(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5214108号
(P5214108)
(45)発行日　平成25年6月19日(2013.6.19)　(24)登録日　平成25年3月8日(2013.3.8)

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

請求項の数 8 (全 26 頁)

(21)出願番号　特願2006-20931(P2006-20931)
(22)出願日　平成18年1月30日(2006.1.30)
(65)公開番号　特開2006-209780(P2006-209780A)
(43)公開日　平成18年8月10日(2006.8.10)
審査請求日　平成21年1月8日(2009.1.8)
(31)優先権主張番号　11/046,127
(32)優先日　平成17年1月28日(2005.1.28)
(33)優先権主張国　米国(US)
(31)優先権主張番号　11/045,756
(32)優先日　平成17年1月28日(2005.1.28)
(33)優先権主張国　米国(US)

前置審査

(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　110001243
特許業務法人　谷・阿部特許事務所
(74)復代理人　100115624
弁理士　濱中　淳宏
(72)発明者　エスベン　ニハース　クリストファーセン
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

最終頁に続く

(54)【発明の名称】エンティティパターンを用いてデータモデルの構造をキャプチャするシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
処理ユニットおよびシステムメモリを有するコンピュータシステムにおいてデータモデルの構造をキャプチャするコンピュータ実行方法であって、前記コンピュータ実行方法は、
前記処理ユニットが、ユーザインターフェースを介して、第１のエンティティロールを、第１のホストエンティティに適用するという指示を受信するステップであって、前記第１のエンティティロールを適用することにより、前記第１のホストエンティティのプロパティに、前記第１のエンティティロールのプロパティがマッピングされ、前記第１のエンティティロールは、第２のエンティティロールとの第１のリレーションが予め指定されている、受信するステップと、
前記処理ユニットが、前記ユーザインターフェースを介して、前記第２のエンティティロールを、第２のホストエンティティに適用するという指示を受信するステップであって、前記第２のエンティティロールを適用することにより、前記第２のホストエンティティのプロパティに、前記第２のエンティティロールのプロパティがマッピングされ、前記第２のエンティティロールは、前記第１のエンティティロールとの第２のリレーションが予め指定され、前記第１のエンティティロールおよび前記第２のエンティティロールは、同じエンティティパターンに含まれる、受信するステップと、
前記処理ユニットが、前記第１のホストエンティティへの前記第１のエンティティロールの適用の妥当性、および前記第２のホストエンティティへの前記第２のエンティティロ

ールの適用の妥当性を検証するステップであって、前記検証するステップは、
前記第１のエンティティロールのプロパティが、前記適用された第１のホストエンティティのプロパティにマッピングされているかどうか、および前記第２のエンティティロールのプロパティが、前記適用された第２のホストエンティティのプロパティにマッピングされているかどうかを検証するステップと、
前記第１のエンティティロールにおいて指定されている前記第１のリレーションにしたがって、前記第１のホストエンティティが、前記第２のホストエンティティと関連付いているかどうか、および前記第２のエンティティロールにおいて指定されている前記第２のリレーションにしたがって、前記第２のホストエンティティが、前記第１のホストエンティティと関連付いているかどうかを検証するステップと
を含む、検証するステップと
を備えたことを特徴とするコンピュータ実行方法。
【請求項２】
前記第１のエンティティロールは第１の対応クラス、前記第２のエンティティロールは第２の対応クラスを各々有し、前記第１の対応クラスおよび前記第２の対応クラスの各々は、前記第１のエンティティロールおよび前記第２のエンティティロールの各々において定義される構造を有し、かつ前記第１のエンティティロールおよび前記第２のエンティティロールと同じエンティティパターンに含まれる他のエンティティロールへのナビゲートを可能にすることを特徴とする請求項１に記載のコンピュータ実行方法。
【請求項３】
前記エンティティパターンは、宣言的に適用された機能性を含むことを特徴とする請求項１に記載のコンピュータ実行方法。
【請求項４】
データモデルの構造をキャプチャするコンピュータ実行方法をコンピュータに実行させるコンピュータ実行可能命令を格納するコンピュータ可読記憶媒体であって、前記コンピュータ実行方法は、
前記コンピュータ上の処理ユニットが、ユーザインターフェースを介して、第１のエンティティロールを、第１のホストエンティティに適用するという指示を受信するステップであって、前記第１のエンティティロールを適用することにより、前記第１のホストエンティティのプロパティに、前記第１のエンティティロールのプロパティがマッピングされ、前記第１のエンティティロールは、第２のエンティティロールとの第１のリレーションが予め指定されている、受信するステップと、
前記処理ユニットが、前記ユーザインターフェースを介して、前記第２のエンティティロールを、第２のホストエンティティに適用するという指示を受信するステップであって、前記第２のエンティティロールを適用することにより、前記第２のホストエンティティのプロパティに、前記第２のエンティティロールのプロパティがマッピングされ、前記第２のエンティティロールは、前記第１のエンティティロールとの第２のリレーションが予め指定され、前記第１のエンティティロールおよび前記第２のエンティティロールは、同じエンティティパターンに含まれる、受信するステップと、
前記処理ユニットが、前記第１のホストエンティティへの前記第１のエンティティロールの適用の妥当性、および前記第２のホストエンティティへの前記第２のエンティティロールの適用の妥当性を検証するステップであって、前記検証するステップは、
前記第１のエンティティロールのプロパティが、前記適用された第１のホストエンティティのプロパティにマッピングされているかどうか、および前記第２のエンティティロールのプロパティが、前記適用された第２のホストエンティティのプロパティにマッピングされているかどうかを検証するステップと、
前記第１のエンティティロールにおいて指定されている前記第１のリレーションにしたがって、前記第１のホストエンティティが、前記第２のホストエンティティと関連付いているかどうか、および前記第２のエンティティロールにおいて指定されている前記第２のリレーションにしたがって、前記第２のホストエンティティが、前記第１のホストエン

