特許請求する発明は、その一態様において、開発者が、アプリケーションまたはサービスを、そのコンポーネントに関して記述し得る革新的なフレームワークを提供する、モデルに基づく管理システムを含む。開発者は、このアプリケーションまたはサービスの所望の状態を、機能、構成、セキュリティ、およびパフォーマンスに関して記述し得る。この記述またはモデルは、アプリケーションとともに提供され、インストール時にシステムがこれを用いて管理サービスを構成する。コンピュータシステムは、インストール時に開発者の記述を用いて管理サービスを構成する。この管理サービスは、構成の管理、問題の検出、診断、および復旧などの自動管理動作を介してアプリケーションの可用性を確保する助けになる。このモデルは、管理者が実施し得る一般のタスクも記述する。
【００１２】
このモデルに基づく管理アーキテクチャは、健全性の状態および復旧、構成設定、および管理タスクなど、アプリケーションを構成するコンポーネント用のモデルと、監視を行うために機器編成および論理を示す、ソースコード内のアトリビューションと、アプリケーションとともに出荷される１つまたは複数のマニフェストとを含む。これら１つ（または複数）のマニフェストは、モデルおよびソースコードアトリビューションからの情報を、管理システムサービスが使用し得る機械可読の形式で含む。このモデルに基づく管理アーキテクチャはさらに、アプリケーションマニフェスト内の情報によって構成された複数のサービスからなる管理システムと、マニフェスト内で定義されるアプリケーション用管理タスクとを含む。
【００１３】
このモデルに基づく管理アーキテクチャのシステムコンポーネントは、アプリケーションの可用性を確保するのに必要なサービスからなる。このシステムは、マニフェスト内で表現され、かつ管理者によって改変された所望の状態を用いて、依存性を検証し、必要なファイル、設定、およびセキュリティのみをインストールするインストール作業と、指定どおりにイベントをサブスクライブ（ｓｕｂｓｃｒｉｂｅ）し転送するイベントサブスクリプションと、機器編成をポーリングして機器編成およびカウンタを周期的に収集することと、タスクをスケジュールして自動管理タスクを実施することと、プログラム機能へのアクセスを制限する、ロールに基づくアクセスと、問題を検出し、根本的原因を診断し、是正処置を講じ、介入が必要なときにはシステム管理者に通知する監視機能と、上記を行うためのポリシーをカスタマイズし、それを多くの機械に適用するための中央構成とを実施する。
【００１４】
本発明の別の態様では、モデルに基づく管理システムは、ハードウエアおよびソフトウエアの分散ネットワークに適用される。それに従って、ローカルおよびリモートのアプリ

ケーションのコンポーネントが、ローカルおよびリモートのマシンおよびサービスとともに記述され管理される。

【００１５】

上記および関連する目的を達成するために、本明細書では、以下の説明および添付の図面に関連して本発明のある種の態様の例を説明する。ただし、これらの態様は、本発明の原理を利用し得る様々な方法を示すほんの数例であり、本発明は、このような態様およびそれらの均等物をすべて包含することを意図している。本発明の他の利点および新規な特徴は、本発明の以下の詳細な説明をこれらの図面と併せ読めば明らかになるであろう。

【発明を実施するための最良の形態】

【００１６】

次に、図面を参照して本発明を説明する。これらの図面を通じて、同じ参照数字を用いて同じ要素を指す。以下の記載では、説明のために、本発明が十分に理解されるように多くの特定の細部を述べる。ただし、これらの特定の細部を用いずに本発明を実施し得ることが当業者には明らかであろう。他の例では、本発明の説明をうまく進めるために、周知の構造および装置はブロック図の形式で示す。

【００１７】

本出願では、「コンポーネント」および「システム」という用語は、コンピュータに関係するエンティティを指すためのものである。これらは、ハードウエア、ハードウエアとソフトウエアの組合せ、ソフトウエア、または実行中のソフトウエアのいずれかである。例えば、コンポーネントは、プロセッサ上で実行中のプロセス、プロセッサ、オブジェクト、実行ファイル、実行の流れ、プログラム、および／またはコンピュータとし得るが、これらに限定されるものではない。例として、サーバ上で実行中のアプリケーション、およびこのサーバはともにコンポーネントとし得る。１つまたは複数のコンポーネントが、実行の過程および／または流れの中に存在することもあるし、あるコンポーネントが、１つのコンピュータ上に局在し、かつ／または２つ以上のコンピュータに分散されることもある。

【００１８】

本明細書では、「推測」という用語は概ね、イベントおよび／またはデータを介してキャプチャされた１組の観察結果から、システム、環境、および／またはユーザの状態について推論または推測するプロセスを指す。推測を用いて、特定の状況または動作を識別し、また、例えば状態全体にわたる確率分布を生成し得る。この推測は、確率的であり得る。すなわち、データおよびイベントの考察に基づいて、対象とする状態全体にわたる確率分布を計算することである。推測は、１組のイベントおよび／またはデータから高水準のイベントを構成するのに用いる技術を指すこともある。このような推測により、１組の観察されたイベントおよび／または記憶されたイベントデータから、これらのイベントが時間的に近接して相関しているかどうかにかかわらず、また、これらのイベントおよびデータが１つまたは複数のイベントおよびデータのソースから得られたかどうかにかかわらず、新しいイベントまたは動作が構築される。

【００１９】

次に図１を参照すると、本発明に従ってアプリケーションまたはサービスのモデルに基づく管理をうまく進めるアーキテクチャ１００が示されている。このモデルに基づく管理手法により、開発者は、アプリケーションまたはサービス１０２を、その構成要素、ならびに機能、構成、セキュリティ、およびパフォーマンスの点での所望の状態に関して記述することができる。そのため、アプリケーションまたはサービスの記述５０４により、少なくとも、モデルコンポーネント１０６、マニフェスト（ｍａｎｉｆｅｓｔ）コンポーネント１０８、システムコンポーネント１１０、およびタスクコンポーネント１１２を含めて、１つまたは複数の管理可能なコンポーネントに関してアプリケーションまたはサービス１０２を記述することが容易になる。モデルに基づく管理システム１００は、アトリビューション（ａｔｔｒｉｂｕｔｉｏｎ）コンポーネント１１４を使用して、モデルコンポーネント１０６からマニフェストコンポーネント１０８へのソースコードのアトリビュー

ションをうまく進める。
【００２０】
コンピュータシステム５１６は、アプリケーション１０２のインストール時にアプリケーションまたはサービスの記述１０４を使用して、このコンピュータのオペレーティングシステムに関連する管理サービス５１８を構成する。次いで、管理サービス１１８は、構成管理、問題検出、診断、および復旧などの自動管理動作によってアプリケーションまたはサービス１０２の可用性を確保する助けとなる。モデル１０６は、管理者が実施し得る共通のタスクも記述する。モデルに基づく管理アーキテクチャ１００は、総所有コストを下げることを容易にし、開発から、デプロイメント（ｄｅｐｌｏｙｍｅｎｔ）、運用、およびビジネス分析までのアプリケーションのライフサイクル全体にわたって用いられる。一般に、開発者は、アプリケーションまたはサービスの１つまたは複数のモデルを生成することから開始する。このモデルは、このアプリケーションがどのように機能するか、その構成要素、開発者が定義し監視することを選択する所望の健全性の状態、少なくともこのアプリケーションまたはサービスをどのようにインストールし、このアプリケーションまたはサービスがどんな設定を必要とするかに関する構成上の態様、ならびに管理タスクおよびそのスケジューリングという点で生成されるものである。次いで、このモデルのソースコードの、明示する特定の領域に属性を与える（あるいは、タグを付ける）。
【００２１】
これらのモデルを機器編成マニフェストにロールアップ（ｒｏｌｌ　ｕｐ）する。これらのモデルは、テキストドキュメント、スプレッドシートドキュメントなどの形式をとる傾向がある。これらは、コード、スクリプト、ツールによって、あるいは手作業で、より多くのＸＭＬスキーマになる傾向があるマニフェストに変換され、さらに機械によって処理され機械によって読み込まれる構造化ドキュメントである。すなわち、モデルドキュメントは、より人間可読であり、マニフェストはより機械可読である。次いで、これらのマニフェストを使用して、システム管理をうまく進める。
【００２２】
アトリビューションサブコンポーネント１１４は、ソースコードアトリビューションに関連するものである。アトリビューションを用いて、管理情報およびこの管理情報が関係するコードを表す。アトリビューションがないと、２つの別々のコードを記述することが必要になる。１つは、通常のアプリケーション処理用のものであり、１つは、アプリケーションを管理にエクスポーズ（ｅｘｐｏｓｅ）するためのものである。ソースコード中のアトリビューションを用いて、（プローブと呼ぶ）コードのどの部分を用いて健全性を判定かつ／または是正し、かつ管理ルールを実行する時点を指定すべきなのかを記述する。プローブは、既存のオペレーティングシステムＡＰＩ（アプリケーションプログラムインターフェース）にアクセスするコンポーネントから、あるいは、実行中のアプリケーションまたはサービスの内部にロードしたコンポーネントからエクスポーズし得る。いずれの場合でも、開発者は、アトリビューションを付加して、これらのコンポーネント内のタイプのどのサブセットをエクスポーズすべきかと、それらをどのように識別すべきかを示す。プローブは、管理者名前空間内のＵＲＩ（ユニフォームリソース識別子）を用いて識別する。実行時に、プローブは、コンピュータ上のすべてのプローブのカタログ内から識別し、このプローブについての関連情報をたどることによって取り出す。
【００２３】
ソースコードアトリビューションは、監視サービス、例えば属性関数に命令も提供し得る。この属性関数は、監視ルールとして使用され、起動時にロードされ、周期的にポーリングされ、イベントに対して実行されるべきものである。このアトリビューションを自動的に処理し、機器編成と同じやり方でマニフェスト中に置くことができる。そのため、アトリビューションは、単に機器編成ではなく、他の管理目的にも用いられる。アトリビューションを用いて、管理タスクおよび／または是正措置（ｃｏｒｒｅｃｔｉｖｅ　ａｃｔｉｏｎ）をサポートすることもできる。
【００２４】

