(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4603106号
(P4603106)
(45)発行日　平成22年12月22日(2010.12.22)　(24)登録日　平成22年10月8日(2010.10.8)

(51)Int.Cl.　F I
G06F 13/00 (2006.01)　G06F 13/00 530A

請求項の数 24　(全 34 頁)

(21)出願番号	特願平10-284392
(22)出願日	平成10年10月6日(1998.10.6)
(65)公開番号	特開平11-288395
(43)公開日	平成11年10月19日(1999.10.19)
審査請求日	平成17年8月1日(2005.8.1)
(31)優先権主張番号	944383
(32)優先日	平成9年10月6日(1997.10.6)
(33)優先権主張国	米国(US)

(73)特許権者　597004720
サン・マイクロシステムズ・インコーポレーテッド
Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃ．
アメリカ合衆国　カリフォルニア州　９５０５４　サンタ　クララ　ネットワーク　サークル　４１５０
(74)代理人　100089705
弁理士　社本　一夫
(74)代理人　100071124
弁理士　今井　庄亮
(74)代理人　100076691
弁理士　増井　忠弐

最終頁に続く

(54)【発明の名称】オブジェクトの遠隔的ブラウズ方法及びシステム

(57)【特許請求の範囲】
【請求項１】
クライアント・マシンから通信ネットワークを通じてリモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法であって、
（ａ）前記リモート・マシンが、当該リモート・マシンにおける少なくとも１つのオブジェクトを登録することにより、予め修正することなく、前記登録された少なくとも１つのオブジェクトへのリモート・アクセスを可能にするステップと、
（ｂ）前記クライアント・マシンからの要求に応答して、前記リモート・マシンがマシン・ページを動的に生成するステップであって、前記マシン・ページは、ＨＴＭＬページを含み、かつ前記登録された少なくとも１つのオブジェクトを含み、前記マシン・ページの生成は前記登録された少なくとも１つのオブジェクトを前記マシン・ページへマッピングすることを含む、ステップと、
（ｃ）前記クライアント・マシンが、前記クライアント・マシン自体におけるブラウザの動作に応答して、前記リモート・マシンにおけるネットワーク・アダプタと前記マシン・ページとを通じて、前記登録された少なくとも１つのオブジェクトをブラウズするステップと、
を含み、前記リモート・マシンは、エージェント・フレームワークを備えており、ステップ（ｂ）において、前記ネットワーク・アダプタは、前記エージェント・フレームワークへの問い合わせを行い、前記登録された少なくとも１つのオブジェクトを識別することを特徴とする方法。

【請求項２】
請求項１記載の方法において、前記フレームワークは、関連するレポジトリ・オブジェクトを備えており、ステップ（ａ）は、前記少なくとも１つのオブジェクトと前記ネットワーク・アダプタとの少なくとも一方を前記レポジトリ・オブジェクトに登録するステップを含むことを特徴とする方法。
【請求項３】
請求項２記載の方法において、前記エージェント・フレームワークと前記ネットワーク・アダプタとの少なくとも一方はビーンであることを特徴とする方法。
【請求項４】
請求項３記載の方法において、ステップ（ｂ）は、イントロスペクションによってビーン・メソッドを抽出するステップを含むことを特徴とする方法。
【請求項５】
請求項１記載の方法において、前記オブジェクトは、１組のプロパティとアクションを実行する１組のメソッドとイベント及びイントロスペクションのサポートとを含むビーンであることを特徴とする方法。
【請求項６】
請求項５記載の方法において、前記ビーンはＨＴＭＬテーブルとして表現され、前記ＨＴＭＬテーブルにおいて、
第１のコラムは、プロパティ名を含み、
第２のコラムは、プロパティ型を含み、
第３のコラムは、アクセス権を含み、
第４のコラムは、プロパティ値を含むことを特徴とする方法。
【請求項７】
クライアント・マシンから通信ネットワークを通じてリモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法であって、
（ａ）前記リモート・マシンが、当該リモート・マシンにおける少なくとも１つのオブジェクトを登録することにより、予め修正することなく、前記登録された少なくとも１つのオブジェクトへのリモート・アクセスを可能にするステップと、
（ｂ）前記クライアント・マシンからの要求に応答して、前記リモート・マシンがマシン・ページを動的に生成するステップであって、前記マシン・ページは、ＨＴＭＬページを含み、かつ前記登録された少なくとも１つのオブジェクトを含み、前記マシン・ページの生成は前記登録された少なくとも１つのオブジェクトを前記マシン・ページへマッピングすることを含む、ステップと、
（ｃ）前記クライアント・マシンが、前記クライアント・マシン自体におけるブラウザの動作に応答して、前記リモート・マシンにおけるネットワーク・アダプタと前記マシン・ページとを通じて、前記登録された少なくとも１つのオブジェクトをブラウズするステップと、
を含み、前記オブジェクトは、１組のプロパティとアクションを実行する１組のメソッドとイベント及びイントロスペクションに対するサポートとを含むビーンであることを特徴とする方法。
【請求項８】
クライアント・マシンから通信ネットワークを通じてリモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法であって、
（ａ）前記リモート・マシンが、当該リモート・マシンにおける少なくとも１つのオブジェクトを登録することにより、予め修正することなく、前記登録された少なくとも１つのオブジェクトへのリモート・アクセスを可能にするステップと、
（ｂ）前記クライアント・マシンからの要求に応答して、前記リモート・マシンがマシン・ページを動的に生成するステップであって、前記マシン・ページは、ＨＴＭＬページを含み、かつ前記登録された少なくとも１つのオブジェクトを含み、前記マシン・ページの生成は前記登録された少なくとも１つのオブジェクトを前記マシン・ページへマッピン

グすることを含む、ステップと、
（ｃ）前記クライアント・マシンが、前記クライアント・マシン自体におけるブラウザの動作に応答して、前記リモート・マシンにおけるネットワーク・アダプタと前記マシン・ページとを通じて、前記登録された少なくとも１つのオブジェクトをブラウズするステップと、
を含み、前記オブジェクトは被管理オブジェクト・ビーンであることを特徴とする方法。
【請求項９】
クライアント・マシンから通信ネットワークを通じてリモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法であって、
（ａ）前記リモート・マシンが、当該リモート・マシンにおける少なくとも１つのオブジェクトを登録することにより、予め修正することなく、前記登録された少なくとも１つのオブジェクトへのリモート・アクセスを可能にするステップと、
（ｂ）前記クライアント・マシンからの要求に応答して、前記リモート・マシンがマシン・ページを動的に生成するステップであって、前記マシン・ページは、ＨＴＭＬページを含み、かつ前記登録された少なくとも１つのオブジェクトを含み、前記マシン・ページの生成は前記登録された少なくとも１つのオブジェクトを前記マシン・ページへマッピングすることを含む、ステップと、
（ｃ）前記クライアント・マシンが、前記クライアント・マシン自体におけるブラウザの動作に応答して、前記リモート・マシンにおけるネットワーク・アダプタと前記マシン・ページとを通じて、前記登録された少なくとも１つのオブジェクトをブラウズするステップと、
を含み、前記オブジェクトは、前記リモート・マシンにおける１組のビーンの中の１つであり、
クライアント・マシンにおいて、前記クライアント・マシンから遠隔的に修正可能である前記リモート・マシンにおけるビーンの表現を表示するステップと、
表示されたビーン表現の前記クライアント・マシンにおけるユーザ選択に応答して、前記クライアント・マシンにおいて、遠隔的に修正可能なビーン・プロパティを表示するステップと、
前記クライアント・マシンにおけるユーザ入力に遠隔的に応答して、前記ビーンの選択されたパラメータを修正するステップと、
を含むステップ（ｄ）を含むことを特徴とする方法。
【請求項１０】
通信ネットワークを通じてリモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法であって、
前記リモート・マシンが、当該リモート・マシンにおける前記オブジェクトの外部的にアクセス可能なマシン・ページへのマッピングを受け取るステップであって、該ステップは前記オブジェクトを登録するステップを含み、前記マシン・ページはＨＴＭＬページを含む、ステップと、
クライアント・マシンからの要求に応答して、前記リモート・マシンが、登録された前記オブジェクトを含む前記マシン・ページを動的に生成するステップと、
前記クライアント・マシンにおけるブラウザの動作に応答して、前記マシン・ページを通じて前記オブジェクトをブラウズするステップと、
を含み、前記リモート・マシンが前記マッピングを受け取るステップは、
前記リモート・マシンにおけるネットワーク・アダプタの起動と、
前記オブジェクトのエージェント・フレームワークへの登録と、
前記ネットワーク・アダプタが、前記エージェント・フレームワークに問い合わせを行い登録されたオブジェクトを識別するステップと、
を含むことを特徴とする方法。
【請求項１１】

通信ネットワークを通じてリモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法であって、
前記リモート・マシンが、当該リモート・マシンにおける前記オブジェクトの外部的にアクセス可能なマシン・ページへのマッピングを受け取るステップであって、該ステップは前記オブジェクトを登録するステップを含み、前記マシン・ページはＨＴＭＬページを含む、ステップと、
クライアント・マシンからの要求に応答して、前記リモート・マシンが、登録された前記オブジェクトを含む前記マシン・ページを動的に生成するステップと、
前記クライアント・マシンにおけるブラウザの動作に応答して、前記マシン・ページを通じて前記オブジェクトをブラウズするステップと、
を含み、前記ブラウズするステップは、
前記ブラウザの動作に応答して、前記マシン・ページにアクセスするステップと、
前記ブラウザの動作に応答して、前記アクセスされたマシン・ページから前記オブジェクトを選択するステップと、
を含み、前記ブラウズするステップは、前記ブラウザの動作に応答して前記選択されたオブジェクトを修正するステップを更に含むことを特徴とする方法。
【請求項１２】
通信ネットワークを通じてリモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法であって、
前記リモート・マシンが、当該リモート・マシンにおける前記オブジェクトの外部的にアクセス可能なマシン・ページへのマッピングを受け取るステップであって、該ステップは前記オブジェクトを登録するステップを含み、前記マシン・ページはＨＴＭＬページを含む、ステップと、
クライアント・マシンからの要求に応答して、前記リモート・マシンが、登録された前記オブジェクトを含む前記マシン・ページを動的に生成するステップと、
前記クライアント・マシンにおけるブラウザの動作に応答して、前記マシン・ページを通じて前記オブジェクトをブラウズするステップと、
を含み、前記オブジェクトは、１組のプロパティとアクションを実行する１組のメソッドとイベント及びイントロスペクションに対するサポートとを含むビーンであることを特徴とする方法。
【請求項１３】
請求項１０記載の方法において、前記ネットワーク・アダプタがＨＴＭＬアダプタであることを特徴とする方法。
【請求項１４】
通信ネットワークを通じて遠隔的にアクセス可能なコンピュータ・システムであって、リモート・マシンにおける外部的にアクセス可能なマシン・ページにオブジェクトをマッピングするように構成されたマッピング機構であって、前記マシン・ページはＨＴＭＬページを含み、前記マッピング機構は、前記オブジェクトを登録する登録機構を備え、前記マッピング機構は、クライアント・マシンからの要求に応答して、登録された前記オブジェクトを含む前記マシン・ページを動的に生成するように更に構成された、前記マッピング機構を備え、よって、前記オブジェクトはブラウザを用い前記マシン・ページを通じて外部的にアクセス可能であり、前記リモート・マシンにおける前記オブジェクトは予め修正することなくアクセスされ、前記マッピング機構は、エージェント・フレームワークを更に含み、前記登録機構は前記オブジェクトを前記エージェント・フレームワークに登録することを特徴とするシステム。
【請求項１５】
請求項１４記載のシステムであって、前記エージェント・フレームワークへの問い合わせを行い登録されたオブジェクトを識別するように構成されたネットワーク・アダプタを備えていることを特徴とするシステム。
【請求項１６】

請求項１５記載のシステムにおいて、前記登録機構は、前記オブジェクトと前記ネットワーク・アダプタとの少なくとも一方が登録されるレポジトリ・オブジェクトを備えていることを特徴とするシステム。

【請求項１７】

請求項１６記載のシステムにおいて、前記オブジェクトは、１組のプロパティとアクションを実行する１組のメソッドとイベント及びイントロスペクションに対するサポートとを含むビーンであることを特徴とするシステム。

【請求項１８】

請求項１７記載のシステムであって、前記マシン・ページを通じて前記ビーンに遠隔的にアクセスするブラウザを備えていることを特徴とするシステム。

【請求項１９】

請求項１８記載のシステムにおいて、前記ブラウザは、前記マシン・ページからの前記ビーンの選択を可能とすることを特徴とするシステム。

【請求項２０】

請求項１９記載のシステムにおいて、前記ブラウザは、更に、前記選択されたビーンの修正を可能とすることを特徴とするシステム。

【請求項２１】

通信ネットワークを通じて遠隔的にアクセス可能なコンピュータ・システムであって、リモート・マシンにおける外部的にアクセス可能なマシン・ページにオブジェクトをマッピングするように構成されたマッピング機構であって、前記マシン・ページはＨＴＭＬページを含み、前記マッピング機構は、前記オブジェクトを登録する登録機構を備え、前記マッピング機構は、クライアント・マシンからの要求に応答して、登録された前記オブジェクトを含む前記マシン・ページを動的に生成するように更に構成された、前記マッピング機構を備え、よって、前記オブジェクトはブラウザを用い前記マシン・ページを通じて外部的にアクセス可能であり、前記リモート・マシンにおける前記オブジェクトは予め修正することなくアクセスされ、前記オブジェクトは、１組のプロパティとアクションを実行する１組のメソッドとイベント及びイントロスペクションに対するサポートとを含むことを特徴とするシステム。

【請求項２２】

請求項１４記載のシステムにおいて、前記マッピング機構はＨＴＭＬアダプタであることを特徴とするシステム。

【請求項２３】

請求項１４記載のシステムにおいて、前記マッピング機構はソフトウエア機構を備えていることを特徴とするシステム。

【請求項２４】

通信ネットワークを通じてオブジェクトへのリモート・アクセスを可能にする、少なくとも１つの記憶デバイス上のソフトウエア・システムであって、

外部的にアクセス可能なマシン・ページにオブジェクトをマッピングするように構成されたマッピング機構であって、前記マシン・ページはＨＴＭＬページを含み、前記マッピング機構は、前記オブジェクトを登録する登録機構を備え、前記マッピング機構は、クライアント・マシンからの要求に応答して、登録された前記オブジェクトを含む前記マシン・ページを動的に生成するように更に構成された、前記マッピング機構を備えており、よって、ブラウザを用い前記マシン・ページを通じて前記オブジェクトにアクセス可能であり、前記オブジェクトは予め修正することなくアクセスされ、前記オブジェクトは、１組のプロパティとアクションを実行する１組のメソッドとイベント及びイントロスペクションに対するサポートとを含むビーンであることを特徴とするシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、オブジェクト、特に、ビーン(bean)として知られているオブジェクトの制御お