ティティと関連付いているかどうかを検証するステップと
を含む、検証するステップと
を備えたことを特徴とするコンピュータ可読記憶媒体。
【請求項５】
前記第１のエンティティロールは第１の対応クラス、前記第２のエンティティロールは第２の対応クラスを各々有し、前記第１の対応クラスおよび前記第２の対応クラスの各々は、前記第１のエンティティロールおよび前記第２のエンティティロールの各々において定義される構造を有し、かつ前記第１のエンティティロールおよび前記第２のエンティティロールと同じエンティティパターンをに含まれる他のエンティティロールへのナビゲートを可能にすることを特徴とする請求項４に記載のコンピュータ可読記憶媒体。
【請求項６】
前記エンティティパターンは、宣言的に適用された機能性を含むことを特徴とする請求項４に記載のコンピュータ可読記憶媒体。
【請求項７】
データモデルの構造をキャプチャするコンピュータシステムであって、前記コンピュータシステムは、
処理ユニットと、
システムメモリと、
前記処理ユニットに実行させる手段であって、前記実行させる手段は、
ユーザインターフェースを介して、第１のエンティティロールを、第１のホストエンティティに適用するという指示を受信する手段であって、前記第１のエンティティロールを適用することにより、前記第１のホストエンティティのプロパティに、前記第１のエンティティロールのプロパティがマッピングされ、前記第１のエンティティロールは、第２のエンティティロールとの第１のリレーションが予め指定されている、受信する手段と、
前記ユーザインターフェースを介して、前記第２のエンティティロールを、第２のホストエンティティに適用するという指示を受信する手段であって、前記第２のエンティティロールを適用することにより、前記第２のホストエンティティのプロパティに、前記第２のエンティティロールのプロパティがマッピングされ、前記第２のエンティティロールは、前記第１のエンティティロールとの第２のリレーションが予め指定され、前記第１のエンティティロールおよび前記第２のエンティティロールは、同じエンティティパターンに含まれる、受信する手段と、
前記第１のホストエンティティへの前記第１のエンティティロールの適用の妥当性、および前記第２のホストエンティティへの前記第２のエンティティロールの適用の妥当性を検証する手段であって、前記検証する手段は、
前記第１のエンティティロールのプロパティが、前記適用された第１のホストエンティティのプロパティにマッピングされているかどうか、および前記第２のエンティティロールのプロパティが、前記適用された第２のホストエンティティのプロパティにマッピングされているかどうかを検証する手段と、
前記第１のエンティティロールにおいて指定されている前記第１のリレーションにしたがって、前記第１のホストエンティティが、前記第２のホストエンティティと関連付いているかどうか、および前記第２のエンティティロールにおいて指定されている前記第２のリレーションにしたがって、前記第２のホストエンティティが、前記第１のホストエンティティと関連付いているかどうかを検証する手段と
を含む、検証する手段と
を含む、実行させる手段と
を備えたことを特徴とするコンピュータシステム。
【請求項８】
前記第１のエンティティロールは第１の対応クラス、前記第２のエンティティロールは第２の対応クラスを各々有し、前記第１の対応クラスおよび前記第２の対応クラスの各々は、前記第１のエンティティロールおよび前記第２のエンティティロールの各々において