次に図２を参照すると、モデルに基づく管理アーキテクチャ１００の主要コンポーネントを記述することに関係する図面マップ２００が示されている。このアーキテクチャは、図３Ａに関連して説明するモデルコンポーネント１０６、図３Ｂに関連して説明するマニフェストコンポーネント１０８、図３Ｃおよび図３Ｄに関連して説明するシステムコンポーネント１１０、ならびに図３Ｅに関連して説明するタスクコンポーネント１１２を含む。アトリビューションはすでに説明してあり、本明細書を通じて参照する。

【００２５】

次に図３Ａを参照すると、モデルに基づく管理アーキテクチャのモデルコンポーネント１０６に関係するブロックが示されている。アプリケーションを構成するコンポーネント、健全性の状態および復旧、構成設定、ならびに管理タスクについてモデルが形成される。

【００２６】

これらに加えて、システムの任意のかつすべてのコンポーネント（ならびにこれらに関連する関係、依存性、およびサービスロール）をモデル化するためのコンポーネントモデルサブコンポーネント３００がある。コンポーネントモデル３００は、ファイル、構成、アプリケーションをインストールし得る様々な方法などを記述する。

【００２７】

健全性モデルサブコンポーネント３０１は、様々な障害状態、およびアプリケーションまたはサービスの障害の発生のしかたが記述されるように形成し得る。健全性モデル３０１は、健全性についての機能を自動化するためにとる必要があるステップを記述する。健全性モデル３０１は、少なくとも、障害の状態、これらの状態の検出、システム状態の検証、診断、および解決を表す。健全性の状態は、完全に健全である、完全に障害がある、および任意の中間状態とみなすにはどんな基準を満たさなければならないかという点で記述し得る。この中間状態は、例えば、パフォーマンスの劣化、部分的に機能している、顧客機能の一部が機能している、顧客機能の一部が、想定したレベルのサービスを提供するアプリケーションまたはサービスである、などである。健全性は、機能は満足のいく状態だが、パフォーマンスが基準以下であり、アプリケーションまたはサービスが健全でないことを示すことも考慮する。

【００２８】

構成モデルサブコンポーネント３０２は、システム構成のモデル化に関係するものである。構成モデル３０２を用いて、アプリケーション設定、ユーザコントロール、既定値、様々な制約などを記述する。管理タスクモデルサブコンポーネント３０３は、管理タスクのモデル化に関係するものであり、開始、停止、ユーザの追加、データベースの追加、および健全性モデル３０１からコールし得る是正処置など、ユーザがシステムに対して取り得る動作を含む。モデル３０２は、アプリケーションまたはサービスによって行い得るすべてのことを列挙する。アーキテクチャモデル３０４を用いて、分散環境および関連するデプロイメントを記述する。これらは通常、例えば、同じまたは類似のハードウェアおよびソフトウェアの設定および構成を有するクライアント、ならびに分散データベースからなる大規模ネットワークに関係するものである。そのため、ローカルアプリケーションは、リモートディスクアレイに依存し得る。デプロイメントの際に、このディスクアレイを、マニフェストによって、かつＵＲＩを用いてデプロイメントレベルでインスタンス化する必要がある。ＵＲＩは機械に依存しないので、分散システムも、モデルに基づく管理システムの利益を得ることができる。パフォーマンスモデル３０５は、開発者が、メトリクス（ｍｅｔｒｉｃｓ）を用いてアプリケーションまたはサービスのパフォーマンスを監視したい場合のやり方が記述されるように形成し得る。これは、システムの健全性と密接に関係する。アプリケーションまたはサービスに関係するセキュリティのタイプを記述するセキュリティモデル３０６を生成し得る。

【００２９】

本明細書で提供するモデルの数は網羅的ではないことに留意されたい。というのは、開発者は、アプリケーションまたはサービスの様々な態様を管理するための多くの異なるモ

デルを提供し得るからである。

【００３０】

この主題のモデルに基づくシステムは、その様々な態様を実施するために様々な人工知能に基づく方式を採用し得る。例えば、モデルに関して、所与のインスタンスまたはインプリメンテーションにはどのモデルを使用し得るかを決定するプロセスを、自動分類システムおよびプロセスによってうまく進めることができる。さらに、このような分類子を用いて、システムのパターンを検出し、何が良好な状態で、何が不良状態であり、何が成功したトランザクションで、何が成功しなかったトランザクションかを学習することを開始するシステムの動作プロファイルを形成し得る。次いで、この情報を、対応するモデルにフィードバックし、後続のシステムに対する更新されたモデルとして使用することができる。このような分類では、（例えば、解析ユーティリティおよびコストを計算に入れる）確率および／または統計に基づく解析を用いて、ユーザが自動的に実施されるように望む動作を予測または推測することができる。例えば、ＳＶＭ（サポートベクトルマシン）の分類子を用いることができる。他の分類手法には、ベイズネットワーク（Ｂａｙｅｓｉａｎ　ｎｅｔｗｏｒｋｓ）、デシジョンツリー（ｄｅｃｉｓｉｏｎ　ｔｒｅｅ）が含まれ、異なる独立パターンを提供する確率分類モデルを用いることができる。本明細書で用いる分類には、優先モデルを形成するのに用いられる統計回帰も含まれる。

【００３１】

本明細書から容易に理解されるように、モデルに基づくシステムは、（例えば、一般的な訓練データによって）明示的に訓練された分類子、ならびに（例えば、ユーザの挙動を観察し、外からの情報を受け取ることによって）暗示的に訓練された分類子を採用し、それによって、これら１つ（または複数）の分類子を用いて、所定の基準に従って、例えば、所与のインプリメンテーションに対してどの初期設定を用いるかを自動的に決定し、次いで、システムが成熟し、データ、インストールされたいくつかのアプリケーション、および対話するノードの数に関する様々なロード条件を経験するにつれ、時間をかけて設定を調整することができる。例えば、十分に理解されているＳＶＭに関して、ＳＶＭは、分類子コンストラクタおよび特徴選択モジュール内での学習または訓練の段階を介して構成される。分類子は、入力属性ベクトルｘ＝（ｘ１、ｘ２、ｘ３、ｘ４、ｘｎ）を、コンフィデンス（ｃｏｎｆｉｄｅｎｃｅ）にマッピングする関数である。コンフィデンスの入力はクラスに属し、すなわち、ｆ（ｘ）＝ｃｏｎｆｉｄｅｎｃｅ（ｃｌａｓｓ）である。管理システムの場合、例えば、属性は、所望の状態のシステムパラメータであり、クラスは、対象とするカテゴリまたは領域（例えば、すべてのドライブ、すべてのネイティブプロセス）である。分類子を用いて、トランザクションログをキャプチャ（ｃａｐｔｕｒｅ）し解析し、パターンを探し、成功したパターンおよび成功しなかったパターンを探すことによってシステムを診断することもできる。

【００３２】

構成の健全性は、例えば、キューのサイズを５から１０に変更し、この変更がアプリケーション、サービス、またはシステムにどんな影響を及ぼし得るかを判定することを含む。同じことが、セキュリティおよびパフォーマンスに当てはまる。この場合、分類子を用いて、パフォーマンスカウンタを監視し、それに従ってシステムの変更を行ってパフォーマンスを最適化することができる。セキュリティをパターンについて監視し分析することもでき、この影響を用いてセキュリティポリシーを提案または変更し得る。このように、健全性は、システムの多くの領域に適用し得る広い概念であることを理解されたい。システム規模の範囲では、パフォーマンスは良好だが、セキュリティが貧弱なことがある。そのため、システムの多くの規範にまたがる包括的な観点が有利である。

【００３３】

管理者が所望する状態をコード中に表現することができ、これを、マニフェストで表面化させ、それを渡して監視サービスによって監視する。システムは、マニフェスト中の命令に基づいてアプリケーションまたはサービスを監視し、アプリケーションまたはサービスがパフォーマンスを満たさないときには管理者に警告し、命令に基づいて是正措置を講

ずることができる。例えば、電子メール用のテスト設定が維持されず、ある期間閾値未満に落ちる場合、負荷が正常な状態に戻るまで別の機械を追加することができ、ネットワークトラフィックを、リソースの数を増やすトリガとして用いて、所与の負荷に対処することもできる。目標は、手作業の動作が必要とされるときにのみ管理者が関与するように、できるだけ多くを自動化することである。

【００３４】

モデルに基づく管理システムは構成可能である。この管理システムは、コンポーネントに基づくものであり、コンポーネントはほぼ何でも含む。そのため、このシステムを、その最も小さい管理可能な部品まで小さくし、元通りに構成し得る。データベースでは、例えば、インスタンス、このデータベース、テーブル、およびストアドプロシージャを有するアプリケーションがあり、１つのファイルにまで小さくし得る。４０１ｋの応用例を考える。４０１ｋの応用例は、データベース、ウエブサーバ、および顧客自身のビジネス論理、さらに、オペレーティングシステムおよび関連のものに依存するデータベースにまで依存し得る。様々なレベルで管理および報告を行うことが望ましい。アプリケーションは、コンポーネント間の関係によって記述される。これらの関係は、個々のアプリケーションをどのように組み立てるか（例えば、ＳＱＬサーバは、サービス、インスタンス、およびデータベースを含む）、プラットフォームの要件（例えば、オペレーティングシステムその他のアプリケーション）、および他のコンポーネント（ＳＱＬサーバに接続するウエブサーバ）への通信を表現し得る。１人の管理者は、データベースおよび１台の機械を管理し、金融管理者は４０１ｋの応用例を管理し、ＣＩＯはこれらのアプリケーションおよび機械のすべてを管理する。これらのモデル、報告、および所望の状態により、個々のメトリクスを参照してシステムが想定どおりのことを行っているかどうかを判定し得るように、すべてのことが処理されるべきである。

【００３５】

すべてのモデルはＵＲＩ名前空間に結びつけられ、それによって、システムを運用し、インストールされているすべてのコンポーネントを列挙し、コンポーネントに問合せを行う標準的な方法が提供される。この問合せには、コンポーネントが何を提供するか、何が健全とみなされるか、コンポーネントがどんなイベントを有するか、この１日またはここ数時間でどんなエラーイベントが生じたか、どんな構成設定が含まれているか、この１時間でどんな変化が生じたか、などが含まれる。