よび／またはアクセスに関するものである。

【０００２】

【従来の技術】

ビーン、例えば、ＪａｖａＢｅａｎｓコンポーネント（ＪａｖａＢｅａｎｓは、サンマイクロシステムズ社(Sun Microsystems, Inc.の登録商標である）は、再利用可能なソフトウエア・コンポーネントであり、ビルダ・ツール（例えば、エディタまたはグラフィカル・ユーザ・インターフェース・ビルダ（ＧＵＩビルダ））において視覚的に操作することができる。ビルダ・ツールの一例に、ＪａｖａＢｅａｎｓ開発キット(JavaBeans Development Kit)がある。ビーンに関する更なる詳細は、多くの異なる書籍において見ることができ、例えば、Synbexによって発行されたLawrence Vanhelsuwe著のMastering JavaBeans（ＪａｖａＢｅａｎｓの習得）と題する本(ISBN-0-7821-2097-0)があげられる。これは、表題に「ＪａｖａＢｅａｎｓ」を有しＪａｖａＢｅａｎｓについて説明する、広い意味で同等な市販の多くの本の一例に過ぎない。Mastering JavaBeansを含むこれらの本の多くは、上述のビーン開発キットに対応する(supply)ものである。

【０００３】

ビーンはそのファンクショナリティ(functionality)が様々に変化するが、典型的に、ある種の共通した定義構造を共有しており、１組のプロパティ、１組のメソッド、ならびにイベント、およびリフレクション(reflection)としても知られているイントロスペクション（introspection:内省）に対するサポート(support)を与える。プロパティは、ビーンのプログラム的な操作を可能とし、ビーンのカスタム化(customization)をサポートする。メソッドはプロパティを実装する。イベントに対するサポートは、ビーンがイベントを発火(fire)することを可能とし、発火することができるイベントを定義する。イントロスペクションに対するサポートは、ビーンのプロパティ、イベントおよびメソッドを、外部から検分することを可能にする。

【０００４】

【発明が解決しようとする課題】

しかしながら、通常、ビーンは、当該ビーンが存在する仮想マシン環境内でのみ、その制御および操作が可能である。米国特許第５，３１５，７０３号および米国特許第５，３６７，６３３号は、変更通知機能を備えた、オブジェクトに基づくシステムについて記載する。しかしながら、これらの特許は、スタンド・アロンのシステムについて記載するに過ぎない。

【０００５】

例えば、ネットワークを通じて、遠隔的にビーンにアクセスしたり、ビーンを制御することができれば望ましいであろう。しかしながら、これは、適切なメソッドをビーン自体の中に実際に含ませ、その必要とされるメソッドを露出させなければ可能ではない。しかしながら、遠隔的なアクセスを可能とするようにビーンを特定的に適合化させなければならないことは望ましくない。したがって、事前の修正をせずに、あらゆるビーンのアクセス、制御、および修正を遠隔的に行うことができれば、望ましいであろう。

【０００６】

したがって、本発明の目的は、この問題に対処することである。

【０００７】

【課題を解決するための手段】

本発明の第１の態様によれば、通信ネットワークを通じて、クライアント・マシンから、リモート・マシンにおけるオブジェクトにアクセスするコンピュータ実装方法が提供され、この方法は、

ａ）少なくとも１つのオブジェクトを、リモート・マシンにおいて登録するステップと、

ｂ）リモート・マシンにおいてマシン・ページを生成し、少なくとも１つの登録したオブジェクトをページに含ませるステップと、

ｃ）クライアント・ステーションにおいてブラウザを用い、リモート・マシンにおいてネットワーク・アダプタおよびマシン・ページを通じて、前記少なくとも１つのオブジェク

トをブラウズするステップと、
から成る。
【０００８】
オブジェクトをマシン・ページと関連付けることにより、ブラウザを用いて、クライアント・マシンにおいて、遠方より、マシンまたはマシン・ページを備えた仮想マシンにおけるオブジェクトにアクセスすることが可能となる。オブジェクト登録の動作は、オブジェクトを予め修正しておくことや、オブジェクトにそれ自体の内部リモート・アクセス・メソッドを含ませることを必要とせずに、ネットワーク・アダプタを通じてオブジェクトへのリモート・アクセスを可能とするように構成される。
【０００９】
本発明は、１組のプロパティ、アクションを実行する１組のメソッド、ならびにイベントおよびイントロスペクションに対するサポートから成るビーンの形態のオブジェクトに対するリモート・アクセスを与える場合に特に適用されるものである。ビーン・メソッドの抽出のために、ネットワーク・アダプタを通じてビーンにアクセスする場合、このイントロスペクションを利用する。フレームワークおよび／またはネットワーク・アダプタも、ビーンとして実装し、柔軟性のある仮想機構構造を備えることが好ましい。
【００１０】
好ましくは、リモート・マシンはエージェント・フレームワークを備えており、ステップ（ｂ）において、ネットワーク・アダプタが、登録されたオブジェクトを識別するように、フレームワークに問い合わせる。これによって、リモート・マシンにおけるオブジェクトに対するアクセスを可能とする、柔軟性および拡張性のある構成が得られる。これは、要求に応じて新たなビーンを追加するための柔軟性のあるメソッドを与える。実際には、一層の改良として、好適な実施形態は、関連するレポジトリ・オブジェクトを備えるフレームワークを採用し、ステップ（ａ）は、オブジェクトおよび／またはネットワーク・アダプタをレポジトリ・オブジェクトに登録するステップを含む。フレームワークおよび／またはネットワーク・アダプタも、ビーンとすることができる。ステップ（ｂ）は、イントロスペクションによってビーン・メソッドを抽出するステップを含むことができる。
【００１１】
好適な実施形態では、前述のマシン・ページは、ビーン即ちオブジェクトが位置する仮想マシンにおけるＨＴＭＬページであり、ネットワーク・アダプタはマシンにおけるＨＴＭＬアダプタである。ビーンは、ＨＴＭＬテーブルとして表すことができ、その際、
第１のコラムがプロパティ名を含み、
第２のコラムがプロパティ型を含み、
第３のコラムがアクセス権（リード／ライト）を含み、
第４のコラムがプロパティプロパティ値を含む。
【００１２】
他の実施形態において、ＨＴＭＬテーブル内に、メソッドおよび／またはイベントに対するサポートを備えることも可能である。
【００１３】
本発明は、アクセスしようとするオブジェクトが被管理マシン内の被管理オブジェクト・ビーンである、ネットワーク管理システムに特に適用されるものである。
【００１４】
オブジェクトは、リモート・マシンにおける１組のビーンの内の１つとすることができ、ステップ（ｄ）は、
クライアント・マシンから遠方より修正可能な、リモート・マシンにおけるビーンの表現を、クライアント・マシンにおいて表示するステップと、
表示されたビーン表現のクライアント・マシンにおけるユーザ選択に応答して、遠方より修正可能なビーン・プロパティをクライアント・マシンにおいて表示するステップと、
クライアント・マシンにおけるユーザ入力に遠方より応答し、ビーンの選択されたパラメータを修正するステップと、を含む。

【００１５】
本発明の別の態様によれば、通信ネットワークを通じてリモートにおけるオブジェクトにアクセスするコンピュータ実装方法が提供され、この方法は、
前記オブジェクトを、前記リモート・マシンにおいて、外部からアクセス可能なマシン・ページにマッピングするステップと、
ブラウザを用いて、前記マシン・ページを通じて前記オブジェクトをブラウズするステップと、を含む。
【００１６】
本発明の更に別の態様によれば、通信ネットワークを通じて遠隔的にアクセス可能なコンピュータ・システムが提供され、このコンピュータ・システムは、リモート・マシンにおいて、外部からアクセス可能なマシン・ページにオブジェクトをマッピングするように構成されたマッピング機構を備え、これによって、ブラウザを用い前記マシン・ページを通じて、前記オブジェクトを外部からアクセス可能とする。
【００１７】
本発明は、更に、通信ネットワークを通じてオブジェクトに対するリモート・アクセスを可能にするための、少なくとも１つの記憶デバイス上のソフトウエア・システムを提供する。このシステムは、外部からアクセス可能なマシン・ページにオブジェクトをマップするように構成されたマッピング機構を備え、これによって、ブラウザを用い前記マシン・ページを通じて、前記オブジェクトにアクセス可能とする。
【００１８】
【発明の実施の形態】
これより本発明の代表的な実施形態について、一例としてのみ、添付図面を参照しながら説明する。図面において、同様の参照符号は、同様のエレメントに関連するものとする。
【００１９】
図１は、マルチ・ステーション・ネットワークを基本としたシステム１の概略図であり、３箇所のステーション、即ち、ノード、またはマシン３、４、５がネットワーク２を通じて接続されている。ネットワーク２は、公衆電話交換網および／またはローカル・エリア・ネットワーク、および／またはローカル・エリアおよび／またはより広いエリア内の専用ネットワーク接続コンピュータおよびその他の機器、および／またはインターネットまたはイントラネットのようなオープン・ネットワーク、あるいはその組み合わせ、あるいは実際には通信管理情報の交換に対応可能な他のあらゆる形式の通信ネットワークを基本とすることができる。ネットワークは、レベル構造、ピラミッド階層等のような、任意の所望の構造でよい。
【００２０】
ステーション３、４、５の任意の１つ又は複数は、ネットワーク管理ステーションとして、そのコンフィギュレーションを設定することができる。図１に示す例では、ステーション３を管理ステーションとし、ステーション４、５を被管理ステーションとする。いずれの数の管理ステーションおよび被管理ステーションでも設けることができ、図１においてステーションを３箇所としたのは、例示の目的のために過ぎないことは認められよう。また、管理および被管理機能は相互に交換可能であり、実際には、管理機能および被管理機能は、１つの同じステーションにおいてサポートすることが可能である。
【００２１】
ステーションは、多くの形態を取ることができる。例示の目的のために、双方ともコンピュータ・ワークステーションから成るものと仮定する。図２は、コンピュータ・ワークステーションのコンフィギュレーションの概略図である。図２に示すように、これは、ディスプレイ８、キーボードおよびその他の入力デバイス９を有するシステム・ユニット１１を備えたサーバ・コンピュータ１０によって実装されている。図２のＡは、システム・ユニット１１の内容の態様を表す概略ブロック図である。図２のＡに示すように、システム・ユニット１１は、プロセッサ１７、メモリ１８、磁気および／または光ディスク・ドライブ１３、１４、および通信ネットワーク２に接続するための１又は複数の通信ライン１

５に接続するための通信アダプタ１６を含む。図２のＡに示すように、このシステム・ユニットのコンポーネントは、バス配列１９を通じて接続されている。図２および図２のＡは、ルータを形成するためのサーバ・コンピュータに対する、１つの可能なコンフィギュレーションの概略図である。また、代わりのコンフィギュレーションも多数提供可能であることも認められよう。

【００２２】

ワークステーションによって管理ステーションが実装される場合には、１又は複数の管理アプリケーションとインターフェース構造とは、典型的に、ソフトウエアで供給され、このソフトウエアはワークステーションのメモリに格納され、ワークステーションのプロセッサによって実行される。

【００２３】

ワークステーションによって被管理ステーションが実装される場合には、エージェントが、ネットワークを通じた管理アプリケーションからの離れた管理要求に応答し、被管理ステーションにおける被管理オブジェクトへのインターフェースを与える。エージェントおよび被管理オブジェクト構造は、典型的に、ソフトウエアで供給され、このソフトウエアはワークステーションのメモリに格納され、ワークステーションのプロセッサによって実行される。

【００２４】

オブジェクトは、典型的に、１台の機器、当該機器のコンポーネント、機器あるいはそのコンポーネントまたはリソースの動作等をモデル化するために用いられるパラメータおよびメソッドを備えている。

【００２５】

通信ネットワークの管理は、種々のサイズおよび複雑度を有するアプリケーションを必要とする。重要なマネージャ(heavy manager)、中間マネージャ、拡張可能なエージェント、スマート・エージェント(smart agent)およびアプライアンス(appliance)は全て、通信ネットワークの管理において役割を有する。本発明を組み込むネットワーク管理システムは、拡張可能なエージェント・フレームワークを提供し、これらの異なるアプリケーション・タイプの全てを、同一アーキテクチャ上で構築することを可能とする。この拡張可能なエージェント・フレームワークは、ネットワーク管理システムのコンポーネントによって与えられる。このコンテキストにおける拡張可能なエージェント・フレームワークに対する代わりの用語として、「動的フレームワーク」または「オープン・フレームワーク」または「ランタイム・コンポーネント」とすることも可能であるが、ここでは「フレームワーク」または「拡張可能なエージェント・フレームワーク」を用いることにする。

【００２６】

ネットワーク管理システムには、１組のコア管理サービスが供給される。この１組のサービスから選択を行うことができ、更に特定のアプリケーションを開発するために拡張することができる。異なるサービスを静的にまたは動的にフレームワークにロードし、特定のアプリケーションの要件を満たすようにする。

【００２７】

本発明を組み込んだネットワーク・エージェントの一例における被管理オブジェクトは、好ましくは、ビーン（bean）として、更に好ましくは、ＪａｖａＢｅａｎｓコンポーネントとして実装される。ビーン（およびJavabeans コンポーネント）は、再利用可能なソフトウエア・コンポーネントであり、構築ツール(builder tool)（例えば、エディタまたはグラフィカル・ユーザ・インターフェース・ビルダ（ＧＵＩビルダ））において視覚的に操作することができる。構築ツールの一例として、ＪａｖａＢｅａｎｓ開発キットがあげられる。ビーンはそのファンクショナリティが様々に変化するが、典型的に、ある種の共通した定義構造を共有しており、１組のプロパティ、アクションを実行する１組のメソッド、ならびにイベントおよびイントロスペクションに対するサポートを与える。プロパティは、ビーンのプログラム的な操作を可能とし、ビーンのカスタム化(customization)をサポートする。メソッドはプロパティを実装する。イベントに対するサポートは、ビーン

ズがイベントを発火(fire)することを可能とし、発火することができるイベントを定義する。イントロスペクションに対するサポートは、ビーンのプロパティ、イベントおよびメソッドを、外部から検分することを可能にする。ＧＥＴ、ＳＥＴ、ＡＣＴＩＯＮ、ＣＲＥＡＴＥおよびＤＥＬＥＴＥというような動作をサポートすることができる。