定義される構造を有し、かつ前記第１のエンティティロールおよび前記第２のエンティティロールと同じエンティティパターンに含まれる他のエンティティロールへのナビゲートを可能にすることを特徴とする請求項７に記載のコンピュータシステム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、ソフトウェア開発のビジネスフレームワークに関する。より詳細には、本発明は、データモデルの構造パターンを再利用可能な形で表現する方法で、アプリケーションやコンピュータプログラムにロジックをフレキシブルに実装することを可能にするフレームワークに関する。

【背景技術】

【０００２】

本特許文書の開示部分には、著作権保護の対象となる資料が含まれている可能性がある。その著作権保有者は、特許文書または特許開示が、特許商標局（Ｐａｔｅｎｔ　ａｎｄ　Ｔｒａｄｅｍａｒｋ　Ｏｆｆｉｃｅ）の特許ファイルまたは記録にある場合は、誰によってファクシミリ複製されても異議はないが、そうでない場合は、どのようなものであれ、すべての著作権を保有する。本文書には、以下の表示が適用されるものとする：Ｃｏｐｙｒｉｇｈｔ（Ｃ）Ｍｉｃｒｏｓｏｆｔ　Ｃｏｒｐ.

ビジネスアプリケーション用ソフトウェアを作成する場合は、ビジネスにおいて、経理、給与、人事、販売注文、従業員管理、顧客関係管理などのビジネスオペレーションを管理および分析するために、様々なメカニズムが一般的に用いられてきたことを考慮しなければならない。そうした機能を提供するツールは、多くの場合、コンピュータソフトウェアを用いて実装され、フレームワークまたはビジネスフレームワークと呼ばれる、コンピュータによって自動化された支援システムの一種を用いて構築されることが可能である。ユーザが、様々なビジネスオペレーションに関連するデータを容易に入力および閲覧できるユーザインターフェースが、ソフトウェアパッケージとして提供されることも可能である。このソフトウェアパッケージは、データベースに保管されているデータにアクセスしたり、それらのデータを更新したりするようにも構成される。

【０００３】

ビジネスアプリケーションは、受注処理や発送など、様々なビジネスイベントを処理するように設計される。ビジネスアプリケーションは、コードを用いて実装されるアプリケーション機能を含む。コードに加えて、ビジネスアプリケーションは、ビジネスアプリケーションの実行時にコードと対話する、いくつかのアブストラクションを含む。例えば、ＭＩＣＲＯＳＯＦＴ　ＢＵＳＩＮＥＳＳ　ＦＲＡＭＥＷＯＲＫ（登録商標）（ＭＢＦ）は、フレームワークで定義される、広い範囲のアブストラクション（エンティティ（Ｅｎｔｉｔｙ）、オペレーション（Ｏｐｅｒａｔｉｏｎ）、．．．）をビジネス開発者に提供し、ビジネス開発者がビジネスロジックを再利用性のためにキャプチャすることを可能にする単一のアブストラクション（プロパティパターン（Ｐｒｏｐｅｒｔｙ　Ｐａｔｔｅｒｎ））を提供する。例えば、あるアブストラクションは、顧客または販売注文に関連するデータの保管をモデル化するビジネスエンティティである。こうしたエンティティ（またはオブジェクト）は、データを保管するクラスを含む。

【発明の開示】

【発明が解決しようとする課題】

【０００４】

ビジネスアプリケーションは、１つまたは複数のこのようなエンティティを関与させる、多くの様々なパターン（すなわち、エンティティパターン（Ｅｎｔｉｔｙ　Ｐａｔｔｅｒｎ））を含む。多くの場合、同一アプリケーション内で同じパターンが手動で何度も繰り返され、パターンをキャプチャする方法は用いられない。その結果、オリジナルのパターンが、最後には実質的に不明瞭なものになる。開発者が同じパターンを再度適用したいと考えても、どのアプリケーション(パターンを適用すること、またはパターンを適用し