【００３６】

次に図３Ｂを参照すると、モデルに基づく管理アーキテクチャのマニフェストコンポーネント１０８に関係するブロックが示されている。アプリケーションとともに出荷されるマニフェストは、モデルおよびソースコードアトリビューションからの情報を、管理システムのサービスが使用するために機械可読形式で含む。アプリケーション用の管理タスクは、このマニフェスト中で定義される。コンポーネント依存性、コンポーネント間の関係、およびサービスロールに関係する第１マニフェストサブコンポーネント３０７、イベント、プローブ、ルール、および動作に関係する第２マニフェストサブコンポーネント３０８、設定およびアサーション（ａｓｓｅｒｔｉｏｎ）に関係する第３マニフェストサブコンポーネント３０９、コマンド（すなわち、コマンドレット）および管理ロールに関係する第４マニフェストサブコンポーネント３１０、分散環境に関係する第５マニフェストサブコンポーネント３１１、ならびにデプロイメントに関係する第６マニフェストサブコンポーネント３１２を含めて、モデルに対応して生成されたいくつかのマニフェストが存在し得る。

【００３７】

マニフェストは、開発者、運用チーム、および管理者の間の「橋渡し」であり、属性を与えられたコードについてモデルを走査するツールによって自動的に生成される。設定エンジンは、コンポーネントマニフェスト３０７を使用して、アプリケーションまたはサービスのインストール方法を決める。コンポーネントマニフェスト３０７は、論理コンポーネント、ファイル、これらのファイルをどこにインストールすべきか、および構成設定（

または任意の設定）を記述する。依存性は、インストール前に定義する必要があるものであり、アプリケーションを異なるモードで、様々なセキュリティの度合いおよび異なる動作プロファイルでインストールすることができるように様々なロールを含む。コンポーネントマニフェスト３０７は、手動および自動で何を行うかをユーザおよび／またはシステムにより簡単に知らせる。マニフェストの粒度は、１コンポーネント当たり１つのマニフェストにまで小さくし得る。

【００３８】

従来方式では、実際に必要とされるよりもはるかに多くのファイルをインストールする。マニフェストにより、必要とされるファイルだけをインストールすることができる。こうすると、少なくともパフォーマンスおよびセキュリティが改善される。ソフトウェアの依存性は、マニフェスト３０７内で定義される。アプリケーションレベルでは、こられの依存性は、単一の機械に固有のものであり、コンポーネントの関係およびハードウェアリソースを定義し得る。コンピュータは、マニフェストによって記述し得る。例えば、アプリケーションは、特定の製造業者のデュアルプロセッサマシン上にデプロイ（ｄｅｐｌｏｙ）すべきである、あるいは、４プロセッサマシンに接続すべきである、などである。マニフェスト３０７は、プロセッサ、メモリ、ドライブなどを、実装に必要とされるハードウェアの粒度のレベルまで記述する。そのため、管理は、従来方式のシステムの場合の事後対応型ではなく、事前対応型になり得る。ハードディスクの障害は、システム温度を長時間にわたって監視し、電源レール電圧を監視し、それらが十分であると判明した場合には、例えば、熱による障害によって生じると判定し得る。

【００３９】

健全性モデル３０１を用いて、健全性マニフェスト３０８を生成する。健全性マニフェスト３０８は、アトリビューションその他のツールを用いて健全性モデル３０１からポピュレートする。イベントはモデル３０１内でコールされず、リソースファイル内でコールされる。ツールがリソースファイルおよび属性を与えられたソースコードを走査し、健全性マニフェスト３０８をポピュレートする。所定のシーケンスのイベントに注意するか、あるいはパフォーマンスカウンタの閾値を監視することによって障害状態を検出し得る。このような障害状態の対処方法に関して、システムに命令を提供し得る。健全性モデルは、ルールに変換される。健全性マニフェスト３０８は、ｅｖｅｎｔ１、ｅｖｅｎｔ２、ｔｉｍｅ３などのパラメータを伴うルールタイプのイベントシーケンスを含む。

【００４０】

構成モデル３０２は、どんな設定が含まれるかを記述し、設定／アサーションマニフェスト３０９に変換される。マニフェスト３０９は、コンポーネントがインストールされるときにシステムが設定を生成するための命令スキーマを提供する。

【００４１】

管理タスクモデル３０３は、コマンドレット／管理ロールマニフェスト３１０を介して動作に変換される。例えば、データのバックアップが必要とされる場合、コマンドレットは、このバックアップタスクをうまく進めるのに用いられる実際のコードまたはＵＲＩになる。多くの管理タスクを実施する必要がある場合、マニフェスト３１０は、これらのコマンドへの、おそらくはコードへのＵＲＩ経路を提供する。コマンドレットは、このコードに対するアサーションによって処理するか、あるいは外部コードを要求し得る。管理ロールは、例えば、アプリケーションまたはサービスを管理するユーザの複数のクラスおよびそれらがそれぞれ実行し得る制御レベルをサポートする別のアブストラクション（ａｂｓｔｒａｃｔｉｏｎ）である。これは、ロールに基づくアクセスに関係する。様々なユーザのロールおよびそれらの許可機能を記述するメタデータが必要とされる。ロールは、誰にインストール権限があるのか、誰が監視を変更し得るのか、誰が健全性を監視し得るのか、誰が警告を解決し得るのか、誰がこれらの様々な動作を取り得るのかなど、このシステムのすべての態様にまたがるものである。

【００４２】

タスクモデル３０３は、管理者がすべきことと開発者が考えることを定義する。これら

は、マニフェスト３１０に表され、運用チームによってその環境に対してカスタマイズされる。これらのカスタマイズ化は、クラスレベルおよびインスタンスレベルで行い得る。クラスレベル、インスタンスレベルでマニフェストに変更を加えることができ、実行時に直接変更を加えることができる。ここで開示するモデルに基づく管理アーキテクチャの極めて強力な特徴は、機能をまずクラスレベルで記述することができ、実行時にアクセスがインスタンス空間に対して行われることである。

【００４３】

アーキテクチャモデル３０４は、分散コンポーネントマニフェスト３１１およびデプロイメントマニフェスト３１２を表面化（ｓｕｒｆａｃｅ）させる。例えば、機械間のネットワーク接続、ハードウェア要件はここで扱う。デプロイメントマニフェスト３１２は少なくとも、ウエブサーバ、中間階層サーバ、およびデータベースサーバを備えるアプリケーションをサポートし、フロントエンド／バックエンドアプリケーション、２つのアプリケーション間のネットワーク接続性を含み、個々のノード間の関係を記述する。デプロイメント期間には、全体的アーキテクチャモデル３０４で記述したもののインスタンスが生成される。

【００４４】

パフォーマンスモデルおよびセキュリティモデル（３０５および３０６）はそれぞれ、対応する関連機能および操作を記述する（図示しない）マニフェストをサポートする。

【００４５】

機械ベースの学習を採用することに戻ると、分類子を用いて、例えば第１のデプロイメント中に、要件に基づいてモデルコードの選択部分のマニフェストを選択し、動的に生成し得る。使用するアトリビューションの数を増減させて、既定モデルを自動的に生成し得る。時間の経過とともにシステムの動作情報が利用可能になるので、この情報を分析して、例えば、最新のデータの傾向およびログに基づいて、特定の領域におけるシステムをより厳密に監視することができるようにマニフェストの粒度レベルを調整し得る。次いで、更新されたマニフェストを生成し直し、必要に応じて、アプリケーションまたはサービスをより厳密に監視するために使用する。

【００４６】

マニフェストに、製造業者からの既定のインストールまたは推奨された最適な実務慣行が記述されている場合、管理者は、それらの事柄を変更したいことがある。例えば、健全性ルールに関して、管理者は、閾値を３０から４０に変更し、コンポーネントをインストールし、またセキュリティポリシーを上書きしたいことがある。これは、マニフェストのカスタマイズされたバージョンを生成して、製造業者がバンドルしたマニフェストを上書きすることによって行い得る。異なるバージョンはインストール中に検出することができ、それによって、既定のマニフェストまたはカスタマイズされたマニフェストの選択権がユーザに与えられる。あるいは、上書き値が列挙された、システムが読み込む別のファイルが存在し得る。この一覧を表示し、それによってユーザが選択して既定のマニフェストに適用するか、あるいは、インストール中に既定の設定が上書きされるようにする。

【００４７】

分散アプリケーションに関して、管理者は、より一般には、これらのうち３つを、そのうちの４つを、それらのうち６つをすべてこの構成内で結線（ｗｉｒｅｄ）したいことを規定し得る。この管理者は、それに従って所与の環境についてのデプロイメントマニフェスト３１２をカスタマイズし得る。

【００４８】

次に図３Ｃを参照すると、モデルに基づく管理アーキテクチャに従ってアプリケーションまたはサービス３１４を管理するのに使用するシステムコンポーネント１１０のコアシステムＡＰＩのブロック図が示されている。システムコンポーネント１１０は、管理すべきアプリケーションまたはサービス３１４を含む。システム１１０は、モデルに基づく管理をうまく進めるために協調して通信を行ういくつかのＡＰＩを含む。システム１１０は、（図３Ｂに関して説明した）アプリケーションマニフェスト内の情報によって構成され

た複数のサービスからなる。

【００４９】

システム１１０は、アプリケーションの可用性を確保するのに必要なサービスからなり、マニフェストコンポーネント１０８内で表現され、かつ管理者によって改変された所望の状態を用いて、依存性を検証し、必要なファイル、設定、およびセキュリティだけをインストールするインストール作業と、イベントをサブスクライブ（ｓｕｂｓｃｒｉｂｅ）し、指定どおりに転送するイベントサブスクリプションと、機器編成をポーリングして、機器編成およびカウンタを周期的に収集することと、および統合的トランザクションまたはユーザシミュレーショントランザクションとを実施する。アプリケーションが利用可能であり、かつ想定どおりに（所望の状態で）実施されているかどうかを判定する最適な方法の１つは、監視システムが、ユーザであるかのようにこのアプリケーションを使用することである。これは能動的監視である。潜在的に可能な第２の方法は、ユーザによる現実のトランザクションを能動的に監視し、集計したデータをシステムに報告して分析することである。これらのステップではループが閉じており、内部アプリケーションデータが十分ではないことが示される。モデルに基づく管理は、アプリケーション外でも機能する。