【００２８】

エージェントにおける被管理オブジェクトは、それがフレームワークに登録されると直ちに管理可能となる。この構成によって、このネットワーク管理システムにしたがって開発されたエージェントは、エージェントの設計に対する影響を最少に抑えて、管理可能となる。

【００２９】

先に示したように、ネットワーク管理システムの一例は、ＪａｖａＢｅａｎｓコンポーネント・モデルを用いることによって、アプリケーションの開発を容易にする。この例では、コア管理サービスの全てが、ＪａｖａＢｅａｎｓコンポーネントとして用意されている。したがって、広く知られたＪａｖａＢｅａｎｓ開発キットのような、Ｊａｖａアプリケーション・ビルダを用いて、それらへのアクセスが可能になる。被管理オブジェクトは、ＪａｖａＢｅａｎｓコンポーネントとして開発される。ネットワーク管理システム・エージェントにおけるＪａｖａＢｅａｎｓコンポーネントは、局所的にまたは遠隔的にアクセスすることができる。これは、このネットワーク管理システムを用いて被管理オブジェクトを開発する場合、どの通信プロトコルに被管理オブジェクトがアクセスするのかについて知る必要がないことを意味する。

【００３０】

ネットワーク管理システムは、拡張可能エージェントの開発を簡略化する。ネットワーク管理システムが提供する１組のコア管理サービスを拡張し、エージェントが実行されている間に、それにロードすることができる。コア管理サービスの殆どはオプションである。これが意味するのは、ネットワーク管理システムを用いて開発したエージェントは、それが用いるサービスを実装するだけでよいということである。この構造は、異なるサイズおよび複雑度のエージェントの開発を可能にする。

【００３１】

また、ネットワーク管理システムを用いて開発したエージェントは、スマート・エージェントでもある。スマート・エージェントは、管理要求を処理するために必要とされるサービスを提供する。スマート・エージェントでは、処理の多くは、エージェント自体の中で局所的に行うことができるので、エージェントと管理システムとの間のネットワーク接続上の負荷が減少する。

【００３２】

図３は、ネットワーク管理システム・エージェントのアーキテクチャの一態様を示し、ネットワーク管理システムのコンポーネント間の関係を含む。また、図３は、ネットワーク管理システム・エージェントと管理アプリケーションとの間の関係も示す。

【００３３】

図３に示すように、ネットワーク管理システム・エージェントは、Ｊａｖａ仮想マシン（ＶＭ：virtual machine）の内側にある多数のコンポーネントから成る。これらは、

－　ｍ－ビーン２９、
－　フレームワーク２４、
－　コア管理サービス２５、２６、２７、２８、
－　被管理オブジェクト・アダプタ・サーバ３０、３２、３４、３６、３８、を含む。これらのコンポーネントについて、以下で更に詳細に説明する。

【００３４】

被管理オブジェクト（managed object）とは、エージェントによって制御され監視される、リソースのソフトウエア抽象化(software abstraction)である。被管理オブジェクト（例えば２８）のことを、管理ビーンまたはｍ－ビーンと呼ぶ。ネットワーク管理システムの一例では、全てのｍ－ビーンはＪａｖａＢｅａｎｓコンポーネントとして実装される。

したがって、先に述べたＪａｖａＢｅａｎｓ開発キットのような、従来のJavaアプリケーション・ビルダを用いて、これらにアクセスすることができる。

【００３５】

他のいずれの被管理オブジェクトに関しても、ｍ－ビーンは１組のプロパティを有し、１組のアクションを実行することができ、１組の通知またはイベントを送出することができる。ネットワーク管理システムは、ｍ－ビーンにおけるリード専用プロパティとリード／ライト・プロパティとの間で区別することを可能とする。

【００３６】

ｍ－ビーンは、フレームワーク２４によって登録されると直ちに管理可能となる。ｍ－ビーンが登録されると、オブジェクト名がそれに関連付けられる。このオブジェクト名が、ｍ－ビーン・レポジトリ（以下を参照のこと）内の当該ｍ－ビーンを一意的に識別する。それによって、管理アプリケーションが、管理処理を実行しようとするｍ－ビーンを識別することが可能になる。ｍ－ビーンのオブジェクト名は、任意の名称であり、ｍ－ビーンがどのように実装されるかには、全く依存しない。

【００３７】

フレームワーク２４は管理サービスを制御し、エージェント２０のｍ－ビーンはフレームワーク２４にロードされる。フレームワーク、即ち、ランタイム・コンポーネント２４は、ＪａｖａＢｅａｎｓコンポーネントであり、１組のプロパティ、アクションを実行するための１組のメソッド、ならびにイベントおよびイントロスペクションに対するサポートを備えている。プロパティは、ゲッタ(getter)およびセッタ(setter)プロパティを含む。メソッドは、ゲッタおよびセッタ・プロパティを実装するメソッドを含む。また、フレームワークは、オブジェクト追加機能(add object function)およびオブジェクト除去機能(remove object function)を実行することも可能である。

【００３８】

エージェント２０が、ネットワーク・アダプタを介して受信した管理処理を実行するように要求されたときはいつでも、フレームワーク２４は、適切なサービスをコールし、要求された処理を実行する。また、フレームワーク２４は、ｍ－ビーン２８と被管理オブジェクト・アダプタ・サーバ３０～３８との間の通信も取り扱う。ｍ－ビーンは、フレームワーク２４に問い合わせを行い、フレームワーク２４の同じインスタンスにロードされている他のｍ－ビーン２８に関する情報を得ることができる。フレームワーク２４の１つのインスタンスのみが、仮想マシン２２内で許される。

【００３９】

ネットワーク管理システムは、多数のコア管理サービスを提供する。このシステムの一例では、コア管理サービスは、Ｊａｖａインターフェースとして定義される。コア管理サービスはオプションである。これが意味するのは、ネットワーク管理システムを用いて開発したエージェントは、それが用いるサービスのみを実装すればよいということである。コア管理サービスは、ｍ－ビーンとして登録することができ、これによって、いくつかの管理処理をその上で形成し、そのビヘービャ(behaviour)を調整(tune)することを可能にする。

【００４０】

このシステムの好適な例では、コア管理サービスは、ＪａｖａＢｅａｎｓコンポーネントとして与えられる。したがって、先に述べたＪａｖａＢｅａｎｓ開発キットのような、従来のＪａｖａアプリケーション・ビルダを用いて、これらにアクセスすることができる。

【００４１】

次に、多数のコア管理サービスについて説明する。

【００４２】

ｍ－ビーン・レポジトリ・サービス２７は、ｍ－ビーンへのポインタを得る。

ｍ－ビーンがフレームワーク２４に登録される毎に、フレームワーク２４は、ｍ－ビーン・レポジトリ・サービス２７をコールし、当該ｍ－ビーンの識別子(identity)を格納する。ｍ－ビーンには名称が関連付けられる。ｍ－ビーン・レポジトリ２７に問い合わせをす

ると、エージェントおよびマネージャ・アプリケーションは、ｍ－ビームをその名称によって識別することができる。ｍ－ビーン・レポジトリ・サービス２７は、異なる実装方法が可能である。あるインプリメンテーションでは、永続的なｍ－ビーンを格納するためにリレーショナル・データベースを用い、一方、別のインプリメンテーションでは、非永続的なｍ－ビームを格納するために単純なメモリを用いる。

【００４３】

レポジトリ・ユニットによって与えられる構造を考慮しながら、エージェントのコンポーネント間の関係の別の表現を図４に示す。これは、被管理オブジェクト・アダプタ・サーバ、管理サービス、および非管理オブジェクトに対するビーンのｍ－ビーン・レポジトリ・サービス・ビーンへの登録(registration)を考慮したものである。

【００４４】

フィルタリング・サービス(filtering service)２９は、ｍ－ビーンを管理処理の主題に選択する。選択は、特定の属性の存在および値を基準とする。例えば、フィルタは、属性カラーが赤にセットされている全てのｍ－ビーンを選択することができる。

【００４５】

メタデータ・サービス(metadata service)２５は、ｍ－ビーンの構造に関する情報を与える。例えば、フレームワーク２４は、メタデータ・サービス２５に問い合わせを行い、ｍ－ビーン内のプロパティを得て設定するためのメソッドにアクセスする。メタデータ・サービス２５は、イントロスペクションを実行するＪａｖａＢｅａｎｓ開発キットによって与えられるリフレクションＡＰＩに基づく。

【００４６】

動的クラス・ローディング・サービス(dynamic class loading service)は、新しいクラスをフレームワーク２４にロードする。新しいクラスがロードされるのは、リモート・エントリが、フレームワーク２４にロードされていないクラスのインスタンスの作成を要求するときである。新しいクラスは、リモート・クラス・サーバからロードすることができる。また、動的ローディング・サービスは、ネットワーク管理システム・エージェントの実行中に、コア管理サービスをこれに追加することを可能にする。例えば、フィルタリング・サービスがなくとも、エージェントを開始することができる。そして後に、フィルタリング・サービスが必要とされるときに、それをエージェントに動的に追加することができる。

【００４７】

アクセス制御サービス(access control service)は、ｍ－ビーンへのアクセスを制御するために提供することができる。ｍ－ビーン上で管理処理を実行しようとする前に、その処理が有効であることを確認するために、アクセス制御サービスに問い合わせするように、フレームワーク２４を構成することができる。

【００４８】

イベント・サービス(event service)は、ｍ－ビーンからイベント報告を受け取り、それらを受け取ることを要求した任意のエンティティにそれらを送出するために提供することができる。

【００４９】

関係サービス(relationship service)は、ｍ－ビーン間の関係を、それらが要求されるときに定義することを可能にするために提供することができる。関係を前もって定義しておく必要はない。ｍ－ビーン間の関係に関する情報は、ｍ－ビーン自体には格納されるのではなく、関係サービスから得ることができる。

【００５０】

動的ネーティブ・ライブラリ・ローディング・サービス(dynamic native library loading service)は、ネーティブ・コードをフレームワーク２４にロードするために提供することができる。ネーティブ・ライブラリは、ネーティブ・コードを含む新しいクラスをロードするときに、ロードすることができる。ロードされるライブラリは、ハードウエア・プラットフォーム、およびフレームワークが実行されるオペレーティング・システムに依存

する。ネーティブ・ライブラリは、離れたエンティティからロードすることができる。
【００５１】
続いて、被管理オブジェクト・アダプタ・サーバ３０～３８の説明を行う。
【００５２】
被管理オブジェクト・アダプタ・サーバとは、通信プロトコルの抽象化を与えるプロトコル・アダプタである。それぞれの被管理オブジェクト・アダプタ・サーバは、特定の通信プロトコルを通じて、ｍ－ビーンへのアクセスを与える。被管理オブジェクト・アダプタ・サーバは、管理アプリケーションが、ネットワーク管理システム・エージェント上で管理処理を実行することを可能にする。
【００５３】
ネットワーク管理システム・エージェントを管理可能とするためには、少なくとも１つの被管理オブジェクト・アダプタ・サーバを介してそれを接続する。しかし、ネットワーク管理システム・エージェントは、任意の数の被管理オブジェクト・アダプタ・サーバに接続可能であり、異なるプロトコルを用いるリモート管理アプリケーションがそれに問い合わせを行うことができる。
【００５４】
ネットワーク管理システムは、被管理オブジェクト・アダプタ・サーバを、標準的なプロトコルおよびプロプライエタリ(proprietary)なプロトコルに与える。
【００５５】
例えば、被管理オブジェクト・アダプタ・サーバを、１又は複数の標準的プロトコル、例えば、ＨＴＭＬ／ＨＴＴＰ、ＳＮＭＰ、およびＣＯＲＢＡのために与えることができる。
【００５６】
標準的なプロトコルのための被管理オブジェクト・アダプタ・サーバは、それらを用いる管理アプリケーションと直接通信する。
【００５７】
例えば、ＨＴＭＬ／ＨＴＴＰ被管理オブジェクト・アダプタ・サーバは、ユーザがウエブ・ブラウザを用いて、ｍ－ビーン内に含まれる管理情報を見たり、ネットワーク管理システム・エージェント上で管理処理を行うことを可能にする。
ＨＴＭＬ／ＨＴＴＰ被管理オブジェクト・アダプタ・サーバは、ｍ－ビーンから管理情報を得て、この管理情報を表すＨＴＭＬページを生成する。
【００５８】
ＳＮＭＰ被管理オブジェクト・アダプタ・サーバは、特定的に定義されたＳＮＭＰ管理情報ベース（ＭＩＢ：management information base）を用い、ＳＮＭＰマネージャがネットワーク管理システム・エージェント上で管理処理を行うことを可能にするように構成することができる。
【００５９】
また、ネットワーク管理システムは、被管理オブジェクト・アダプタ・サーバを、１又は複数の次のプロプラエタリなプロトコル、ＲＭＩ、Ｊａｖａ／ＨＴＴＰ、およびセキュア・ソケット・レイヤ（ＳＳＬ：Secure Sockets Layer）のために与えるように構成することも可能である。ネットワーク管理システムの一例では、被管理オブジェクト・アダプタ・クライアントは、Ｊａｖａ　ＡＰＩを提供する。したがって、被管理オブジェクト・アダプタ・クライアントを用いるどの管理アプリケーションも、Ｊａｖａ言語で書かれることになる。
【００６０】
ネットワーク管理システムを用いて開発されたエージェントは、異なる通信または管理プロトコルを用いて管理することができる。標準的な管理プロトコルを用いて管理するためには、ネットワーク管理システム・エージェントを、当該プロトコルのための被管理アダプタ・サーバに接続する必要がある。標準的なプロトコルのための被管理オブジェクト・アダプタ・サービスは、それを用いる管理アプリケーションと直接通信する。
【００６１】

図４のエージェントの表現を用いて、この構造の一例を図５に示す。図５では、エージェントには、ウエブ・ブラウザ４６によってアクセスする。ここでは、ＨＴＭＬアダプタによって、ビーンがＨＴＭＬページにマップされることが可能になる。ＨＴＭＬのようなプロトコルを用いることによって、クライアント・ステーション９０におけるブラウザ４６が、リモート・ステーション２０においてビーンをブラウズし、従来のウエブ・ブラウザ機能を用いてそれらにアクセスし、更に必要であれば修正することができる。図４に示す構造によれば、レポジトリ・サービス２７はフレームワーク２４に登録(register)されており、ＨＴＭＬアダプタ３４およびビーン（複数のビーン）２９はレポジトリ・サービスに登録されており、これによってビーン（複数のビーン）は、ＨＴＭＬアダプタ３４がサポートするＨＴＭＬページに有効に登録される。図５において、同様の参照番号は、図４に示した構成の同様の構造(feature)に関連する。