たアプリケーションソフトウェア)がオリジナルであったか見分けがつかなくなる。オリジナルパターンを変更した場合は、すべてのアプリケーションがそのパターンに対して妥当かどうかを手動で検査し、必要に応じて変更しなければならない。

【０００５】

この点において、複数のエンティティと、そのエンティティの妥当性を設計時に自動的に検査する関連ロジックとからなる構造パターンをキャプチャする方法を開発者に提供するシステムおよび方法が必要である。

【課題を解決するための手段】

【０００６】

本発明は、上述の、従来技術の不十分な点に鑑み、複数のエンティティと、そのエンティティの妥当性を設計時に自動的に検査する関連ロジックとからなる構造パターンをキャプチャするシステムおよび方法を提供するとともに、適用されるすべてのパターンをサポートし、そのパターン固有のコードを実行する実行時フレームワークを提供する。開発者は、ソフトウェアの開発時に、そのキャプチャされたパターンを宣言的に再利用することが可能である。一般的な利点として、アプリケーションは、実行時にモデルドリブンであり、実行時にモデルデータを解釈する。パターンのアプリケーションおよび／または使用は、アプリケーションまたはソフトウェアプログラムのモデルの一部をなす。

【０００７】

本発明のいくつかの実施形態では、少なくとも１つのエンティティ構造に基づいてエンティティパターンをデータモデル内に作成することを含む、データモデルの構造をキャプチャする方法を提供する。エンティティパターンを適用する場合は、そのアプリケーションがエンティティパターンに適合しているかどうかを検査される。この妥当性検査のプロセスは、エンティティパターンのアプリケーションの一部としてエンティティパターン情報を保持することと、そのアプリケーションがエンティティパターンに適合しているかどうかを、前記保持されたエンティティパターン情報を用いて調べることとを含むことが可能である。

【０００８】

エンティティパターンは、そのアプリケーションがエンティティパターンに適合しているかどうかを調べることに用いられるために、少なくとも１つのエンティティロール（Ｅｎｔｉｔｙ　Ｒｏｌｅ）を有する。さらに、このプロセスは、エンティティロールによって記述される構造を有する、エンティティロールの第１の対応クラスを生成することを含むことが可能である。この第１のクラスは、実行時にエンティティパターンの他のロールにナビゲートすることを可能にする。開発者に対して隠されている可能性があるメタデータを取得する１つの方法は、利用可能なメタデータを含む、エンティティロールの第２の対応クラスを実行時に生成することである。本発明の他の有利点および特徴については後述する。

【０００９】

本発明による、エンティティパターンを用いてデータモデルの構造をキャプチャするシステムおよび方法について、添付図面を参照しながら、さらに詳しく説明する。

【発明を実施するための最良の形態】

【００１０】

本発明の各種実施形態について十分な理解が得られるように、以下の記述および添付図面において、特定の具体的な細目について説明する。以下の開示では、本発明の各種実施形態を不要に不明瞭にすることを避けるために、多くはコンピューティング技術およびソフトウェア技術に関連する、特定のよく知られた細目については説明しない。さらに、関連技術分野の当業者であれば、後述する細目の１つまたは複数がなくても本発明の他の実施形態を実施することが可能であることを理解されよう。最後に、以下の開示では、ステップおよびシーケンスを参照して各種方法を説明しているが、そのように説明しているのは、本発明の実施形態の明確な実施態様を提供するためであって、それらのステップおよびステップのシーケンスが本発明の実施に必須であると理解されてはならない。