【００５０】

また、システム１１０は、マニフェストコンポーネント１０８内で表現された所望の状態を用いて、自動タスク管理を行うためのタスクスケジューリングと、プログラムの機能へのアクセスを制限するためのロールに基づくアクセスと、問題を検出し、根本的原因を診断し、是正処置を講じ、介入が必要な場合にはシステム管理者に通知するために監視を行うことと、上記を行うためにポリシーをカスタマイズし、それを多くの機械に適用するための中央構成（ｃｅｎｔｒａｌ　ｃｏｎｆｉｇｕｒａｔｉｏｎ）とを実施する。

【００５１】

アプリケーション３１４と通信するインストールＡＰＩ　３１６が提供され、それによって、このアプリケーションのインストール、アプリケーションのアップデート、およびパッチがうまく進む。インストールＡＰＩ　３１６は、コードを介してマニフェストアセンブリを取得し、これらのアセンブリをインスタンス化する。これは、この機械上に、このコンポーネント、このマニフェスト、およびこのバージョンをインストールするようにシステムに命令することによって行われる。インストールＡＰＩ　３１６は、それに関連するプロトコル３１８およびビューア３２０を有する。プロトコル３１８は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３２０は、インストールＡＰＩ　３１６に関係するデータを表示する。インストールＡＰＩ　３１６は、単一の機械上へのインストールだけでなく、ローカルシステムおよびリモートシステムがともに関与する分散アプリケーションまたはサービス、ならびにハードウェアのプロビジョニング（ｐｒｏｖｉｓｉｏｎｉｎｇ）およびアブストラクションもうまく進める。分散データセンタ環境では、ハードウェアシステムを全体的により細かい粒度に、個々の機械のアブストラクションに抽象化し得ることが重要である。プロトコルは、ＡＰＩに関して本明細書で企図されているように、ＡＰＩ関連データの送受信を司るルールである。ビューア３２０は、この記述の中で理解されるように、ＡＰＩ、ここではインストールＡＰＩ　３１６に関連するデータを表示するプログラムである。ＡＰＩデータは、例えば、サウンドファイル、ビデオファイル、その他のタイプのデータファイルを含むが、これらに限定されるものではない。

【００５２】

システム１１０は、アプリケーション３１４と通信する構成ＡＰＩ　３２２を含み、それによって、アプリケーション３１４の構成がうまく進む。構成ＡＰＩ　３２２は、それに関連するスキーマ３２３、プロトコル３２４、およびビューア３２６を有する。スキーマ３２３は、ＡＰＩ　３２２とアプリケーション３１４の間で渡されるデータの構造および内容を定義する。プロトコル３２４は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３２６は、構成ＡＰＩ　３２２に関係するデータを表示する。

【００５３】
分散環境で多数対１の管理をうめく進める管理ＡＰＩ　３２８も含まれる。ＡＰＩ　３２８は、管理されるアプリケーション３１４と通信し、（図示しない）リモートシステムとも通信する。ＡＰＩ　３２８は、それに関連するプロトコル３３０およびビューア３３２を有する。
【００５４】
システム１１０は、アプリケーション３１４と通信するパフォーマンスカウンタＡＰＩ　３３４を含み、それによって、アプリケーション３１４を管理するのに使用するカウンタ変数の追跡がうまく進む。カウンタＡＰＩ　３３４は、それに関連するプロトコル３３６およびビューア３３８を有する。プロトコル３３６は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３３８は、カウンタＡＰＩ　３３４に関係するデータを表示する。パフォーマンスカウンタは、アプリケーション３１４によってエクスポーズされ、ビューア３３８を介してカウンタをパブリッシュ（ｐｕｂｌｉｓｈ）する。
【００５５】
アプリケーション３１４と通信する機器編成ＡＰＩ　３４０が提供され、それによって、機器編成を構成し、機器編成データをアプリケーション３１４に渡すことがうまく進む。機器編成ＡＰＩ　３４０は、それに関連するプロトコル３４２およびビューア３４４を有し、ビューア３４４を介して機器編成がエクスポーズされる。プロトコル３４２は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３４４は、機器編成ＡＰＩ　３４０に関係するデータを表示する。機器編成ＡＰＩ　３４０は、ＩＰＣ（プロセス間通信）３４６を介して、管理されるアプリケーション３１４と通信する。ＩＰＣは、同じコンピュータ内の、あるいはネットワークを介して、１つのプログラムと別のプログラムの間でデータを自動的に交換する。ＩＰＣ機能の一例は、ユーザがクリップボードを使用して、１つのファイルから別のファイルに手作業でデータをカットアンドペーストするときに実施されるものである。カウンタは、共有メモリを介して常にパブリッシュされ、機器編成は、要求時に送達される。機器編成ＡＰＩ　３４０は、イベントスキーマに類似のやり方で機器編成クラスのサーフェイス（ｓｕｒｆａｃｅ）を記述するスキーマ３４８も含む。（図示しない）機器編成ログも含まれることがあるが、管理者の多くは、イベントログを使用することを好む。
【００５６】
システム１１０は、コンポーネントおよびモード情報を追跡しキャッシュする記憶部であるカタログ３４７を含む。このモード情報は、インストール時にマニフェストから送られ、その一部は動的であり、実行時に更新される。カタログ３４７は、カタログＡＰＩを含み、カタログ３４７内に記憶されるデータへのアクセスを提供する。これらのデータのいくつかのタイプとして、イベント、カウンタ、機器編成、および構成のデータが挙げられる。カタログ３４７へのアクセスは、プロトコル３５１およびビューア３５３によってうまく進む。中央構成データベースは、複数の管理ノード全体にわたってロールアップされた、あるいは集計されたカタログの一覧を含む。
【００５７】
システム１１０は、アプリケーションまたはサービス３１４と通信するイベントＡＰＩ　３５０を含み、それによって、アプリケーション３１４を管理するのに使用するイベントの実装および追跡がうまく進む。イベントＡＰＩ　３５０は、発生するすべてのイベントの記憶部として働くイベントログ３５２に接続される。イベントＡＰＩ　３５０は、それに関連するプロトコル３５４およびビューア３５６を有する。プロトコル３５４は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３５６は、イベントＡＰＩ　３５０に関係するデータを表示する。イベントＡＰＩ　３５０とアプリケーション３１４の通信は、それらの間で渡されるデータの構造および内容を定義するイベントスキーマ３５８に従う。これらのイベントは、それらが記述または発生したときにパブリッシュされる。このスキーマは、イベントのサーフェイスを記述する

。
【００５８】
システム１１０は、アプリケーション３１４と通信する自動化ＡＰＩ　３６０を含み、それによって、普通ならアプリケーション３１４と対話的に行われる手順の自動化がうまく進む。自動化ＡＰＩ　３６０は、それに関連するプロトコル３６２およびシェル３６４を有する。プロトコル３６２は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。シェル３６４は、ユーザと自動化ＡＰＩ　３６０の対話をうまく進めるユーザインターフェースを提供し、それによって、自動化プロセスおよびこれらの自動化プロセスのユーザコントロールに関係するデータの入力および表示が行われる。
【００５９】
システム１１０はさらに、アプリケーション３１４および自動化ＡＰＩ　３６６と通信するスケジュールされたタスクＡＰＩ　３６６を含む。スケジュールされたタスクＡＰＩ　３６６により、少なくとも自動化ＡＰＩ　３６０および管理されるアプリケーション３１４についてスケジューリングを行うジョブまたはプログラムがうまく進む。スケジュールされたタスクＡＰＩ　３６６は、実行すべきジョブの一覧を維持し、それに従ってリソースを割り当てる。スケジュールされたタスクＡＰＩ　３６６は、それに関連するプロトコル３６８およびビューア３７０を有する。プロトコル３６８は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３７０は、スケジュールされたタスクＡＰＩ　３６６に関係するデータを表示する。タスクスキーマ３７２は、タスクＡＰＩと他のコンポーネントの間で渡されるデータの構造および内容を定義する。
【００６０】
自動化およびタスクのデータは、タスクモデルおよびコマンドレットモデルから受け取る。これらの機能は、管理シェルを介してローカルまたはリモートで自動化し得る。このスケジューリングシステムは、これらの機能、例えば午前３時のバックアップを実行し得る。
【００６１】
図３Ｃで説明したコンポーネントは、ローカルな実施形態のものを表し、図３Ｄのコンポーネントは、多くの機械およびソフトウェアのシステム全体にわたって解析が行われる分散実施形態に関連するものを表し得ることを理解されたい。そのため、分散実施形態では、図３Ｄのコンポーネントは、図３Ｃのローカルシステムの少なくとも１つと通信する。ただし典型的には、複数のこのようなローカルな実施形態は、有線および／または無線の方式をとる。ローカルな実施形態では、システム１１０は、ローカル監視サービスＡＰＩ　３６５を含めて、図３Ｄのコンポーネントのいずれか、またはすべても含み得る。ローカル監視サービスＡＰＩ　３６５は、プロトコル３６７、ビューア３６９、およびスキーマ３７１も含み、これらはそれぞれ、他のＡＰＩのこのようなコンポーネントに類似の機能をうまく進める。分散実施形態では、ローカル監視サービスＡＰＩ　３６５は、以下で説明する分散監視サービスに監視情報を渡す。
【００６２】
次に図３Ｄを参照すると、モデルに基づく管理アーキテクチャのシステムコンポーネント１１０の管理に関連するＡＰＩのブロック図が示されている。構成データベースサブコンポーネント３７４が提供され、このサブコンポーネントに、中央構成ＡＰＩ　３７６を介してアクセスおよび制御が提供される。中央構成ＡＰＩ　３７６は、システム１１０のすべてのサブコンポーネントに接続される。中央構成ＡＰＩ　３７６は、それに関連する通信および対話用のプロトコル３７８およびビューア３８０、ならびにアサーションおよび既定値などの構成設定および属性のシェイプ（ｓｈａｐｅ）を記述するスキーマコンポーネント３８２を有する。プロトコル３７８は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。
【００６３】

管理システムの運用関連データ、例えば、報告、現在の状態、および履歴データなどのリポジトリ（ｒｅｐｏｓｉｔｏｒｙ）として働く運用データベースサブコンポーネント３８３も提供される。監視ＡＰＩ　３８４は、運用データベース３８３およびモデルに基づく管理システムのすべてのサブコンポーネントに接続される。監視ＡＰＩ　３８４はさらに、それに関連するプロトコル３８５、ビューア３８６、およびスキーマ３８７を有する。プロトコル３８５は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３８６は、監視ＡＰＩ　３８４に関係するデータを表示する。スキーマ３８７は、運用データベース３８３全体の定義を、少なくともその構造と、この構造内の各データ要素が含み得る内容のタイプとに関して提供する。