【００６２】

図６は、リモート・マシンにおいてビーンのプロパティの修正を可能にするステップを示すフロー・チャートである。ステップ９２において、ＨＴＭＬアダプタ３４が、リモート・ステーション２０における仮想マシン２２内で起動され、ステップ９４において、遠方からアクセス可能であるべきその仮想マシン２２のビーン（複数のビーン）２９を、フレームワーク、更に特定すれば、上述のようなレポジトリ・サービス２７に登録する。これは、ＨＴＭＬアダプタがフレームワーク２４に問い合わせする場合、フレームワーク２４は、レポジトリ・サービスを参照して、アクセスすべきビーン２９を識別し、ＨＴＭＬアダプタにそのビーン２９にアクセスさせることができるということを意味する。

【００６３】

ＨＴＭＬアダプタ３４は、従来のＨＴＴＰ交換を用いて、ネットワーク上での通信を可能にする。これは、ＨＴＴＰサーバとして振る舞う。要求を受信した場合、現在レポジトリ・オブジェクト２７に登録されているビーン（オブジェクト）２９のリストを含むページを動的に生成する。

【００６４】

ビーンは、ＨＴＭＬでは、ＨＴＭＬテーブルとして表され、その中では、
第１コラムがプロパティ名を含み、
第２コラムがプロパティ型を含み、
第３コラムがアクセス権（リード／ライト）を含み、
第４コラムがプロパティ値を含む。

【００６５】

前述のように、プロパティがリード／ライトの場合、ＨＴＭＬフォームが生成される。

【００６６】

ステップ９６では、前述のように、ＨＴＴＰ交換を用いてＨＴＭＬアダプタとの通信を行う従来のウエブ・ブラウザを用いてＨＴＭＬページにアクセスすることにより、ビーンのＨＴＭＬ表現を用いて、クライアント・ステーションにおいて（ディスプレイ９８によって表されている）ビーンを表示する。次に、ステップ１００において、ユーザは、従来のウエブ・ブラウザ機能を用いて、ビーンを選択することができる。次に、ウエブ・ブラウザは、ＨＴＴＰ　ＧＥＴ要求をＨＴＭＬアダプタ３４に発行する。ＨＴＭＬアダプタは、イントロスペクションを用いて、ビーン・プロパティを抽出し、ＨＴＭＬポスト応答(HTML post response)をブラウザに返し、これによって、ステップ１０２に示すように、ブラウザはプロパティを表示することができ、更に、ビーンにサポートされるアクションおよびイベントも表示することができる。従来のブラウザ機能を用いてブラウザを更に利用することにより、ユーザは、ビーンの態様に対する修正、例えば、プロパティに対する変更を選択し指定することができる。ＨＴＭＬ　ＧＥＴおよび／またはＳＥＴ要求およびＰＯＳＴ応答を更に交換することにより、ウエブ・ブラウザおよびＨＴＭＬアダプタは、ステップ１０４で、リモート・ステーションにおいてビーンの対応するプロパティを修正し、これらの変更をクライアント・ステーションにおいてユーザに表示することができる。

【００６７】

このように、この機構は、クライアントのマシンから、リモート・マシンにおけるビーンのようなオブジェクトに、リモート通信ネットワークを通じてアクセスするコンピュータによる実装方法を提供する。その際、リモート・マシンにおける外部からアクセス可能なマシン・ページに当該オブジェクトをマップし、ブラウザを用いてそのマシン・ページを介してオブジェクトをブラウズする。

【００６８】

図３に示す構造の別の例を図７に示す。ここでは、図３の表現を用い、ネットワーク管理システム・エージェント２０が、ＳＮＭＰマネージャ・アプリケーションによって管理される。図７において、同様の参照番号は、図３に示した構成の同様の構造に関連する。

【００６９】

図８に表すネットワーク管理システムの一例は、アダプタＡＰＩを備え、Ｊａｖａ管理アプリケーションが、ネットワーク管理システム・エージェントと通信することを可能にする。アダプタＡＰＩは、特定の通信プロトコルを通じて被管理オブジェクトにアクセスするために、被管理オブジェクト・アダプタ・クライアントを与える。ネットワーク管理システムは、Ｊａｖａ管理アプリケーションのためにｍ－ビーンの表現を定義し、ｍ－ビーンからそのような表現を自動的に生成するためのコンパイル・ツールを与える。名称サービスを供給し、Ｊａｖａ管理アプリケーションを、特定の通信プロトコルとは独立させる。

【００７０】

被管理オブジェクト・アダプタ・クライアントとは、Ｊａｖａ管理アプリケーションが被管理オブジェクトにアクセスできるようにする、抽象Ｊａｖａクラス(abstract Java class)である。Ｊａｖａ管理アプリケーションに対する被管理オブジェクトのプログラム的表現は、被管理オブジェクト・アダプタ・クライアントによって決定される。かかる機構は、同一の被管理オブジェクトを異なる表現で、異なるＪａｖａ管理アプリケーションに提示することを可能にする。ネットワーク管理システムは、１つ以上の次のプロトコル、ＲＭＩ、ＨＴＴＰ、およびＳＳＬを通じて被管理オブジェクトにアクセスするために、被管理オブジェクト・アダプタ・クライアントを与える。

【００７１】

被管理オブジェクト・アダプタ・クライアントは、アダプタ管理オブジェクト・インターフェース(adaptor managed object interface)の定義を与える。アダプタ管理オブジェクト・インターフェースは、Ｊａｖａ管理アプリケーションが、ネットワーク管理システム・エージェント上で、次の管理処理の１つ又は複数を実行することを可能にする。

【００７２】

－　ｍ－ビーンを検索すること、
－　リモートｍ－ビーンのプロパティを得るまたはセットすること、
－　リモートｍ－ビーンのメソッドをコールすること、
－　ｍ－ビーンのインスタンスを作成すること、
－　ｍ－ビーンを削除すること、および
－　リモートｍ－ビーンによって送出されたイベントを受信すること。

【００７３】

被管理オブジェクト・アダプタ・クライアントは、Ｊａｖａ管理アプリケーションに、リモート・エージェントにおける被管理オブジェクト上の「ハンドル」を与える。これらのハンドルは、Ｊａｖａ管理アプリケーションが、被管理オブジェクトを直接操作することを可能にする。Ｊａｖａ管理アプリケーションは、被管理オブジェクトによって用いられるプロトコルに関する情報を必要としない。Ｊａｖａ管理アプリケーションが必要とするのは、被管理オブジェクトが表すオブジェクトのクラスだけである。例えば、アカウントを処理するためのＪａｖａ管理アプリケーションは、アカウントを表すための抽象クラスを用いる。アカウントを操作するために、Ｊａｖａ管理アプリケーションは、被管理オブジェクト・アダプタ・クライアントから、アカウント被管理オブジェクト(account managed object)を得る。次に、アカウント被管理オブジェクトを、当該オブジェクトを表す抽

象クラスに組み込む(cast)。このようにして、アプリケーション・コードは、被管理オブジェクトがどのように実装されるかとは無関係となる。

【００７４】

図８は、クライアント・ビーン（ｃ－ビーン）５４とｍ－ビーン２８との間の関係の概略図である。ｃ－ビーン５４は、Ｊａｖａ管理アプリケーションに対する、ｍ－ビーンの表現である。本発明の好適実施例では、ｃ－ビーン５４は、ｍ－ビーン２８と同様に、ＪａｖａＢｅａｎｓコンポーネントとして実装される。ｃ－ビーン５４は、Ｊａｖａ管理アプリケーションがどのようにしてｍ－ビーン２８にアクセスするのかを定義する。

【００７５】

図８に見られるように、ｃ－ビーン５４は、ｍ－ビーンのメソッドの内どれがＪａｖａ管理アプリケーションにアクセス可能かを定義する被管理オブジェクト・インターフェース（ＭＯインターフェース）５６と、ＭＯインターフェース５６において定義されたメソッドを実装する被管理オブジェクト・スタブ（ＭＯスタブ）５８とから成る。

【００７６】

Ｊａｖａ管理アプリケーションは、アダプタＭＯインターフェース６０を用いることによって、ｃ－ビーンへの参照を得る。アダプタＭＯインターフェースは、ｃ－ビーン５４をインスタンス化する。ｃ－ビーン５４の同じインプリメンテーションが、アダプタＭＯインターフェース６０を実装する任意の被管理オブジェクト・アダプタ・クライアント上で実行することができる。同じ被管理オブジェクトの異なるインプリメンテーションを、異なるＪａｖａ管理アプリケーションに提示することができる。このようにして、単一のｍ－ビーンを、いくつかのｃ－ビーン５４と関連付けることができる。

【００７７】

Ｊａｖａ管理アプリケーションは、それに関連するｃ－ビーン５４のメソッドをコールすることにより、ｍ－ビーン上で管理処理を実行する。Ｊａｖａ管理アプリケーションから見ると、ｃ－ビーン５４は、リモートＪａｖａオブジェクト（ｍ－ビーン２８）のローカル表現となる。

【００７８】

アダプタＭＯインターフェース６０は、ｃ－ビーン５４をインスタンス化する。ｃ－ビーンの同じインプリメンテーションは、アダプタＭＯインターフェース６０を実装する、任意の被管理オブジェクト・アダプタ・クライアント６２上で実行することができる。同じ被管理オブジェクトの異なる表現を、異なるＪａｖａ管理アプリケーションに提示することができる。このようにして、単一のｍ－ビーンをいくつかのｃ－ビーンと関連付けることができる。

【００７９】

Ｊａｖａ管理アプリケーションは、それに関連するｃ－ビーンのメソッドをコールすることによって、ｍ－ビーン上で管理処理を実行する。Ｊａｖａ管理アプリケーションから見ると、ｃ－ビーンはリモートＪａｖａオブジェクト（ｍ－ビーン）のローカル表現となる。

【００８０】

図９は、ｍ－ビーンからのｃ－ビーン５４の生成を表す概略図である。本発明の一実施例では、ｃ－ビーンは、リフレクションＡＰＩを用いて、ｍ－ビーン２８から自動的に生成される。生成されたｃ－ビーンは、ｍ－ビーンと同じプロパティ、メソッド、およびイベントを有する(exhibit)。これは、例えば、アクセス制御ポリシー(access control policy)が執行されない場合である。

【００８１】

ｍ－ビーン２８からｃ－ビーン５４を自動的に生成するために、コンパイラ６０が、ｍ－ビーン２８を入力として取り込み、ｃ－ビーン５４のＭＯインターフェースおよびＭＯスタブを出力として生成する。例えば、accountと称するＪａｖａクラスを表すｍ－ビーン２８をコンパイルする場合、コンパイラ６０は、accmountMOと称するＭＯインターフェース５６およびaccountMOSTUB５８と称するＪａｖａクラスを生成する。accountMOSTUB５８

は、accoutMOインターフェース５６を実装する。

【００８２】

Ｊａｖａ管理アプリケーションは、ＭＯインターフェースの定義を用いて、開発することができる。異なるスタブをロードすることによって、adaptorMOインターフェースは、実行時に、Ｊａｖａ管理アプリケーションのビヘービャを修正することができる。例えば、リード・オンリ・スタブをロードする場合、Ｊａｖａ管理アプリケーションは、ｍ－ビーン２８内のプロパティを修正することはできない。

【００８３】

コンパイラ６０は、リード・オンリ・スタブまたはリード／ライト・スタブを生成することができる。生成されたスタブは、adaptorMOインターフェースを利用する。したがって、それらのビヘービャは、被管理オブジェクト・アダプタ・クライアントのインプリメンテーションとは無関係に、adaptorMOインターフェースを実装する任意の被管理オブジェクト・アダプタ・クライアント上においても同一である。

【００８４】

図１０は、図８に示す構造を用いて、ローカル管理（クライアント）ステーションから、リモート（サーバ）ステーションにおけるビーンにアクセスするためのステップを示すブロック図である。

【００８５】

管理ステーションにおける管理アプリケーション（例えば、Ｊａｖａ管理アプリケーション）は、ステップ８０において、クライアント・ステーションにおけるadaptorMOへの獲得要求(get request)を生成する。ステップ８１において、adaptorMOは、クライアント・ステーションにおけるＭＯスタブおよびネットワーク・アダプタを用いて、適切なネットワーク・プロトコル（例えば、ＨＴＴＰ）にしたがって要求を生成する。

【００８６】

このようにネットワークを通じて被管理システムに送られる要求は、例えば、被管理オブジェクトの管理プロパティのためのＧＥＴ要求の形態とすることができる。

【００８７】

適切な被管理オブジェクト・アダプタ・サーバ３０～３８が、用いられるプロトコルに従って、ステップ８２において、外部要求を受信する。次に、ステップ８３において、フレームワーク２４にアクセスし、適切なメソッドを得る。フレームワークは、ステップ８３において、当該要求に対する被管理オブジェクト・メソッドを獲得し、ステップ８４において、その被管理オブジェクトの管理プロパティをアダプタに返す。一方、アダプタは、ステップ８５において、同じネットワーク・プロトコル（例えば、ＨＴＴＰ）にしたがって、リターン・メッセージをその結果と合成する。結果のメッセージは、ステップ８６において、クライアント・アダプタおよびadaptorMOにおいて受信され、その結果を管理アプリケーションに返す。

【００８８】

管理アプリケーションを特定の通信プロトコルから独立させる名称サービスが提供される。Ｊａｖａ管理アプリケーションは、この名称サービスを用いて、エージェントにアクセスするためにどの被管理オブジェクト・アダプタ・クライアントを用いるべきかを決定する。図１１は、名称サービスの処理を示す概略フロー・チャートである。

【００８９】

ステップ６２において、管理アプリケーションは、アクセスすべきエージェントの識別子(identity)を主サービスに渡す。

【００９０】

ステップ６４において、名称サーバは、被管理オブジェクト・アダプタ・クライアントのクラス名、例えば、Ｊａｖａ　ＲＭＩシステムによるエージェントのアクセスの場合であれば、sunw.jaw.moa.rmiを返す。

【００９１】

ステップ６６において、Ｊａｖａ管理アプリケーションは、そのデフォルトのクラス・ロ

ーダを用いて、被管理オブジェクト・アダプタ・クライアントを動的にロードし、それをインスタンス化する。Ｊａｖａ管理アプリケーションは、次に、通信プロトコルには無関係に、被管理オブジェクト・アダプタ・クライアントを通じて、エージェントと相互処理を行う。