【００１１】
概要
エンティティパターンを用いてデータモデルの構造をキャプチャするシステムおよび方法について説明する。本発明は、例えば、ＭＩＣＲＯＳＯＦＴ ＢＵＳＩＮＥＳＳ ＦＲＡＭＥＷＯＲＫ（登録商標）（ＭＢＦ）を用いて実装することが可能なエンティティパターンに関する。しかしながら、本発明は、例示的な実装の説明のためにＭＢＦに関連する用語で説明されることがしばしばあるものの、特定のフレームワークに限定されるものではなく、本明細書で説明する概念は、他の開発フレームワーク（例えば、ＯＲＡＣＬＥ ＡＰＰＬＩＣＡＴＩＯＮ ＤＥＶＥＬＯＰＭＥＮＴ ＦＲＡＭＥＷＯＲＫ（登録商標）（ＡＤＦ）など）にもおおむね適用可能である。エンティティパターンは、本明細書で開示されるように、エンティティグラフに関する構造情報と機能情報の両方を包含し、これらの情報が具体的なアプリケーションに適用された場合に、エンティティパターンに基づいてこれらの情報の妥当性を検証(検査)することによって、アプリケーションの全体的な品質および一貫性を高めることが可能である。
【００１２】
最初に、エンティティパターンの設計、エンティティパターンロールを適用すること、および適用されたエンティティパターンロールの妥当性検査の各プロセスについて説明し、その後に、エンティティパターンロールのプログラミングモデルについて説明する。次に、エンティティパターンアーキテクチャを定義する生成コードおよび実行時フレームワークについて、サンプルの生成コードリストを用いて説明する。最後に、図６および７で、本明細書で説明するシステムおよび方法と関連して使用することに一般的に好適と考えられるコンピューティング環境およびネットワーク環境を示す。図６および７の内容は、説明を目的とする一般的なものであり、これに対応する記述は、本明細書の最後の、「例示的コンピューティング環境およびネットワーク環境」のセクションにある。本明細書で用いる定義および用語の簡単なリストを付録Ａに示す。
【００１３】
エンティティパターンの設計およびエンティティパターンロールを適用すること
図１に、エンティティを作成および使用するプロセスの高レベルビューのフローチャートを示す。通常、パターンを使用していることは、開発者だけが認識しているか、せいぜい、生成ウィザードが認識しているだけである。適用されたパターンについての情報は、そのアプリケーションとともには保持されない。エンティティパターンは、この情報を適用の一部として保持する。これによって、適用またはパターンが変更された場合に、適用されたパターンにすべてのアプリケーションが適合しているかどうかを再検査することが可能になる。
【００１４】
最初に、ＭＢＦモデルエディタで、エンティティパターンを設計する（１０１）。次に、エンティティパターンロールを具体的なエンティティに適用する（１０２）。次に、具体的なエンティティがエンティティパターンに適合することを確認するために、エンティティパターンロールを基準にして、具体的なエンティティの妥当性を検証する（１０３）。エンティティパターンは、他の任意のＭＢＦアブストラクションと同様に、モデルエディタ内で設計される。エンティティパターンを設計する場合は、ＢＥＤとよく似たビジュアルデザイナを用いることが可能である。
【００１５】
次に、パターン固有のコードが実行時実行される（１４０）。実行時フレームワークは、すべての適用されたパターンをサポートし、パターン固有のコードを実行する。一般的な有利点として、実行時オペレーションは、モデルドリブンであって、実行時にモデルデータを解釈し、エンティティは、実行時に、適用されたロールを照会されることが可能であって、ロールクラスのインスタンスを返されることが可能である。パターンのアプリケーションおよび使用は、アプリケーション／ソフトウェアプログラムのモデルの一部をなす。

【００１６】
次に、図２は、モデルエディタ１０７に表示された例示的エンティティパターン１０４を示すサンプルスクリーンショットである。エンティティパターン１０４は、エンティティロールクラス（Ｅｎｔｉｔｙ　Ｒｏｌｅ　Ｃｌａｓｓ）１０５を含む最上位要素である。例として表示されているのは、ＨｅａｄｅｒＬｉｎｅＰａｔｔｅｒｎエンティティパターン１０６である。エンティティロールは、プロパティパターンと同じクラス（タイプ）／インスタンスパターンの後に続く。プロパティパターンがプロパティパターンタイプ（Ｐｒｏｐｅｒｔｙ　Ｐａｔｔｅｒｎ　Ｔｙｐｅ）のインスタンスであるのと同様に、エンティティロールは、エンティティロールクラス（ＥｎｔｉｔｙＲｏｌｅＣｌａｓｓ）１０５のインスタンスである。プロパティパターンに関しては、モデルエディタ１０７内でクラスとインスタンスの両方にエンティティロールという用語を用いることによって、プロパティパターンを（ネーミングに関して）開発者から隠すことを選択することが可能である。以下では、クラスとインスタンスの両方に、エンティティロールという用語を用いる。