【００６４】

中央構成は、すべてのＡＰＩに接続することができ、構成の詳細を設定するために管理者が使用するものである。構成の詳細は、どの機械にアプリケーションをインストールすべきかなど、分散アプリケーションのシナリオついての詳細を含み得る。構成は、監視構成も含み得る。例えば、すべての機械は、５分間にわたってＣＰＵの利用率が８０％未満にならないことを示さなければならない。そのため、この監視システムは、構成システムを使用する。監視は、アプリケーションが１つのモデルごとに、挙動し、構成され、インストールされることを、管理者が管理システムを介して保証する方法である。監視は、想定される機能、所望のセキュリティの程度、正しく実施されること、およびユーザの想定どおりにデータが送達されることを保証することも含む。そのため、監視は、これらすべての領域にまたがる。全体的なプロセスは、インストールし、構成し、要求時にタスクを実行し、イベントを消費し、機器編成、構成を提供し、データおよび結果を記憶することである。健全性マニフェストは、監視システムに作業命令を、監視システムへの命令であるルールの形式で提供する。一般に、このマニフェストは実行時の命令を含み、これら実行時の命令は所望の状態を実装する。

【００６５】

監視サービスは、ローカルサービスであるだけでなく、中央または分散型のメカニズムである。分散実施形態では、健全性は、ローカルマシンの健全性ならびにローカルマシンとリモートマシンの間の関係も含む。例えば、一まとまりの１０台の機械があると仮定し、６台が正しく機能している限り、このシステムは健全であるとみなされる。しかし、５台以下の機械しか動いていない場合、このシステムの健全性の状態は、警戒状態まで劣化している。４台以下の機械しか動いていない場合、このシステムの健全性は障害状態とみなし得る。ハードウェアのアブストラクションにより、１つまたは複数のクラスタマシンに障害が発生するか、あるいはオフラインになった場合、１つまたは複数のバックアップシステムまたはアプリケーション／サービスをオンラインにすることが容易になる。そのため、命令に基づいて、共同のアイドル状態のリソースまたは共有リソースを制御し得る。この機能は、データセンタ環境では特に有用である。システムが、最適な、あるいは少なくとも最低限の機能を確実に維持するために、自動化された動作を実装し得る。

【００６６】

モデルに基づく管理アーキテクチャの一態様により、開発者は、システムが健全であるとみなされるために満たさなければならない基準を表す多数のルールを記述し得る。監視ＡＰＩ　３８４は、ルールの暗示的な同時処理をうまく進めるルールランタイムエンジンを含む。このルールエンジンは、これらのルールを中間形式として表す入力命令を受け取る。これらのルールは、ＲＤＬ（ルール定義言語）を用いて表現される。このルールエンジンは、構成データベース３７４から、ルールコードをインスタンス化するのに用いる構成データも受け取る。翻訳器は、これらの入力命令を読み込み、これらを並列実行形式に変換する。ランタイムエンジンは、これら翻訳された命令を読み込み、並列実行をうまく進める。このルールコードは、ランタイムエンジンに、どのルールを実行するかと、このルールを実行するのに必要とされるパラメータとを指定する構成データをロードすることによってインスタンス化される。ルールのパラメータは、実行時に変更し得る。例えば、問題が検出されたときにのみ、システムに大きな影響を及ぼすルールを実行可能にする。

このように、これらのルールは動的であり、それに従って変更し得る閾値もそうである。監視ＡＰＩ　３８４は、システム１１０のすべてのサブコンポーネントにも接続される。

【００６７】

管理者が使用するための、マニフェストを記憶し編集するサービス３８８も提供される。マニフェストサービス３８８は、それに関連するプロトコル３８９およびビューア３９０を有し、それによって、これらのマニフェスト機能を管理者にエクスポーズする。マニフェストサービス３８８は、プロトコル３８９およびビューア３９０を介してこれらのマニフェストを管理者に供給し、それによって管理者が、インストール前に、これらのマニフェストを閲覧し変更することができる。マニフェストサービス３８８により、アップデートおよびカスタマイズに従って、これらのマニフェストをバージョニング（ｖｅｒｓｉｏｎｉｎｇ）もうまく進む。

【００６８】

モデルに基づく管理システムのすべてのサブコンポーネントに接続されるロールに基づくアクセスＡＰＩ　３９１も提供される。ロールに基づくアクセスＡＰＩ　３９１はさらに、それに関連するプロトコル３９２およびビューア３９３を有する。プロトコル３９２は、システム１１０の他のコンポーネントへのＡＰＩ関連データの通信をうまく進める。ビューア３９３は、ロールに基づくアクセスＡＰＩ　３９１に関係するデータを表示する。このＡＰＩ　３９１は、監視コンポーネントおよび構成コンポーネントの上のレベルで示されており、そのため、モデルに基づく管理システムの様々なコンポーネントおよび態様へのアクセスを全体的に管理する。ロールに基づくアクセスＡＰＩ　３９１は、プロトコル３９２およびビューア３９３を含む必要はない。というのは、これらの機能は、システム１１０の他のコンポーネントによってうまく進めることができるからである。

【００６９】

このシステムは、機械ベースの学習および制御用の分類子３９４も含む。上記で示したように、分類子３９４は、システムの、いくつか例を挙げると、パフォーマンスや健全性を高めるために多くのやり方で使用することができる。機械ベースの学習をうまく進めるために、分類子３９４は、このシステムのすべてのコンポーネントにアクセスし、そのデータを使用し得るように、中央構成サービス３７６に接続される。

【００７０】

次に図３Ｅを参照すると、モデルに基づく管理アーキテクチャのタスクコンポーネント１１２の主要サブコンポーネントが示されている。これらのタスクは、管理タスクモデルによって記述される。これらのタスクは、監視サブコンポーネント３９５、トラブルシューティングサブコンポーネント３９６、および管理サブコンポーネント３９７の３つのサブコンポーネントに分けられる。

【００７１】

監視サブコンポーネント３９５のタスクは、健全性、セキュリティ、パッチ、構成、パフォーマンス、およびアプリケーションデータを監督することを含む。トラブルシューティングサブコンポーネント３９６のタスクは、健全性の状態の診断、警告の処理、ならびにイベントログ、機器編成ログ、およびパフォーマンスログを更新することを含む。管理サブコンポーネント３９７のタスクは、中央構成／ポリシー、スケジューリング、および更新デプロイメントを含む。管理には、単一のシステムの管理だけでなく、例えば、多数の機械、アプリケーション、およびシステム、ならびにポリシー、バックアップ時間、変更、およびアップデートを管理することも含まれる。

【００７２】

モデルに基づく管理アーキテクチャでは、抽象リソースまたは物理リソース、あるいはリソースコレクション（ｃｏｌｌｅｃｔｉｏｎ）を一意に識別するためにＵＲＩを用いる。リソース用のスキーマは、このリソース用のプレースホルダ（ｐｌａｃｅｈｏｌｄｅｒ）を伴うＵＲＩによって識別される。プレースホルダを伴うＵＲＩをＵＲＩテンプレートと呼ぶ。このシステムのカタログは、特定のインスタンスを参照せずに機器編成を記述するために、ＵＲＩテンプレートに依存する。ＵＲＩテンプレートにより、実際に特定のイ

ンスタンスについてのプローブを取り出すことなく、プローブを識別し、このプローブの特徴を理解することができる。インスタンスとは切り離して機器編成をあらかじめ定義する機能を保護すると、ルールのデプロイメントおよび記述が比較的容易になり、関連するオペレーティングシステムが管理可能になる。

【００７３】

モデルに基づく管理フレームワークでは、ソフトウェアおよびハードウェアの可用性を監視するために、ＲＤＬを用いてルールの定義を可能にする。ＲＤＬで記述されたルールは、ランタイムエンジンによって監視サービスの一部として実行される。ＲＤＬの目的は、アサーションをテストし、実行時の情報を用いて制約を実行し、推測を行い、相関を実施し、動的テストの結果を他のコンポーネントに送ることである。ＲＤＬによりルールタイプ（すなわちクラス）を定義し、別のＸＭＬ（拡張マークアップ言語）ドキュメントを用いて、インスタンス化に必要なパラメータ値を指定することによってこのルールタイプのインスタンスを生成する。問題の検出、診断、解決、検証、および警告を行うためにこのシステムが取るべきステップのシーケンスを記述するためにスキーマが存在する。このシーケンスが、監視システムによって、モデルに記述され、マニフェストに表現され、実行／管理される。

【００７４】

モデルに基づく管理フレームワークでは、前に示したように、イベントおよびパフォーマンスカウンタの更新値を用いて、サービスおよびテストまたは統合的トランザクションの健全性モデル（またはステータス）を示す。健全性モデル３０１は、サービスまたはコンポーネントにどのように障害が発生し得るかについてのグラフィカルな、かつ／またはテキストによる表現であり、サービスの様々なイベントおよびパフォーマンスカウンタの重大さを管理者が理解し、観察された機器編成データに基づいてなんらかの動作を取るかどうかを効率的に判断する助けになる。開発者は、健全性モデル３０１を、次いで、このモデルおよびソースコードアトリビューションから生成される対応するファイルとともに構築する。

【００７５】

健全性モデル３０１は、コンポーネントの依存性に加えて、コンポーネントの関係の記述を含む。検出された問題の状況に応じて、システムは、関係ツリーを移動し、他のコンポーネントの健全性に基づいて根本的原因を決定しようと試みることができる。この手法は、モデルおよびマニフェストによって補佐される。

【００７６】

ここで開示したアーキテクチャは、サービス定義モデルシステムに適用される。本特許の譲受人の特許出願の主題であるこのアーキテクチャの様々な態様の第１のものは、２００３年１０月　　出願の「Architecture for Distributed Computing System and Automated Design, Deployment, and Management of Distributed Applications」という名称の米国特許出願第　　号明細書であり、第２のものは、２００３年１０月　　出願の「Integrating Design, Deployment, and Management Phases for an Application」という名称の米国特許出願第　　号明細書である。