【００９２】

前述のように、管理ビーン即ちｍ－ビーンは、ネットワーク管理システム・エージェントにおける被管理オブジェクトである。被管理オブジェクトは、当該エージェントによって制御および監視されるリソースのソフトウエア抽象化(software abstraction)である。ネットワーク管理システムの例では、全てのｍ－ビーンは、ＪａｖａＢｅａｎｓコンポーネントとして実装される。これらには、Ｊａｖａ　Ｂｅａｎｓ開発キットのような、従来のＪａｖａアプリケーション・ビルダを用いてアクセスすることができる。ｍ－ビーン内では、ネットワーク管理システムによって提供されるサービスのコールおよび使用が可能である。

【００９３】

ＪａｖａＢｅａｎｓコンポーネントは、当該ＪａｖａＢｅａｎｓコンポーネントの外観またはビヘービャに影響を与え得る個別の名称付き属性(named attribute)を形成するプロパティを含む。例えば、イーサネット・ドライバを表すｍ－ビーンは、入来するパケット数を表すIpacketsと称するプロパティを有することができる。プロパティは任意の値を有することができ、ビルトインＪａｖａエンド型およびjava.awt.colorのようなクラスまたはインターフェース型の双方を含む。プロパティは、常に、それらを所有するオブジェクト上でのメソッド・コールを通じてアクセスされる。読み取り可能なプロパティでは、プロパティ値を読み取るためのgetメソッドがある。書込可能なプロパティでは、プロパティ値を更新可能にするsetメソッドがある。

【００９４】

プロパティを位置付けるために、次のようなデフォルトの設計パターンを使用することができる。

【００９５】

public PropertyType get PropertyName();
public void set PropertyName (PropertyType値);
クラス定義が、セッタ(setter)のパラメータ型に対応するゲッタ(getter)のリターン型と一致する１対のget PropertyNameおよびset PropertyNameメソッドを含む場合、これらのメソッドは、リード／ライト・プロパティを定義する。クラス定義がこれらのメソッドの一方のみを含む場合、その名称は、PropertyNameと呼ばれるリード・オンリまたはライト・オンリ・プロパティのいずれかを定義する。

【００９６】

ブーリアン・プロパティ(Boolean property)に加えて、次の設計パターンを用いてgetメソッドを定義することが可能である。

【００９７】

public boolean isPropertyName()
このPropertyNameメソッドは、getPropertyNameメソッドの代わりに提供することもでき、その場合、getPropertyNameメソッドに加えて備えてもよい。

【００９８】

インデックス・プロパティとは、以下の形態のメソッドによってアクセスされるアレイPropertyElement[]である。

【００９９】

public PropertyElement getPropertyName (int index);
public void setPropertyName (int index, PropertyElementb)
クラス定義がいずれかの種類のメソッドを含む場合、PropertyNameはインデックス・プロパティとなる。これらのメソッドは、プロパティ値を読み取るためおよび書き込むために用いることができる。これらのメソッドは、単純なプロパティのために定義されるメソッ

ドに加えて定義することができる。したがって、インデックス・プロパティは、４つのアクセサ・メソッド(accessor method)によって表すことができる。

【０１００】

デフォルトでは、次の設計パターンを用いて、どのイベントをｍ－ビームがマルチキャスト(multicast)することができるのかについて判定を行なう。

【０１０１】

public void addEventListenerType(EventListenerType a);
public removeEventListenerType(EventListenerType a);
これらのメソッドは双方共、EventListenerType 型のアーギュメントを取り、EventlistenerType型はjava.util.EventListenerインターフェースに継承し、最初のメソッドはaddから開始し、２番目のメソッドはremoveから開始し、EventListenerType型名は、Listenerで終わる。

【０１０２】

この設計パターンは、ＪａｖａビーンがEventListenerType インターフェースにおいて指定されたイベントに対して、マルチキャスト・イベント・ソースとして動作することを想定している。

【０１０３】

ＪａｖａＢｅａｎｓモデルに従うために、ＪａｖａＢｅａｎｓコンポーネントの全てのパブリック・メソッドは、コンポーネントによるアクセスのためにコンポーネント環境内では外部メソッドとして露出させなければならない。デフォルトでは、これはプロパティ・アクセサ・メソッドを、そして最終的にリスナ・レジストリ・メソッド(listener registry method)を含む。

【０１０４】

設計パターン・エレメントのためのＪａｖａＢｅａｎｓコンポーネント・モデルに加えて、ネットワーク管理システムは、アクションを、遠方からコールする意味のあるｍ－ビーンのパブリック・メソッドとして定義することができる。アクションは、他のローカルｍ－ビーンに対して露出されるｍ－ビーンのパブリック・メソッドの、遠方からコールすることができるパブリック・メソッドからの差別化を容易にする。アクションのための設計パターンは次の通りである。

【０１０５】

public AnyJavaType performAnAction (AnySignature);
ｍ－ビーンは、ネーティブなライブラリを含むことができる。これは、ｍ－ビーンの言語（例えば、Ｊａｖａ）で書かれていないライブラリである。ネットワーク管理システムは、ｍ－ビーンと同じリモート・クラス・サーバからネーティブなライブラリをロードする機構を提供することができる。ｍ－ビーンがこの機構を使用することを可能とするために、フレームワーク・クラスの静的なloadLibraryメソッドをコールすることができ、その際、コール元が、当該コール元のＪａｖａクラスへの参照を含む。かかる情報は、フレームワーク２４が、クラスをロードするために用いるクラス・ローダを識別するために用いられる。

【０１０６】

上述のコア管理サービスは、全てのエージェントに共通である多くの管理処理のためのものであり、これによってエージェントの開発を簡略化する。これらのコア・サービスを提供することにより、ネットワーク管理システムは、特定のアプリケーションに特有のエージェントの部品、即ち、ｍ－ビーンおよびそれらを制御し管理するアプリケーションの開発に、努力を集中させることができる。

【０１０７】

図１２は、ネットワーク管理システム・エージェント２０の初期化を示す概略フロー・チャートである。この初期化プロセスは、以下のステップを含む。

【０１０８】

－　ステップ７０において、フレームワーク２４のインスタンスを作成する。

【０１０９】
－　ステップ７２において、ｍ－ビーン・レポジトリ・サービス２７を追加する。
【０１１０】
－　ステップ７４において、メタデータ・サービス２９を追加する。
【０１１１】
－　ステップ７６において、少なくとも１つの被管理オブジェクト・アダプタ・サーバ（３０～３８）を追加し、管理アプリケーションがエージェントにアクセスできるようにする。
【０１１２】
ネットワーク管理システム・エージェント２０がいったん初期化されると、エージェントを開始する前に、それ以上の管理サービスを追加する必要はない。これらは、エージェントの実行中に、動的にそれに追加することができる。
【０１１３】
フレームワーク２４は、ネットワーク管理システムを用いて開発したエージェント２０の管理サービスおよびｍ－ビーンを制御する。この好適実施例では、これは、Ｊａｖａクラスjava.jaw.agent.cmf.Frameworkによって実装される。エージェントは、フレームワークのインスタンスを１つ含まなければならない。即ち、好適実施例では、java.jaw.agent.cmf.Frameworkクラスの１つのインスタンスを含まなければならない。
【０１１４】
この好適実施例では、ｍ－ビーンは、エージェント２０のｍ－ビーン・レポジトリ２７にオブジェクト名を登録した場合にのみ管理することができる。したがって、エージェント２０が動作状態となる前に、ステップ７２においてｍ－ビーン・レポジトリ・サービス２７を追加する。ｍ－ビーン・レポジトリ・サービス２７は、ｍ－ビーンとそのオブジェクト名との間の関連(association)を格納し検索するために用いられる。この好適実施例のｍ－ビーン・レポジトリ・サービスは、Ｊａｖａインターフェースjava.jaw.agent.services.MoRepSrvIfとして定義される。エージェントは、常に一度に１つのｍ－ビーン・レポジトリ・サービスとのみ関連付けることができる。しかし、エージェントが動作状態にある間には、当該エージェントが関連付けられているｍ－ビーン・レポジトリ・サービスを変更することは可能である。
【０１１５】
ｍ－ビーン・レポジトリは、揮発性記憶装置または永続的記憶装置として実現することができる。揮発性レポジトリでは、ｍ－ビーンに関する全ての情報をメモリに格納する。揮発性レポジトリ内の全ての情報は、エージェントを停止したときに失われる。したがって、揮発性レポジトリを用いてエージェントを開始する場合には、情報をレポジトリに再ロードしなければならない。永続的レポジトリでは、ｍ－ビーンに関する情報の全てはデータベースに格納され、それによって、エージェントを停止した場合でも、永続的レポジトリ内の情報は失われない。また、混合レポジトリを実施することによって、ｍ－ビーンに関する情報をメモリまたはデータベースに格納することも可能である。
【０１１６】
先に引用したメタデータ・サービスを用いて、ｍ－ビーンがによってサポートされるプロパティおよびアクションを得る。メタデータ・サービスの好適な実施形態は、イントロスペクションを実行する公知のＪａｖａ開発キットによって与えられるリフレクションＡＰＩに基づく。
【０１１７】
前述のように、少なくとも１つの被管理オブジェクト・アダプタ・サービスが、ネットワーク管理システム・エージェントとして、サーバの仮想マシンにおいて実行していなければならない。ネットワーク管理システムは、特定のインターフェース定義またはインプリメンテーションに従うために、被管理オブジェクト・アダプタ・サーバを必要としない。被管理オブジェクト・アダプタ・サーバは、フレームワーク２４にアクセスして、エージェント内に含まれる情報を検索し変更するように構成されている。与えられる被管理オブ

ジェクト・アダプタ・サーバは、ｍ－ビーンとして実装される。被管理オブジェクト・アダプタ・サーバの例については、先に述べた。本発明のこの好適実施例では、被管理オブジェクト・アダプタ・サーバは、適切なＪａｖａクラスとして実装される。

【０１１８】

例えば、ＲＭＩ被管理オブジェクト・アダプタ・サーバは、ＪａｖａクラスSunw.jaw.agent.adaptor.rmi.AdaptorServerImplとして実装することができる。これは、Ｊａｖａ管理アプリケーションが、Ｊａｖａリモート・メソッド呼び出し（ＲＭＩ：remote method invocation）システムを用いてエージェントにアクセスすることを可能にする。上述のように、Ｊａｖａ管理アプリケーションは、ＪａｖａクラスSunw.jaw.agent.adaptor.rmi.AdaptorClientとして実装された被管理オブジェクト・アダプタ・クライアントを通じて、このサーバにアクセスする。

【０１１９】

前述のようにエージェントがいったん初期化されると、そのエージェントの実行中に、エージェントにコア管理サーバを追加することができる。次の方法のいずれかで、コア・サービスをエージェントに追加することができる。

【０１２０】

－　フレームワーク・クラス内のサービスに対して直接setメソッドをコールする。

【０１２１】

－　ｍ－ビーンに対する方法と同じ方法で、ｍ－ビーン・レポジトリにサービスを追加する。

【０１２２】

コア管理サービスを直接追加すると、コア管理サービスをｍ－ビーン・レポジトリに追加する場合よりも、動作(performance)の高速化が図れる。その理由は、フレームワークが、サービスを得るためにｍ－ビーン・レポジトリに問い合わせする必要がないからである。しかし、直接追加されたコア管理サービスには、次のようないくつかの制約が適用される場合がある。

【０１２３】

－　サービスは、リモート・アプリケーションには可視的でない。

【０１２４】

－　サービスに関する情報を、永続的記憶装置に格納することは不可能である。

【０１２５】

したがって、コア管理サービスがリモート・アプリケーションから可視的であるのが望ましい場合には、サービスをｍ－ビーン・レポジトリに追加する必要がある。コア管理サービスに関する情報を永続的記憶装置に格納することが望まれる場合も、サービスをｍ－ビーン・レポジトリに追加する必要がある。サービスが追加されるｍ－ビーン・レポジトリは、永続的格納をサポートしていなければならない。

【０１２６】

クラス・サービス名は、エージェントのために実装されているサービスをフレームワーク２４が識別する名称を含む。フレームワーク２４は、以下のようにして、必要なサービスを検索する。

【０１２７】

１．フレームワーク２４は、サービスが直接setメソッドを用いて定義されたのか否かについてチェックを行なう。サービスがこのように定義された場合、フレームワーク２４はこのサービスを用いる。

【０１２８】

２．サービスが、直接setメソッドを用いて定義されたのではない場合、フレームワークは、ｍ－ビーン・レポジトリに問い合わせを行い、当該サービスを実装するクラスのインスタンスである全てのｍ－ビーンを得る。例えば、メタデータ・サービスでは、フレームワーク２４は、ｍ－ビーン・レポジトリに問い合わせを行い、ServiceName.METAと称するクラスのインスタンス全てを得る。

【０１２９】
－　レポジトリが、そのクラスのインスタンスをいくつか含む場合、フレームワーク２４は、ｍ－ビーン・レポジトリによって返された最初のインスタンスを用いる。
【０１３０】
－　ｍ－ビーン・レポジトリがそのクラスのインスタンスを全く含まない場合、フレームワークは、ServiceNotFound例外（Exception）を投げる。
【０１３１】
ネットワーク管理サービス・エージェント内では、種々の処理を行なうことができる。例えば、あるエージェント内のオブジェクトは、コア管理サービスを用いて、
－　ｍ－ビーンをインスタンス化する、
－　ｍ－ビーンをｍ－ビーンに登録する、
－　ｍ－ビーン・レポジトリからｍ－ビーンを検索する、
－　ｍ－ビーン内のプロパティの値を獲得し、セットする、そして、
－　ｍ－ビーン間の関係を定義する。
【０１３２】
オブジェクト名は、ｍ－ビーンを一意的に識別する。管理アプリケーションは、オブジェクト名を用いて、管理処理を実行するｍ－ビーンを識別する。いずれの命名方式を使用することも可能である。例えば、本発明の好適実施例では、Microsoft Corporation（マイクロソフト社）がハイパー・メディア管理方式（ＨＭＭＳ：Hyper Media Management Scheme)のために定義した命名方式を用いることができる。
【０１３３】
ｍ－ビーンをインスタンス化するために、フレームワーク・クラスの以下のメソッドの１つをコールすることができる。
【０１３４】
ｍ－ビーンを格納するためのユーザのデフォルト格納機構に対するnewObject、
ｍ－ビーンが永続的であることを指定するnewDBObject。
【０１３５】
これらのメソッドのいずれかを用いる場合には、次のものを提供することが必要である。
【０１３６】
－　インスタンス化するｍ－ビーンのＪａｖａクラス、および
－　ｍ－ビーンを登録するために用いられるオブジェクト名。
【０１３７】
デフォルトでは、フレームワーク２４は、デフォルト・クラス・ローダを用いて、作成すべきｍ－ビーンのＪａｖａクラスを配置する。次に、そのクラスのインスタンスを作成する。いったんｍ－ビーンをインスタンス化すれば、それを初期化し、登録して、フレームワーク２４にアクセス可能となるようにする。
【０１３８】
－　ｍ－ビーン自体に定義されているメソッド、または
－　フレームワーク２４、
を用いることによって、ｍ－ビーンの初期化および登録を行なうことができる。
【０１３９】
ｍ－ビーン自体に定義されているメソッドを用いてｍ－ビーンのレジスタを初期化するためには、当該ｍ－ビーンのＪａｖａクラス定義は、以下のものを含まなければならない。
【０１４０】
－　初期化メソッド、
－　ｍ－ビーン自体をｍ－ビーン・レポジトリに登録するために必要なコード。
【０１４１】
いったんｍ－ビーンがインスタンス化されたならば、フレームワーク２４はメタデータ・サービス２７を用いて、新たに作成されたｍ－ビーン内において初期化メソッドを見つけ出す。かかるメソッドがｍ－ビーン内に存在する場合、フレームワーク２４は、