【００７７】

次に図４を参照すると、モデルに基づく管理のプロセスの流れ図が示されている。説明を簡単にするために、例えば流れ図の形式で本明細書に示す１つまたは複数の方法を一連の動作として示し説明するが、本発明は、動作の順序によって限定されず、いくつかの動作は、本発明に従って、本発明で示し説明するものと異なる順序で、かつ／または他の動作と同時に実施し得ることを理解かつ認識されたい。例えば、一連の相互に関連する状態またはイベントとして、状態図などの形で方法を代わりに表現し得るはずであることが当業者には理解かつ認識されよう。さらに、本発明による方法を実施するのに、例示した動作が必ずしもすべて必要とされないことがある。

【００７８】

４００で、インストールすべきアプリケーションまたはサービスを、そのコンポーネン

トに関して記述する。４０２で、このアプリケーションまたはサービスを、機能、構成、セキュリティ、およびパフォーマンスについての所望の状態の形で記述する。４０４で、この記述は、インストール中にアプリケーションまたはサービスとともに提供され、それによって、システムがこの記述を用いて、システムの管理サービスを構成する。次いで、このプロセスは、停止（Ｓｔｏｐ）ブロックに達する。

【００７９】

次に図５を参照すると、モデルに基づく管理を実施するプロセスのより詳細な流れ図が示されている。５００で、アプリケーションのコンポーネント、健全性の状態および復旧、構成の設定、ならびに管理タスクに関するモデルを形成する。５０２で、ユーザは、環境に応じて、システム／ルール／モデルをカスタマイズする。５０４で、ソースコードにアトリビューションを挿入して、監視を行うための機器編成および論理を示す。５０６で、モデル情報およびソースコードアトリビューションのマニフェストを提供し、それを管理システムのサービスが使用する。このマニフェストは、管理システムのサービスが使用できるように機械可読形式で提供される。５０８で、マニフェスト情報に基づいて、管理システムのサービスの１つまたは複数を構成する。５１０で、システムによるコマンドレットの登録、スケジュールの設定など、マニフェスト内でこのアプリケーションについての管理タスクを定義する。次いで、このプロセスは、停止（Ｓｔｏｐ）ブロックに達する。

【００８０】

次に図６を参照すると、モデルに基づく管理の所望の状態を実施するプロセスの流れ図が示されている。６００で、マニフェストから所望の状態にアクセスする。６０２で、依存性を検証し、必要なファイル、設定、およびセキュリティ機能だけをインストールする。６０４で、マニフェストの指定どおりにイベントをサブスクライブし、転送する。６０６で、機器編成データおよびカウンタデータ、ならびに実施されるテストおよび統合的トランザクションを周期的に収集する。６０８で、自動管理タスクを実施する。６１０で、プログラム機能へのアクセスを制限する。ただし、モデルに基づく管理をうまく進めることにこれを含める必要はない。６１２で、問題を検出し、根本的問題を診断し、是正措置を講じ、介入時にシステム管理者に通知する。６１４で、上記すべてについてのポリシーをカスタマイズして、多くの他のタイプの機械およびシステムに適用する。次いで、このプロセスは、停止（Ｓｔｏｐ）ブロックに達する。

【００８１】

次に図７を参照すると、開示したアーキテキチャを実行するように動作可能なコンピュータのブロック図が示されている。本発明の様々な態様についての追加の状況を提供するために、図７および以下の検討は、本発明の様々な態様を実施し得る適切なコンピューティング環境７００を簡潔かつ全体的に説明することを意図している。１つまたは複数のコンピュータ上で実行し得るコンピュータにより実行可能な命令の一般的な状況において本発明をこれまで説明してきたが、当業者には、他のプログラムモジュールと組み合わせて、かつ／または、ハードウェアおよびソフトウェアの組合せとして、本発明を実施することもできることが理解されよう。一般に、プログラムモジュールは、特定のタスクを実施し、また特定の抽象データタイプを実装するルーチン、プログラム、コンポーネント、データ構造などを含む。さらに、当業者には、シングルプロセッサまたはマルチプロセッサのコンピュータシステム、ミニコンピュータ、大型コンピュータ、ならびにパーソナルコンピュータ、ハンドヘルドコンピューティング装置、マイクロプロセッサベースまたはプログラム可能な民生用電子機器などを含めて、他のコンピュータシステム構成によって本発明の方法を実施し得ることが理解されよう。これらはそれぞれ、１つまたは複数の関連装置に動作可能に結合し得る。例示した本発明の態様は、通信ネットワークを介して接続された遠隔処理装置によってある種のタスクが実施される分散コンピューティング環境で実施することもできる。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は、ローカルおよびリモートのメモリ記憶装置に配置することができる。

【００８２】

再度図７を参照すると、本発明の様々な態様を実施するための、コンピュータ７０２を含む環境の例７００が示されている。コンピュータ７０２は、処理装置７０４、システムメモリ７０６、およびシステムバス７０８を含む。システムバス７０８は、システムメモリ７０６を含めてシステムコンポーネントを処理装置７０４に接続するが、システムコンポーネントの例はこれに限定されるものではない。処理装置７０４は、様々な市販のプロセッサのいずれかとし得る。処理装置７０４として、デュアルマイクロプロセッサその他のマルチプロセッサアーキテキチャも使用し得る。

【００８３】

システムバス７０８は、様々な市販のバスアーキテクチャのいずれかを利用して、（メモリコントローラ付き、またはメモリコントローラを伴わない）メモリバス、ペリフェラルバス、およびローカルバスをさらに相互接続し得るいくつかのタイプのバス構造のいずれかとし得る。システムメモリ７０６は、ＲＯＭ（読出し専用メモリ）７１０およびＲＡＭ（ランダムアクセスメモリ）７１２を含む。ＢＩＯＳ（基本入出力システム）は、ＲＯＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭなどの不揮発性メモリ７１０内に格納される。このＢＩＯＳは、例えば起動時に、コンピュータ７０２内の要素間で情報を転送する助けとなる基本ルーチンを含む。ＲＡＭ　７１２は、データをキャッシュするスタティックＲＡＭなどの高速ＲＡＭも含み得る。

【００８４】

コンピュータ７０２は、ハードディスクドライブ７１４、（例えば、リムーバブルディスク７１８に対して読書きを行う）磁気ディスクドライブ７１６、および（例えば、ＣＤ－ＲＯＭディスク７２２を読み込み、また、ＤＶＤ（デジタルビデオディスク）などの他の大容量光メディアに対して読書きを行う）光ディスクドライブ７２０をさらに含む。ハードディスクドライブ７１４、磁気ディスクドライブ７１６、および光ディスクドライブ７２０はそれぞれ、ハードディスクドライブインターフェース７２４、磁気ディスクドライブインターフェース７２６、および光ドライブインターフェース７２８よってシステムバス７０８に接続し得る。これらのドライブおよびそれらに関連するコンピュータ可読メディアは、データ、データ構造、コンピュータにより実行可能な命令などを記憶する不揮発性記憶装置を提供する。コンピュータ７０２の場合、これらのドライブおよびメディアは、適切なデジタルフォーマットでブロードキャストプログラムを記憶することに対応する。上記のコンピュータ可読メディアの説明では、ハードディスク、リムーバブル磁気ディスク、およびＣＤについて言及したが、当業者には、ｚｉｐドライブ、磁気カセット、フラッシュメモリカード、デジタルビデオディスク、カートリッジなど、コンピュータによって読込み可能な他のタイプのメディアも、例示の動作環境で使用し得ること、さらに、このようなメディアはいずれも、本発明の方法を実施するコンピュータにより実行可能な命令を収容し得ることが理解されよう。

【００８５】

オペレーティングシステム７３０、１つまたは複数のアプリケーションプログラム７３２、他のプログラムモジュール７３４、およびプログラムデータ７３６を含めて、いくつかのプログラムモジュールをこれらのドライブおよびＲＡＭ　７１２内に記憶することができる。オペレーティングシステム、アプリケーション、モジュール、および／またはデータの全部または一部を、ＲＡＭ　７１２内にキャッシュすることもできる。

【００８６】

様々な市販のオペレーティングシステムまたはオペレーティングシステムの組合せによって、本発明を実施し得ることを理解されたい。

【００８７】

ユーザは、キーボード７３８、およびマウス７４０などのポインティングデバイスを介してコンピュータ７０２にコマンドおよび情報を入力することができる。（図示しない）他の入力装置は、マイクロホン、ＩＲリモートコントロール、ジョイスティック、ゲームパッド、衛星パラボラアンテナ、スキャナなどを含み得る。上記その他の入力装置は、システムバス７０８に結合されたシリアルポートインターフェース７４２を介して処理装置

７０４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「ＵＳＢ」（ユニバーサルシリアルバス）、ＩＲインターフェースなど、他のインターフェースによって接続することができる。モニタ７４４その他のタイプのディスプレイ装置も、ビデオアダプタ７４６などのインターフェースを介してシステムバス７０８に接続される。コンピュータは一般に、モニタ７４４に加えて、スピーカ、プリンタなどの（図示しない）他の周辺出力装置も含む。

【００８８】

コンピュータ７０２は、１つ（または複数）のリモートコンピュータ７４８など、１つまたは複数のリモートコンピュータへの有線および／または無線による通信を介した論理接続部を利用するネットワーク環境で動作し得る。１つ（または複数）のリモートコンピュータ７４８は、ワークステーション、サーバコンピュータ、ルータ、パーソナルコンピュータ、ポータブルコンピュータ、マイクロプロセッサベースの娯楽機器、ピアデバイス（ｐｅｅｒ　ｄｅｖｉｃｅ）その他一般のネットワークノードとすることができ、一般に、コンピュータ７０２に関連して上記で説明した要素の多くまたはすべてを含むが、簡単にするために図にはメモリ記憶装置７５０だけを示す。図に示す論理接続部は、ＬＡＮ（ローカルエリアネットワーク）７５２およびＷＡＮ（ワイドエリアネットワーク）７５４を含む。このようなネットワーク環境は、一般事務所、企業規模の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一般的なものである。

【００８９】

ＬＡＮネットワーク環境で用いられるとき、コンピュータ７０２は、有線または無線による通信ネットワークインターフェースまたはアダプタ７５６を介してローカルネットワーク７５２に接続される。アダプタ７５６により、ＬＡＮ　７５２への有線または無線による通信が容易になり得る。ＬＡＮ　７５２は、無線アダプタ７５６と通信するためにＬＡＮ　７５２上に配設された無線アクセスポイントも含み得る。ＷＡＮネットワーク環境で用いられるとき、コンピュータ７０２は一般に、モデム７５８を含むか、または、ＬＡＮ上の通信サーバに接続されるか、あるいは、インターネットなどのＷＡＮ７５４を介して通信を確立するための他の手段を有する。内蔵型または外付け、および有線または無線による装置とし得るモデム７５８は、シリアルポートインターフェース７４２を介してシステムバス７０８に接続される。ネットワーク環境では、コンピュータ７０２に関連して示すプログラムモジュールまたはその一部は、リモートメモリ記憶装置７５０に格納することができる。図に示すネットワーク接続部は例であり、コンピュータ間で通信リンクを確立する他の手段を使用し得ることを理解されたい。

【００９０】

コンピュータ７０２は、任意の無線装置、または無線通信の形で動作可能に配設されたエンティティ、例えば、プリンタ、スキャナ、デスクトップ型および／またはポータブル型のコンピュータ、ポータブルデータ端末、無線により検出可能なタグに関連する任意の機器または場所（例えば、キオスク、新聞・雑誌販売所、休憩室）、ならびに電話と通信するように動作可能である。これは、少なくともＷｉ－ＦｉおよびＢｌｕｅｔｏｏｔｈ（登録商標）による無線技術を含む。このように、通信は、従来方式のネットワークの場合と同様に所定の構造とすることもできるし、あるいは、単に少なくとも２つの装置間のその場限りの通信とすることもできる。

【００９１】

Ｗｉ－Ｆｉすなわちワイヤレスフィデリティ（Ｗｉｒｅｌｅｓｓ　Ｆｉｄｅｌｉｔｙ）により、自宅のソファから、またはホテルの部屋のベッドから、あるいは仕事での会議室から、電線なしにインターネットに接続することができる。Ｗｉ－Ｆｉは携帯電話のような無線技術であり、それによって、基地局の範囲内であればどこでも、例えばコンピュータなどの装置が屋内外でデータの送受信を行うことができる。Ｗｉ－Ｆｉネットワークは、安全で、信頼性が高く、高速な無線接続性を提供するＩＥＥＥ８０２．１１（ａ、ｂ、ｇなど）と呼ぶ高周波技術を利用する。Ｗｉ－Ｆｉネットワークを使用して、コンピュータの相互接続、インターネットへの接続、（ＩＥＥＥ８０２．３すなわちイーサネット（

登録商標）を使用する）有線ネットワークへの接続が可能である。Ｗｉ－Ｆｉネットワークは、無認可の２．４ＧＨｚおよび５ＧＨｚの高周波帯域で、１１Ｍｂｐｓ（８０２．１１ｂ）または５４Ｍｂｐｓ（８０２．１１ａ）のデータレートで、あるいは、両方の帯域を含む（デュアルバンド）製品で動作し、そのため、これらのネットワークは、多くの一般事務所で使用されている基本１０ＢａｓｅＴ有線イーサネット（登録商標）ネットワークに類似の現実のパフォーマンスを提供し得る。

【００９２】

次に図８を参照すると、本発明によるコンピューティング環境の例８００の概略ブロック図が示されている。システム８００は、１つまたは複数のクライアント８０２を含む。１つ（または複数）のクライアント８０２は、ハードウェアおよび／またはソフトウェア（例えば、スレッド、プロセス、コンピューティング装置）とし得る。１つ（または複数）のクライアント８０２は、例えば、本発明を利用することによって、１つ（または複数）のクッキーおよび／または関連するコンテキスト情報を収容し得る。システム８００は、１つまたは複数のサーバ８０４も含む。１つ（または複数）のサーバ８０４も、ハードウェアおよび／またはソフトウェア（例えば、スレッド、プロセス、コンピューティング装置）とし得る。サーバ８０４は、例えば、本発明を利用することによって、変換を実施するためのスレッドを収容し得る。クライアント８０２とサーバ８０４の間の１つの可能な通信は、２つ以上のコンピュータプロセス間で送信されるように適合されたデータパケットの形態を取り得る。このデータパケットは、例えば、クッキーおよび／または関連するコンテキスト情報を含み得る。システム８００は、１つ（または複数）のクライアント８０２と１つ（または複数）のサーバ８０４の間の通信をうまく進めるために使用し得る通信フレームワーク８０６（例えば、インターネットなどのグローバル通信ネットワーク）を含む。

【００９３】

通信は、（光ファイバを含めて）有線および／または無線技術によってうまく進めることができる。１つ（または複数）のクライアント８０２は、１つ（または複数）のクライアント８０２に局在する情報（例えば、１つ（または複数）のクッキーおよび／または関連するコンテキスト情報）を記憶するのに使用し得る１つまたは複数のクライアントデータ記憶部８０８に動作可能に接続される。同様に、１つ（または複数）のサーバ８０４は、サーバ８０４に局在する情報を記憶するのに使用し得る１つまたは複数のサーバデータ記憶部８１０に動作可能に接続される。

【００９４】

上記で示したように、ここで開示したモデルに基づく管理アーキテクチャは、企業タイプのシステム管理に適用される。例えば、クライアント８０２の１つは、ローカルなアプリケーションまたはサービスだけでなく、例えばサーバ８０４のリモートノードのアプリケーションまたはサービスも管理し得る。すべての態様が、ローカルクライアントの単一のインスタンスから、複数のネットワークノードのリモートのシステムおよびアプリケーションにまたがる複数のインスタンスまで、健全性の監視をサポートするように適用される。機械ベースの学習を、ローカルなレベルから企業レベルまで、さらにそれを超えて利用し、それによってシステムのパフォーマンスおよび機能を自動化し改善することができる。

【００９５】

以上説明してきたことには、本発明の実施例が含まれる。当然のことながら、本発明を説明するためにコンポーネントまたは方法の考え得るすべての組合せを説明することは不可能であり、本発明の多くの別の組合せおよび並替えが可能であることが当業者には理解されよう。したがって、本発明は、添付の特許請求の範囲の趣旨および範囲に含まれるすべてのこのような改変形態、修正形態、および変形形態を包含することを意図している。さらに、「含む」という用語を詳細な説明または特許請求の範囲において使用する限りにおいては、このような用語は、「備える」という用語が特許請求の範囲において用いられる際には暫定的な言葉として解釈されるように、「備える」と同様に包括的であることが

意図されている。

【図面の簡単な説明】

【００９６】

【図１】本発明によるルールエンジンを使用する、モデルに基づく管理アーキテクチャを示す図である。

【図２】モデルに基づく管理アーキテクチャの主要コンポーネントを記述することに関係する図面マップを示す図である。

【図３Ａ】モデルに基づく管理アーキテクチャのモデルコンポーネントに関係するブロックを示す図である。

【図３Ｂ】モデルに基づく管理アーキテクチャのマニフェストコンポーネントに関係するブロックを示す図である。

【図３Ｃ】モデルに基づく管理アーキテクチャに従ってアプリケーションまたはサービスを管理するのに使用するシステムコンポーネントのコアシステムＡＰＩのブロック図である。

【図３Ｄ】モデルに基づく管理アーキテクチャのシステムコンポーネントの管理に関連するＡＰＩのブロック図である。

【図３Ｅ】モデルに基づく管理アーキテクチャのタスクコンポーネントの主要サブコンポーネントを示す図である。

【図４】モデルに基づく管理のプロセスの流れ図である。

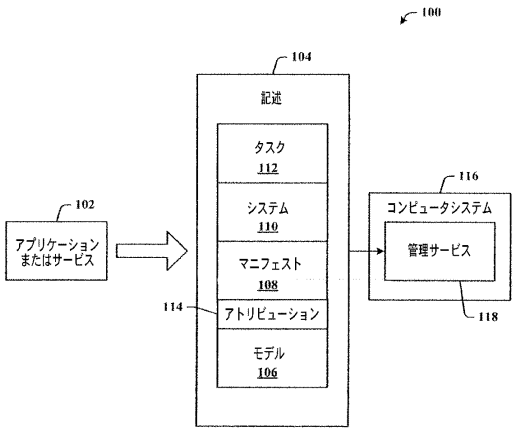
【図５】モデルに基づく管理を実施するプロセスのより詳細な流れ図である。

【図６】モデルに基づく管理の所望の状態を実施するプロセスの流れ図である。

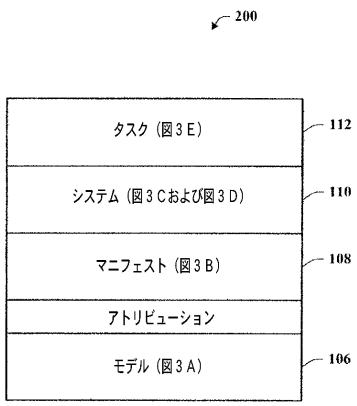
【図７】本発明によるアーキテキチャを実行するように動作可能なコンピュータのブロック図である。