－　第１のパラメータとして、それ自体に対する参照、
－　第２のパラメータとして、ｍ－ビーンの登録に用いるためのオブジェクト名、
を与えて、メソッドをコールする。
【０１４２】
したがって、ｍ－ビーンは、与えられたコードを用いて、それ自体をｍ－ビーン・レポジトリに登録することができる。
【０１４３】
ｍ－ビーンに初期化メソッドが与えられていない場合、フレームワークは、この目的のために与えられた機能を用いて、そのｍ－ビーンを初期化し、登録する。
【０１４４】
ＪａｖａＢｅａｎｓコンポーネントをｍ－ビーン・レポジトリ２５に登録することによって、そのコンポーネントをエージェント２０によって管理することが可能となる。ＪａｖａＢｅａｎｓコンポーネントを登録する際、当該ＪａｖａＢｅａｎｓコンポーネント自体におけるコードの変更を必要としない。代わりに、必要なのは、ｍ－ビーン・レポジトリ内にそれを登録するためのコードの追加だけである。したがって、あらゆる既存のＪａｖａＢｅａｎｓコンポーネントを、ｍ－ビーン・レポジトリ内に登録することができる。いったん登録されたならば、エージェント２０は、あらゆるｍ－ビーンと同様に、ＪａｖａＢｅａｎｓコンポーネントを管理する。ｍ－ビーンを登録する際、これにはオブジェクト名が割り当てられる。オブジェクト名は、明示的に特定することができる。オブジェクト名が明示的に特定されていない場合には、フレームワーク２４は、デフォルトの名称を当該ｍ－ビーンに割り当てる。
【０１４５】
ネットワーク管理システムは、ｍ－ビーン・レポジトリからｍ－ビーンを検索するためのサービスを与える。これらのサービスは、
－　オブジェクト名に対するパターン照合（matching）、または
－　それらが含むＪａｖａプロパティに関するクエリ（フィルタ）
を用いて、ｍ－ビーンの検索を可能とする。
【０１４６】
ｍ－ビーンのオブジェクト名のパターン照合を用いることによって、
－　そのオブジェクト名全体を用いる特定のｍ－ビーン、
－　当該オブジェクト名で表されるのと同じ論理クラスを共有する１組のｍ－ビーン、
－　同じドメイン名を共有する１組のｍ－ビーン、または
－　エージェントに含まれる全てのｍ－ビーン、
を検索することができる。
【０１４７】
クエリ（queries）を用いると、Ｊａｖａプロパティおよびｍ－ビーン内におけるそれらの値にしたがって、ｍ－ビーンの検索が可能となる。ｍ－ビーン・レポジトリは、プロパティを評価することが可能な場合、それを行なう。さもなければ、フレームワーク自体がクエリを評価する。レポジトリがクエリにアクセス可能か否かについて判定を行なうために、フレームワークは、この目的のために、クエリ・メソッドを生成する。
【０１４８】
ネットワーク管理システムは、ｍ－ビーンのプロパティを得るためおよびセットするためにサービスを与える。エージェントがメタデータ・サービスを与える場合、ｍ－ビーンのgetメソッドまたはsetメソッドに対するコールにおいて供給する必要があるのは、以下のもので全てである。
【０１４９】
－　検索するまたはセットすべきプロパティの名称、
－　当該プロパティを含むｍ－ビーンのオブジェクト名。
【０１５０】
エージェントがメタデータ・サービスを与えない場合、ｍ－ビーンのgetメソッドまたはs

etメソッドの直接コールすることが依然として可能である。この場合も、コールするメソッドの名称および署名(signature)を、コール元に供給する必要がある。

【０１５１】

関係サービスは、ｍ－ビーンが要求されたときに、それらの間の関係を定義することを可能にする。関係は、前もって定義しておく必要はない。ｍ－ビーン間の関係に関する情報は、ｍ－ビーン自体に格納するのではなく、関係に格納する。関係は、ｍ－ビーン・レポジトリに格納する必要がある。関係は、

－　１組の役割(role)、（例えば、所有関係では、人が本を所有し、本は人に所有される）

－　関係において、要求された役割の数に対応する度合い（degree）、

によって定義される。

【０１５２】

ある関係に関与するｍ－ビーンは、それらのオブジェクト名によって、関係内において参照される。起動時、またはエージェントの実行中に、特定のクラス・ローダをエージェントに追加することができる。ｍ－ビーン・レポジトリに登録されているのであれば、同一エージェント内にいくつかのクラス・ローダを有することが可能である。新たなオブジェクトの作成が要求された場合、オブジェクトをロードするために用いられるクラス・ローダを指定することができる。この場合、クラス・ローダは、そのオブジェクト名によって識別される。システムは、ネットワーク・クラス・ローダのいくつかのインプリメンテーションを与え、各インプリメンテーションは、異なるプロトコルを用い、異なるクラス・サーバを必要とする。

【０１５３】

このシステムは、異なる通信プロトコルを通じてイベントのフィルタリング、ロギング(logging)、および送出(forwarding)を行なうために、イベント・ハンドリング・サービスを与える。イベントを放出するために、センド・イベント・メソッド(send event method)をコールする。このメソッドがコールされると、フレームワーク２４は、イベント・ハンドリング・サービスに対応する全てのｍ－ビーンを検索し、検索された各イベント・ハンドラのイベント・ハンドリング・メソッドをコールする。このメソッドは、イベントを扱う役割を担う。

【０１５４】

図９は、ｍ－ビーン２８からのｃ－ビーン５４のコンパイルを表す概略図である。コンパイラ６０は、ｃ－ビーンを生成するための１つの変換方式を実装する。しかしながら、要件に応じて、異なる変換方式を実装することも可能である。

【０１５５】

図１３は、Ａと称する単一のｍ－ビーンをコンパイルするコンパイラ６０の出力の一例を示す。図１４は、コンパイルされたｍ－ビーンが、特定のイベントAnEventに対するaListenerを含む場合の、コンパイラ６０の出力を示す。

【０１５６】

したがって、ｍ－ビーンがリスナ(listener)を含む場合、コンパイラ６０は、－　ＭＯインターフェース内に含まれるリスナに対するＪａｖａインターフェース、

－　ｍ－ビーン・イベントを捕獲し、それらをフレームワーク２４に送出するためのｍ－ビーン・リスナのインプリメンテーションであるリスナ・スタブ、および

－　EventMOと呼ばれる、リスナに関連するイベントのＪａｖａ管理アプリケーションのビュー(view)、を生成する。

【０１５７】

コンパイラ６０は、適用可能な設計パラメータを用いて、ｍ－ビーンを解析（パーシング）する。パーシングの後、コンパイラ６０は、多数の規則を用いてｃ－ビーンを生成する。

【０１５８】

ｍ－ビーンの各プロパティは、同じアクセサ・メソッドを有するｃ－ビーン内に存在する

。したがって、あるプロパティがｍ－ビーンにおいてリード・オンリである場合、そのプロパティはｃ－ビーンにおいてもリード・オンリである。

【０１５９】

図１５のＡは、図１５のＢに示すコード例によって定義されたｍ－ビーンをコンパイルした場合に生成される単純なｃ－ビーンの図である。加えて、コンパイラ６０は、単純なＭＯインターフェースのインプリメンテーションを含むファイルを生成する。

【０１６０】

プロパティ・アクセサに加えて、コンパイラ６０は、リモート・アクセスを与えることが意味のあるパブリック・メソッドに対してのみ、コードを生成する。他のパブリック・メソッドは、コンパイラ６０によって生成されるｃ－ビーンには現れない。

【０１６１】

図１６のＡは、図１６のＢにおけるように定義されたｍ－ビーンにおいてアクション・メソッドをコンパイルした場合にコンパイラ６０が生成する、ＭＯインターフェースのｃ－ビーンにおけるアクション・メソッドのサブセットを示す。

【０１６２】

ｍ－ビーンが、Ａと称するリスナを含む場合、コンパイラ６０は、ｃ－ビーン内にAIfMOと称するリスナを含む。

【０１６３】

ｃ－ビーンを用いる場合、アプリケーションは、Aifmoインターフェースを実装し、ｃ－ビーン内のリスナを追加または除去しなければならない。通常、リスナは、所定数のメソッドを含むインターフェースである。各メソッドは、１つの入力パラメータを有する。入力パラメータは、関係するイベント・オブジェクトに継承する。

【０１６４】

一例では、リスナＡ内に定義された各メソッドは、この例の目的のために、AnEventと呼ばれるイベント・オブジェクトを参照する。したがって、Aifmoインターフェースにおいては、このイベント・オブジェクトはAnEventMOと呼ばれる。このように、リスナＡに対して、コンパイラ６０は、

－　Aifmo.java、

－　AneventMO.java、というファイルを生成する。

【０１６５】

加えて、コンパイラ６０は、Astub.javaと称するリスナＡのインプリメンテーションを生成する。

【０１６６】

コンパイラ６０によって生成されたコードは、ＪａｖａＢｅａｎｓコンポーネント・モデルによって設計された設計パラメータを用いてコンパイルする。このため、コンパイラ６０によって生成されたオブジェクトはこのモデルに従う開発環境に統合することができる。加えて、コンパイラ６０は、ＪａｖａＢｅａｎｓコンポーネント・モデルによって定義された設計パターンに従わない、いくつかのパブリック・メソッドを追加する。追加されたメソッドは、ｍ－ビーンとｃ－ビーンとの間のネットワーク・トラフィックを制限するように設計される。例えば、ｃ－ビーン上である関数をコールすることにより、対応するｍ－ビーンの全てのプロパティを読み出すことができる。

【０１６７】

コンパイラ６０は、Ｊａｖａソース・コードを生成する。生成したコードを編集し、それを修正して、ｍ－ビーンの特定のビューを定義することができる。インターフェースおよびスタブの双方を修正する代わりに、インターフェースを保持し、スタブのみを修正する方がよい。コンパイラ６０によって生成されたコードは、インターフェースを用いてアプリケーションを構築することを可能にする。一度に、adaptorMOによってどのスタブがロードされたかに応じて、ビヘービャは変化する。例えば、コンパイラは、同じインターフェースに、リード・オンリ・スタブまたはリード－ライト・スタブを生成することができる。したがって、ｍ－ビーン・ブラウザは、インターフェースに基づいて開発することが

できる。前述のように、ブラウザは、これによって、リード・オンリ・スタブまたはリード－ライト・スタブのどちらがロードされたかに応じて、異なるビヘービャを有することになる。

【０１６８】

adaptorMOインターフェースは、エージェント２０を管理するために定義されたＪａｖａインターフェースである。ネットワーク管理システムは、前述のように、異なる通信プロトコルに基づいて、adaptorMOインターフェースのいくつかのインプリメンテーションを与える。しかしながら、adaptorMOインターフェースは、プロトコルには独立している。したがって、インターフェースを用いて書かれたコードのいずれの部分でも、ネットワーク管理システムによって与えられたインプリメンテーションのいずれにおいても実行することができる。

【０１６９】

adaptorMOインターフェース内には、２つの異なるレベルがある。リモート・オブジェクト（ｍ－ビーン）をそれらの名称を用いて操作する低レベル、およびリモート・オブジェクト（ｍ－ビーン）を当該リモート・オブジェクトのローカル・ビュー（ｃ－ビーン）を用いて操作する高レベルである。高レベルは、低レベル・インターフェースの上に構築する。

【０１７０】

低レベル・インターフェースを用いることは、汎用的なアプリケーション（ＨＴＭＬオブジェクト・ビューアまたはＭＩＢブラウザのような）を構築するためには実用的である。高レベルのインターフェースを用いることは、それより低いレベルのインターフェースを用いるよりも、かなり容易である。しかしながら、アプリケーションが操作するｃ－ビーンのセマンティック(semantic)を、当該アプリケーションが知っていることを前提とする。加えて、アプリケーション（またはadaptorMOインターフェース）が、ＭＯおよびＭＯスタブに対するアクセスを有することを要求する。アダプタを初期化する第１ステップは、adaptorMOインターフェースのインプリメンテーションを初期化することから成る。

【０１７１】

アダプタを初期化するために、クライアント・アプリケーションは、adaptorMOインターフェース内に定義されている、"connect"メソッドを呼び出す。エージェントのホスト名、使用するポート番号、および通常基本的な通信機構に依存する論理名に関連するパラメータを与える。使用するアダプタのインプリメンテーション名を与えるのと同時に、名称サーバまたはディレクトリ・サービスから、情報の異なる部分を得ることができる。adaptorMOインターフェースが用いる、基本的な通信機構に応じて、"connect"へのコールは、クライアントとエージェントとの間でメッセージ交換を全く伴わない場合もある。

【０１７２】

アダプタは、

－　特定のｍ－ビーン（そのオブジェクト名によって判明する）を表すために用いる、Ｊａｖａクラス名を得るための名称サーバ、

－　ｃ－ビーンをロードするためのクラス・ローダ、

を利用する。

【０１７３】

クライアント・アプリケーションが実行中のマシン上に存在する場合、ｃ－ビーンに対するＪａｖａクラス全てが特定のクラス・ローダを使用する必要はない。代わりに、ネットワークを通じてクラスを得るためのネットワーク・クラス・ローダを用いることができる。

【０１７４】

ネットワーク・クラス・ローダを用いるために、クライアント・アプリケーションは、ネットワーク・クラス・ローダをインスタンス化する必要がある。オブジェクトをインスタンス化する場合、アプリケーションはオブジェクト名を与える。オブジェクト名は、いずれかのドメインおよびいずれかのクラス名を含むことができる。