【図８】本発明によるコンピューティング環境の例の概略ブロック図である。

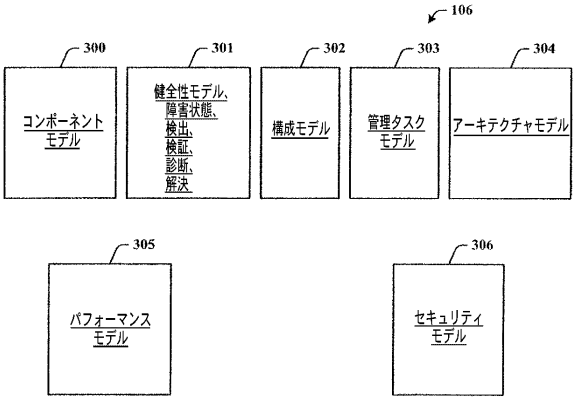
【図１】



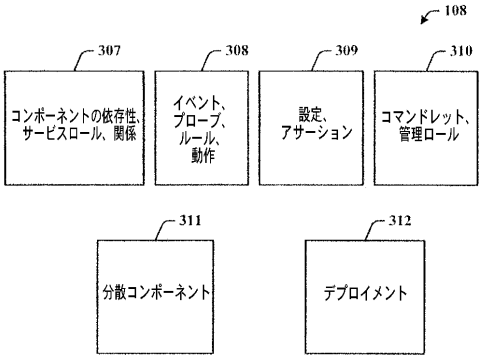
【図２】



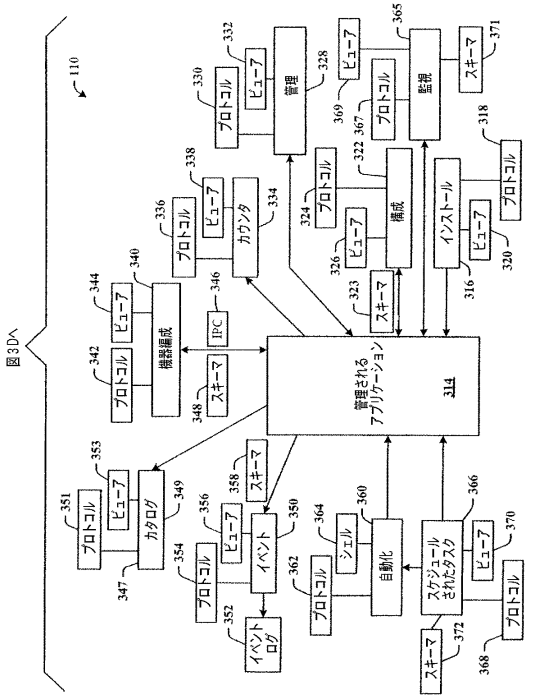
【図３Ａ】



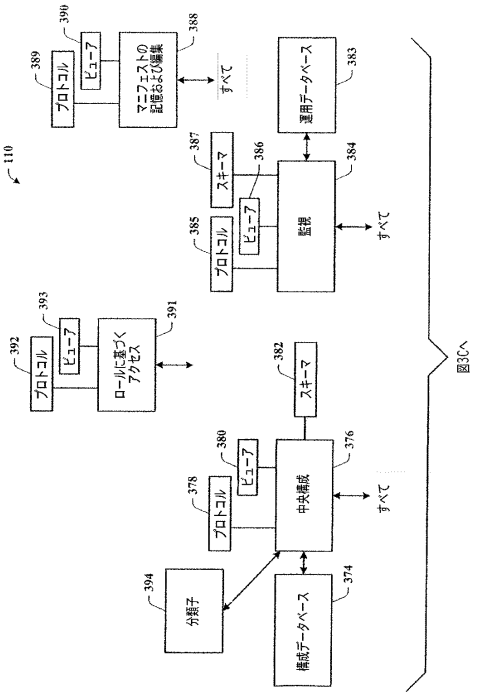
【図３Ｂ】



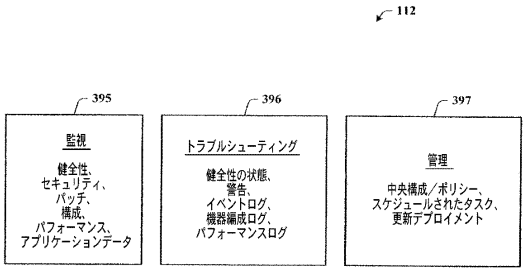
【図３Ｃ】



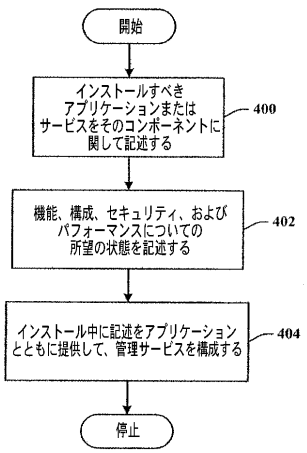
【図３Ｄ】



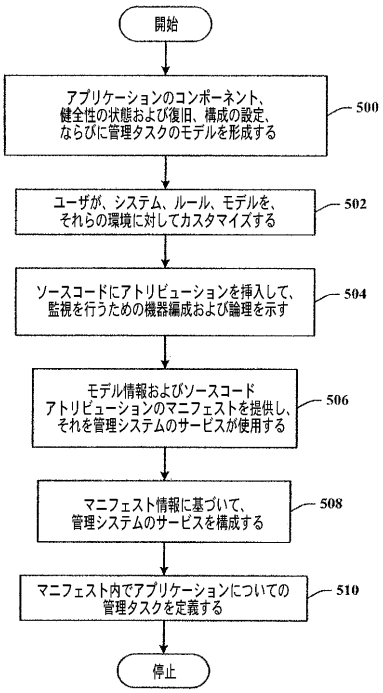
【図３Ｅ】



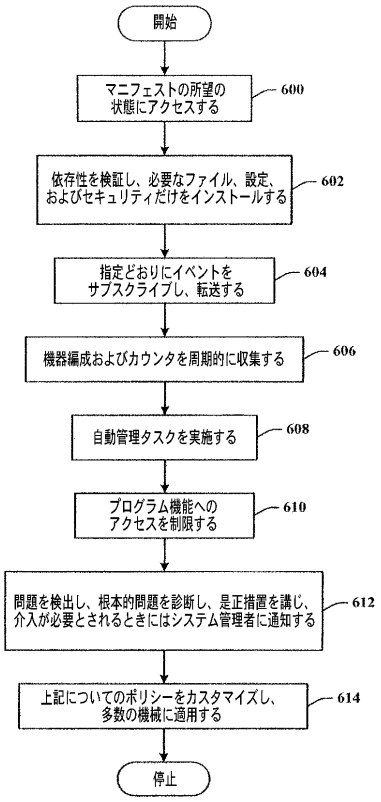
【図４】



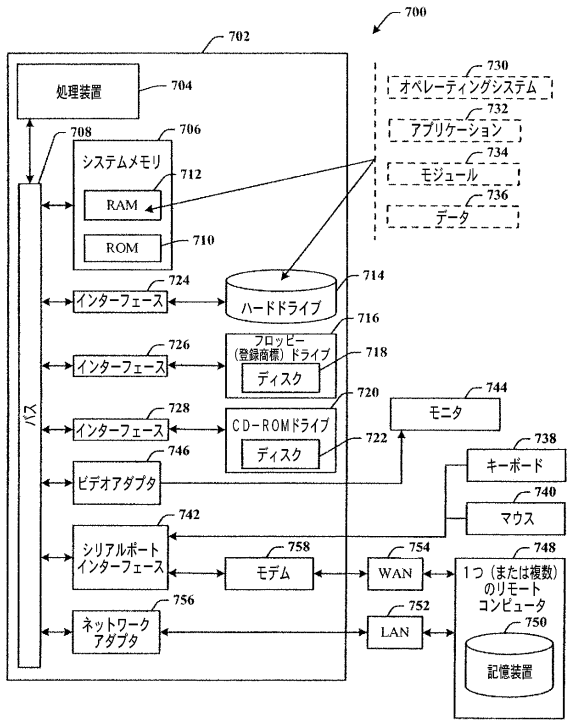
【図５】


開始
アプリケーションのコンポーネント、健全性の状態および復旧、構成の設定、ならびに管理タスクのモデルを形成する
500
ユーザが、システム、ルール、モデルを、それらの環境に対してカスタマイズする
502
ソースコードにアトリビューションを挿入して、監視を行うための機器編成および論理を示す
504
モデル情報およびソースコードアトリビューションのマニフェストを提供し、それを管理システムのサービスが使用する
506
マニフェスト情報に基づいて、管理システムのサービスを構成する
508
マニフェスト内でアプリケーションについての管理タスクを定義する
510
停止

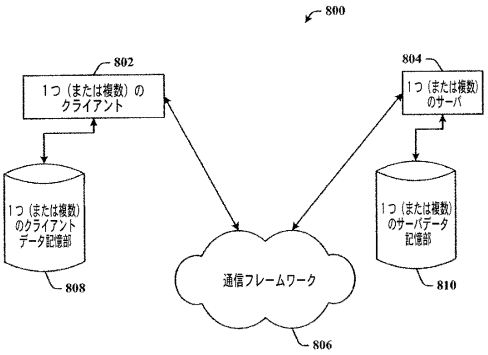
【図６】


開始
マニフェストの所望の状態にアクセスする
600
依存性を検証し、必要なファイル、設定、およびセキュリティだけをインストールする
602
指定どおりにイベントをサブスクライブし、転送する
604
機器編成およびカウンタを周期的に収集する
606
自動管理タスクを実施する
608
プログラム機能へのアクセスを制限する
610
問題を検出し、根本的問題を診断し、是正措置を講じ、介入が必要とされるときにはシステム管理者に通知する
612
上記についてのポリシーをカスタマイズし、多数の機械に適用する
614
停止

【図７】


700
702
処理装置
704
システムメモリ
706
708
RAM
712
ROM
710
バス
インターフェース
724
ハードドライブ
714
インターフェース
726
フロッピー（登録商標）ドライブ
716
ディスク
718
インターフェース
728
CD-ROMドライブ
720
ディスク
722
ビデオアダプタ
746
シリアルポートインターフェース
742
モデム
758
ネットワークアダプタ
756
WAN
754
LAN
752
1つ（または複数）のリモートコンピュータ
748
記憶装置
750
モニタ
744
キーボード
738
マウス
740
オペレーティングシステム
730
アプリケーション
732
モジュール
734
データ
736

【図８】


800
1つ（または複数）のクライアント
802
1つ（または複数）のサーバ
804
1つ（または複数）のクライアントデータ記憶部
808
1つ（または複数）のサーバデータ記憶部
810
通信フレームワーク
806

フロントページの続き

(31)優先権主張番号 10/693,072
(32)優先日 平成15年10月24日(2003.10.24)
(33)優先権主張国 米国(US)

(72)発明者 ラドゥ アール．パランカ
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 イェルク ティ．プフェニング
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 アレクサンダー エム．サットン
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 マーク アール．ブラウン
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

審査官 川崎 優

(56)参考文献 特開２００４－２５２９７５（ＪＰ，Ａ）
特表２００６－５２００２７（ＪＰ，Ａ）
Bigus,J.P., Hellerstein,J.L.,Jayram,T.S., and Squillante,M.S., AutoTune: A Generic Agent for Automated Performance Tuning , Proc. of Practical Application of Intelligent Agents and Multi Agent Technology, ２０００年, ＵＲＬ, http://www.research.ibm.com/PM/AutoTune.doc

(58)調査した分野(Int.Cl., ＤＢ名)
G06F 9/44,445,46-54,11/28-36