【０１７５】
しかしながら、オブジェクト名は、以下のキー・プロパティを含まなければならない。
【０１７６】
－　ホスト（クラス・サーバが実行中のホストの名称）、
－　ポート（使用するポート番号）、
－　サービス（呼び出すべきＲＭＩサービスの名称）。
【０１７７】
アダプタがいったん初期化されると、アプリケーションは、リモート・エージェント上で管理処理を実行する準備ができたことになる。アプリケーションは、エージェントによって管理されているｍ－ビーンのサブセットまたは全てを検索することができる。オブジェクトを検索する場合、アプリケーションはクエリを指定することができる。検索の結果は、次の２つの異なる方式で得ることができる。
【０１７８】
－　オブジェクト名（ベクトルによって表される）のリスト、
－　ｃ－ビーン（検索された各オブジェクト名毎に、アダプタはｃ－ビーンをインスタンス化する）のリスト。
【０１７９】
リモートｍ－ビーンのプロパティを読み取るために、低レベルadaptorM0インターフェース・レベルを用いる場合、プロパティ名が必要となる。高レベル・インターフェースを用いる場合、ｃ－ビーンを検索し、次いでプロパティに関連するゲッタを呼び出す。
【０１８０】
リモートｍ－ビーンのプロパティ、プロパティ名、およびプロパティ・オブジェクト型をセットすることは、低レベルadaptorM0インターフェース・レベルを用いる場合に必要とされる。ある値をセットする場合、オペレータ・クラスの名称を指定することができる。エージェント側では、指定されたオペレータがインスタンス化され、プロパティ値をセットするために呼び出される。低レベルadaptorM0インターフェース・レベルを用いる場合、変更のリストを用い、１つのメソッド・コールによって、いくつかのプロパティをセットすることができる。
【０１８１】
高レベルadaptorM0インターフェース・レベルを用いる場合、ｃ－ビーンが得られ、次いでデベロッパは、プロパティに関連する前述の１組を呼び出す。
【０１８２】
adaptorM0インターフェースによって、リモート・エージェント内においてｍ－ビーンのインスタンス化を要求することが可能となる。インスタンス化を要求する場合、新しいクラス・ローダを指定することができ、これを通じて、エージェントはインスタンス化する新しいクラスをロードすることになる。クラス・ローダは、そのオブジェクト名を用いて指定することができる。クラス・ローダを指定しない場合、エージェントはデフォルトのクラス・ローダを用いる。ローカルｍ－ビーンをインスタンス化する場合、新しく作成したｍ－ビーンを表すために、ｃ－ビーンを直接得ることができる。名称が与えられず、名称サーバが指定された場合、アダプタは、エージェント側でインスタンス化するＪａｖａクラスの名称を得るために、名称サーバに問い合わせを行なう。その他の場合、インスタンス化するクラスのクラス名を判定するのは、エージェントの役割である。エージェントにおいてｍ－ビーンをインスタンス化する場合、オブジェクトが永続的であることを明示的に要求することができる。
【０１８３】
adaptorM0インターフェースによって、Ｊａｖａオブジェクトをクライアントからエージェントに転送することが可能となる。これを行なうために、adaptorM0インターフェースは、オブジェクトをシリアル化し、当該オブジェクトを送り、エージェントにおいてこのオブジェクトをデシリアル化(deserialise)する。
【０１８４】

更に、adaptorMOインターフェースによって、リモート・エージェントからｍ－オブジェクトを除去することも可能となる。ｍ－ビーンは仮想マシンから除去されるのではなく、エージェントのオブジェクト・レポジトリからだけ除去される。

【０１８５】

図１７は、上述の管理システムを作成し動作させるステップの概要を示すフロー・チャートであり、ビーンに基づく環境を用いて少なくとも１つの管理ビーンを含むネットワーク管理モデルを定義するステップと、前記モデルをコンパイルして前記コンピュータ・ネットワーク管理システムを前記ビーンに基づく環境において実装するステップとを含む。

【０１８６】

ステップ１１０において、ビーンに基づく環境を用いてモデルを作成する。好適なビーンに基づく環境は、ビーンをＪａｖａＢｅａｎｓとしたＪａｖａ環境である。

【０１８７】

前述のように、ビーンは、１組のプロパティ、アクションを実行する１組のメソッド、ならびにイベントおよびイントロスペクションに対するサポートを与える。従来より、プロパティは、ビーンをプログラム的に操作すること、およびビーンのカスタム化に対応することを可能にし、メソッドはプロパティを実装し、イベントに対するサポートは、ビーンがイベントを発火させ、発火させることができるイベントを定義することを可能にする。イントロスペクションに対するサポートは、ビーンのプロパティ、イベント、およびメソッドを、外部から検査することを可能にする。

【０１８８】

したがって、ステップ１１０は、被管理リソースのプロパティをモデル化するための少なくとも１つのプロパティ、および／または前記被管理リソースのためのアクションをモデル化するためのメソッドおよび／または前記リソースのイベントをモデル化するための少なくとも１つのイベントに対するサポート、および／または前記ビーンの組成の外部分析を可能にするイントロスペクションに対するサポートをを与える、少なくとも１つの前記管理ビーンを生成することを含む。また、このステップは、被管理リソース間の関係および相互作用を表すものとして、管理ビーン間の関係および相互作用を定義することも含む。更に、このステップは、少なくとも１つの管理ビーンに関して、少なくとも１つのリスナ・イベントを定義することを含む。

【０１８９】

このネットワーク管理システムに関して、ビーンは従来の役割を超えて使用し、管理すべきリソースを直接モデル化するための管理ビヒクル（管理ビーン）を与えることができることが、初めて認識された。例えば、被管理リソースのプロパティ（例えば、メモリのサイズ、バッファに受信されているメッセージの数等のリソースの属性）をモデル化するために、管理ビーン内のあるプロパティを用いることができる。被管理リソースのアクション（例えば、システム休止）をモデル化するために、管理ビーンのメソッドを用いることができる。また、ビーンは、被管理リソースのためのイベント（例えば、ディスク・フル・イベント）に対するサポートを与えるために用いることも可能である。例えば、管理ビーンがリスナ・イベントに対するサポートを与えることができ、これによって、１つの管理ビーンが別の管理ビーン上のイベントに応答することが可能となる。イントロスペクションに対するサポートによって、管理ビーンは、ビーンの組成の外部的な分析、およびビーン間の情報の交換が可能となる。管理ビーンは、被管理リソース間の関係および相互作用を表すものとして、管理ビーン間の関係および相互作用の定義を含むことができる。

【０１９０】

ビーンは再利用可能なソフトウエア・コンポーネントであり、ビルダ・ツール（例えば、エディタまたはグラフィカル・ユーザ・インターフェース・ビルダ（ＧＵＩビルダ））において視覚的に操作することができるので、ステップ１１０において、ユーザは、ＪａｖａＢｅａｎｓ開発キットのような従来のビルダ・ツールを用いて、システム・リソース、それらの関数およびシステム・リソース間の関係および相互作用を定義するビーンを含む、管理システム・モデルを生成することができる。ビーン・モデルは、例えば、Ｊａｖａ

仮想マシン内においてＪａｖａＢｅａｎｓを用いて、ビーンに基づく環境を形成する仮想マシン内で定義される。これによって、管理システム・モデルの生成が格段に容易に行なえるようになる。モデルの確認およびデバッギングは、イントロスペクションおよびその他の技法を用いて、容易に行なうことができる。

【０１９１】

ステップ１１２において、一旦モデルが生成されたなら、例えば、図９に示したコンパイラ６０のような、ビーンに基づく環境のための従来のコンパイラを用いて、モデルを直接コンパイルすることができる。ビーンに基づく環境においてモデルを定義することにより、ビーンに基づくモデルを直接コンパイルし、例えば、Ｊａｖａコンパイラを用いてＪａｖａＢｅａｎｓをコンパイルすることによって、ビーンに基づく環境のための標準的なコンパイラを用いて、ビーンに基づくインプリメンテーションを形成することが可能となる。したがって、結果的に得られるビーン管理システムも、同じＪａｖａ環境内において定義される。これによって、コンパイラが、モデルの信頼性高く類似したインプリメンテーションを強制するので、この管理システムの生成段階が大幅に簡略化される。

【０１９２】

したがって、実行時において、ステップ１１４では、本明細書の冒頭に記載した管理システムは、従来のネットワーク管理システムの難問や欠点がなく、ロバスト性が高く容易に修正可能な構造を与える。

【０１９３】

ステップ１１４は、１つ以上の管理ビーンを、拡張可能なエージェント・フレームワークに登録し、ネットワーク通信プロトコルのための１つ以上のネットワーク・アダプタ（例えば、ネットワーク・アダプタ・ビーン（複数のビーン））を拡張可能なエージェント・フレームワークに登録し、ネットワーク・アダプタ（複数のアダプタ）を通じた管理ビーン（複数のビーン）に対する、ネットワークを通じた外部アクセスを可能にする、図１２に関連して説明したステップを含む。

【０１９４】

図１２に関して説明したように、拡張可能なエージェント・フレームワークは、関連するレポジトリ・ビーンを含み、１つ以上の管理ビーンおよび／またはネットワーク・アダプタを登録するステップは、１つ以上の管理ビーンおよび／またはネットワーク・アダプタをレポジトリ・ビーンに登録するステップから成るものとすることができる。

【０１９５】

以上、通信ネットワークを通じて、クライアント・マシンから、リモート・マシンにおけるビーンのようなオブジェクトにアクセスするコンピュータ実装方法について説明した（特に、図５および図６の説明を参照）。この方法は、リモート・マシンにおいて外部からアクセス可能なマシン・ページにオブジェクトをマッピングし、マシン・ページにおいてブラウザを用いてオブジェクトをブラウズするステップを含む。また、この方法は、リモート・マシンにおいてネットワーク・アダプタを起動するステップと、オブジェクトをエージェント・フレームワークに登録するステップと、ネットワーク・アダプタに、登録したオブジェクトを識別するように、エージェント・フレームワークに問い合わせを行なわせるステップと、ブラウザによってページにアクセスするステップと、アクセスしたページからオブジェクトを選択するステップとを含むことができる。

【０１９６】

特に、リモート・マシンにおいて、例えば、リモート・マシンにおけるエージェント・フレームワークにビーンを登録するステップと、リモート・マシンにおいて、マシン・ページ例えばＨＴＭＬページを生成し、登録したビーンを当該ページに含ませるステップと、クライアント・ステーションにおいてブラウザを用いて、リモート・マシンにおいてネットワーク・アダプタおよびマシン・ページによってビーンをブラウズするステップとを含む方法について説明した（特に、図５および図６の説明を参照）。ビーンがリード・オンリ・プロパティを有する場合、ＨＴＭＬフォームを生成することができる。ネットワーク・アダプタは、恐らく全てのビーンが登録されているレポジトリを介して、登録されてい

るビーンを識別するようにフレームワークに問い合わせる構成とすることができる。ビーンとの相互作用は、クライアント・マシンにおいて、遠隔的に修正可能なビーンの表現を表示し、表示ビーン表現のユーザ選択に応答して、遠隔的に修正可能なビーンのプロパティを表示し、ユーザ入力に応答して、ビーンの選択したパラメータを遠方より修正することによって実行することができる。

【０１９７】

更に、ビーンが位置するネットワーク・ステーションにおいて、ＨＴＭＬページ上にビーンをマッピングすることによって、リモート・ビーン・アクセスを行なうシステムおよび機構について説明した。同じ仮想マシン上で実行するＨＴＭＬ生成器を用いて、ビーンをＨＴＭＬページに動的にマッピングする。こうして、ＨＴＴＰまたはＨＴＭＬプロトコルに対応するウエブ・ブラウザを用いて、ビーンをブラウズし修正するためのリモート・アクセスが可能となる。リモート・ビーン・アクセス方法の用途は、ネットワーク管理システムのためのネットワーク・エージェントという分野にある。

【０１９８】

以上、本発明の特定実施形態について説明したが、特許請求の範囲に定義される本発明の精神および範囲内において、多くの変更／追加および／または交換も可能であることは認められよう。特許請求の範囲を参照することにより、本発明の精神および範囲内において、請求項に明示的に列挙されているもの以外の従属項の特徴を他の従属項および／または独立項の特徴と適宜組み合わせることも可能であることを注記すべきであろう。

【図面の簡単な説明】

【図１】通信ネットワークを通じて接続された３箇所のステーションを表す図である。

【図２】図２と図２のＡとから構成される。図１のステーションに対するコンピュータ・サーバの概略図を形成する図である。

【図３】管理対象ステーションに対するエージェントの概略図である。

【図４】図３のエージェントを示す別の図である。

【図５】エージェントのコンフィギュレーションの一例を示す図である。

【図６】図５に示すエージェントを用いる動作を示すフロー・チャートである。

【図７】管理システムのコンフィギュレーションの一例を示す図である。

【図８】管理システムのコンフィギュレーションの別の例を示す図である。

【図９】図８のコンフィギュレーションの作成の一態様を示す図である。

【図１０】図８のシステムの動作を示すフロー・チャートである。

【図１１】命名サービスの動作を示すフロー・チャートである。

【図１２】汎用エージェントの作成の処理を示すフロー・チャートである。

【図１３】コンパイラの別の処理を示す図である。

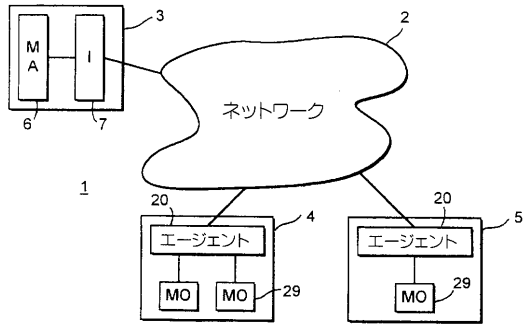
【図１４】コンパイラの別の処理を示す図である。

【図１５】Ａ及びＢから構成される。管理ビーンをコンパイルする効果を示すために用いる図である。

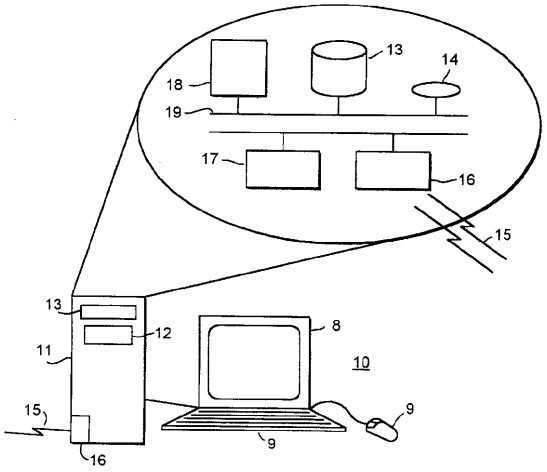
【図１６】Ａ及びＢから構成される。同様に管理ビーンをコンパイルする効果を示すために用いる図である。

【図１７】管理システムを生成するステップを示すフロー・チャートである。

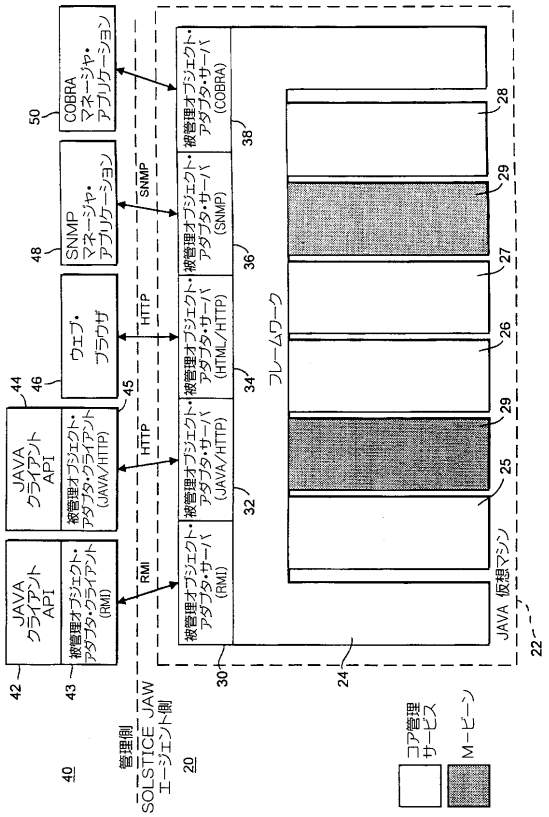
【図１】



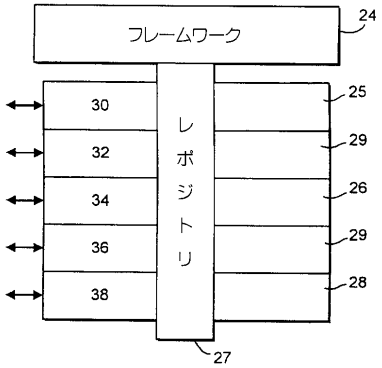
【図２】



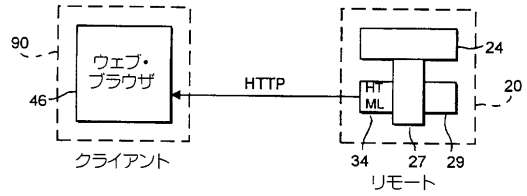
【図３】



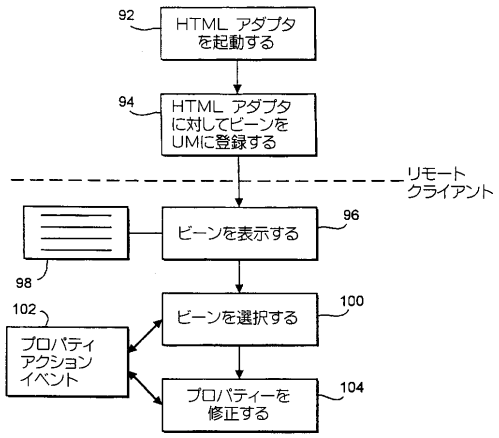
【図４】



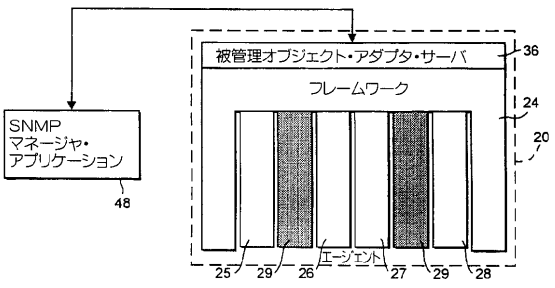
【図５】



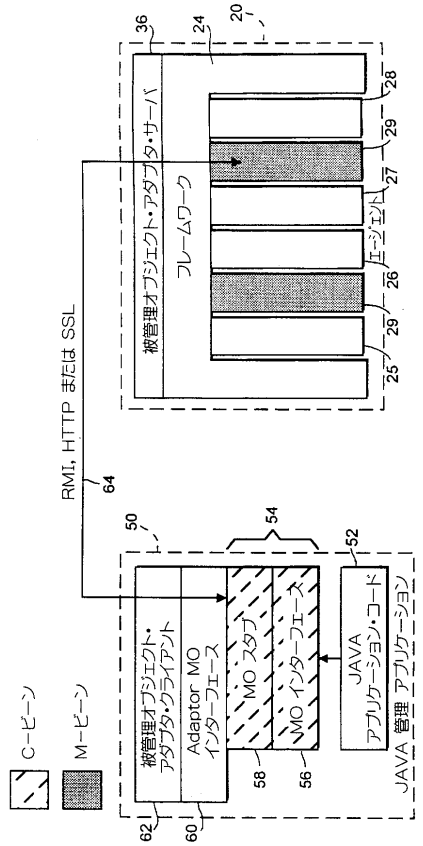
【図６】



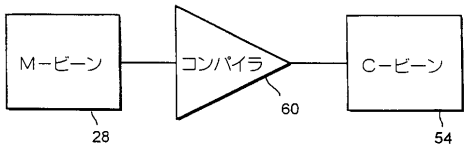
【図７】



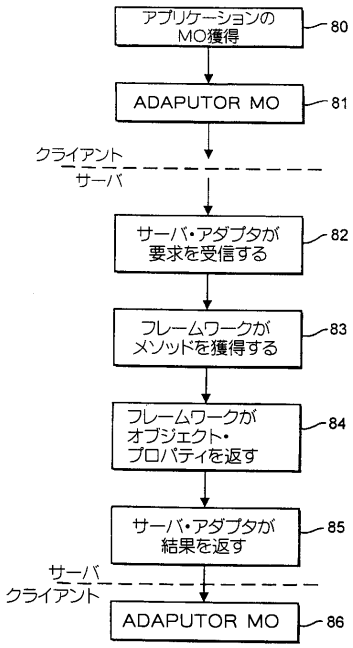
【図８】



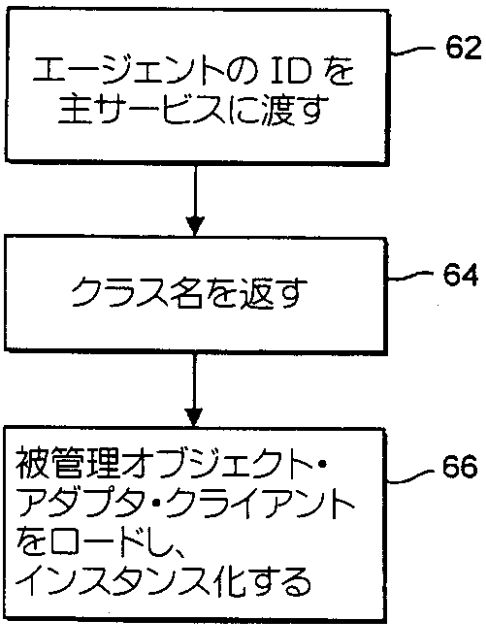
【図９】



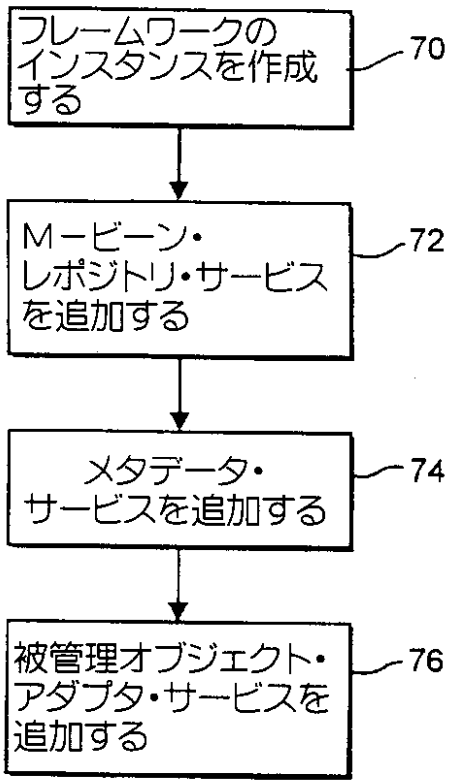
【図１０】



【図１１】

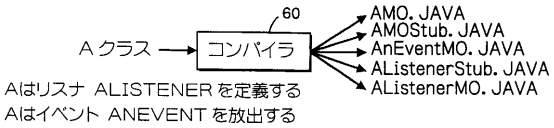


【図１２】



【図１３】

【図１４】



【図１５】

A

```
package sunw.jaw.example.mo.Simple;
/**
 * Generated by mogen Compiler version;
 *        @ (#) Main.java 1.11 97/05/16 SMI
 */
import java.jaw.common.*;
import java.jaw.client.mo.*;

public interface SimpleMO extends ManagedObject {
  public String getState()
     throws InstanceNotFoundException, CommunicationException,
       IllegalAccessException, ServiveNotFoundException,
       PropertyNotFoundException;

  public Integer getNbChanges ()
     throws InstanceNotFoundException, CommunicationException,
       IllegalAccessException, ServiceNotFoundException,
       PropertyNotFoundException;

  public void setState (String value)
     throws InstanceNotFoundException, CommunicationException,
       IllegalAccessException, ServiceNotFoundException,
       PropertyNotFoundException, InvalidPropertyValueException,
       InstantiationException, ClassNotFoundException;

}
```

B

```
package sunw.jaw.example.mo.Simple;

/**
 * Very simple of definition of a Beans MO
 * No special import required.
 */

public class Simple implements java.io.Serializable {
        private String state;
        private Integer nbChanges;

        public Simple () {
                state       = "construct";
                nbChanges = new Integer (0);
        }

        public String getState () {
                return (state);
        }

        public void setState (String s) {
                state = s;
                nbChanges = new Integer (nbChanges.intValue () + 1);
        }

        public Integer getNbChanges () {
                return (nbChanges) ;
        }
}
```

【図１６】

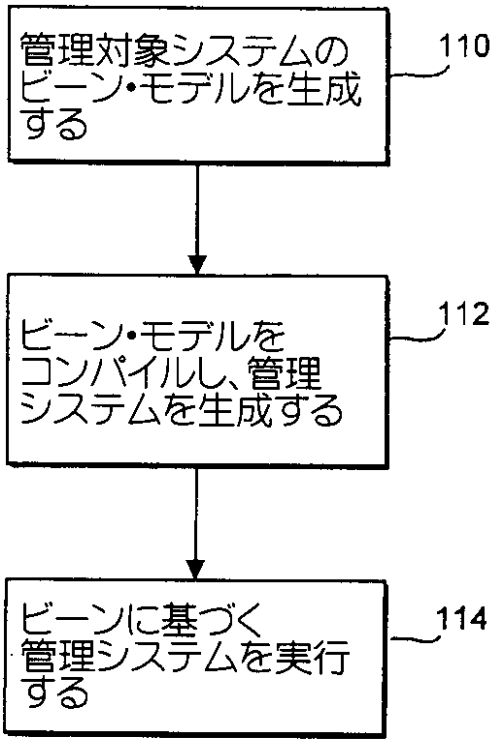
A

```
....
public interface simpleMBeanMO extends ManagedObject {
....
public Boolean performReverse (integer p0)
     throws InstanceNotFoundException, CommunicationException,
     IllegalaccessException, ServiceNotFoundException,
     NoSuchMethodException;

  public void performReverse
    throws InstanceNotFoundException, CommunicationException,
     IllegalAccessException, ServiceNotFoundException,
     NoSuchMethodexception;

....
}
```

B

```
....
public class simpleMBean implements java.io.Serializable {
...
public void performReverse () {
        StringBuffer buf = new StringBuffer (name);

        name = buf.reverse ().toString ();
   }

   public Boolean performReverse (Integer nFirst) {
        StringBuffer buf  = new StringBuffer ();
        int             n     = nFirst.intValue ();

        if (name.length () < n + 1) {
          return (new Boolean (false));
        }

        buf.append (name.substring (0, n));

        name = new String (buf.reverse ().toString () +
name.substring (n));
        return (new Boolean (true));
   }
 ...
 }
```

【図１７】



フロントページの続き

(74)代理人 100075236
弁理士 栗田 忠彦
(74)代理人 100075270
弁理士 小林 泰
(74)代理人 100087424
弁理士 大塚 就彦
(72)発明者 オスマン・アブドゥール・イスマエル
フランス共和国 ３８０００ グルノーブル，リュ・ダンフェル・ロシュロー ２１
(72)発明者 セルジュ・アンドレ・リゴリ
フランス共和国 ３８１２０ プロヴィズオー，プランファイ

審査官 須藤 竜也

(56)参考文献 安村義孝，WWW－ＯＯＤＢ連携システムを用いたコンテンツ管理の実現，第５５回（平成９年後期）全国大会講演論文集（３），日本，社団法人情報処理学会，１９９７年 9月２６日，p.341～342
エバンス エリック，WWW環境における分散アプリケーション開発，日経コンピュータ，日本，日経ＢＰ社，１９９７年 9月２９日，第４２７号，ｐ．２４７～２５７
成田 雅彦 他，ＣＯＲＢＡとＪａｖａ 分散オブジェクト技術，日本，株式会社ソフト・リサーチ・センター，１９９７年 5月２５日，初版，ｐ．１２２～１２６
【特集】サーバ・サイドのJavaとActiveX，Open Network １９９７年8月号 ，株式会社アスキー，１９９７年 8月 1日，pp.45-55
谷島 宣之，岐路に立つＪａｖａ 基幹系構築ユーザーの期待高まるサンの仕様決定遅れには不満の声，日経コンピュータ NIKKEI COMPUTER，日本，日経ＢＰ社，１９９７年 9月１５日，第426号，pp.162～169
磯辺 裕一，ＷｅｂサーバーとＤＢエンジンの連携 Ｗｅｂサーバーと既存システムの連携 3層モデルによる連携システム構築，ＵＮＩＸ ＵＳＥＲ 第5巻 第１２号，日本，ソフトバンク株式会社 SOFTBANK Corporation，１９９６年１２月 1日，pp.39～61
岩山 知三郎，Ｊａｖａの近未来－世界が変わる半年後 Ｂｅａｎｓの登場でソフトウェア開発パラダイムが変わる，Ｃｏｍｐｕｔｏｐｉａ，株式会社コンピュータ・エージ社，１９９７年 9月 1日，第32巻 第372号，pp.64-67

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/46
G06F 13/00