【００１７】
エンティティロール１０５は、当然ながら、プロパティ（Ｐｒｏｐｅｒｔｉｅｓ）１０８、アソシエーション（Ａｓｓｏｃｉａｔｉｏｎ）１０９、コンポジション（Ｃｏｍｐｏｓｉｔｉｏｎ）１１０、プロパティパターンなど、多くの同じモデル要素（エンティティ）からなる。それらの要素のほとんどは、それぞれの具体的な相手として、同じ属性のサブセットを有する。プロパティ１０８に関しては、エンティティロール１０５上のプロパティの定義は、定義されたプロパティをエクスポーズし、それらを、実行時メタデータを用いて、それぞれの対応する、ホストエンティティ上のプロパティにマッピングするエンティティロール１０５実行時クラスを生成するための十分な情報を含んでいなければならない。例えば、Ａｃｃｅｓｓｏｒ、ＡｌｌｏｗＮｕｌｌ、ＡｒｒａｙＬｅｎｇｔｈ、およびＴｙｐｅである。アソシエーション１０９およびコンポジション１１０に関しては、これらは、エンティティロール１０５レベルにあって、他のエンティティロールとのリレーションを定義する。指定されたリレーションシップは、エンティティロール１０５が実際のエンティティに適用される際に満たされなければならない。デコレーション（Ｄｅｃｏｒａｔｉｏｎｓ）１１１は、エンティティロール１０５が具体的なエンティティに適用される場合に指定されなければならない、メタデータの定義である。生成されたエンティティロールクラス１０５は、このメタデータをラップするプロパティを有する。デコレーションは、既にプロパティパターンに対して用いられている。

【００１８】
ホストエンティティ上の他のエンティティロールの存在は、エンティティロール１０５上で定義されることが可能である。これは、エンティティロール間の継承がどのように記述されるかである。例えば、Ｃｏｍｐａｎｙエンティティロールは、Ｐａｒｔｙエンティティロールがホストエンティティ上にも存在しなければならないことを定めている。

【００１９】
ホストエンティティ上のプロパティパターンの存在は、エンティティロール１０５上で定義可能である。後で詳述するエンティティロール適用ウィザード（Ａｐｐｌｙ　Ｅｎｔｉｔｙ　Ｒｏｌｅ　Ｗｉｚａｒｄ）の一部として、定義されたプロパティパターンをマッピングするか作成するかの選択肢が提示される。プロパティパターンロールは、このレベルで定義されたプロパティおよびリレーションを用いてナビゲートされることによって、エンティティパターン１０４内でバインドされることが可能である。ロールは、それらをバインドする、エンティティロール１０５のコンシューマのために「バインドされない」ままであることも可能である。最後に、プロパティバリデータの存在は、プロパティパターンによって定義可能であるのと同様に、エンティティロールクラスプロパティ上で定義可能である。さらに、エンティティを用いることが可能である方法で、エンティティロール１０５を処理するオペレーションを設計することが可能である。また、生成されたエンティティロールクラスは、部分的であり、エンティティロールクラス１０５モデル要素内

で定義されたプロパティアブストラクションに対して動作する別のメソッドおよびコモンランゲージランタイム（Ｃｏｍｍｏｎ Ｌａｎｇｕａｇｅ Ｒｕｎｔｉｍｅ）（ＣＬＲ）プロパティにより拡張可能である。ＣＬＲは、．ＮＥＴプログラムコードの実行を管理する実行時環境であって、メモリおよび例外の管理、デバッグおよびプロファイリング、セキュリティなどのサービスを提供する。ＣＬＲは、．ＮＥＴフレームワークの主要コンポーネントである。ＣＬＲは、仮想実行システム（Ｖｉｒｔｕａｌ Ｅｘｅｃｕｔｉｏｎ Ｓｙｓｔｅｍ （ＶＥＳ））とも呼ばれている。

【００２０】

エンティティロールは、モデルエディタ１０７内で、ドラッグアンドドロップを用いてエンティティに適用される。エンティティロール１０５は、エンティティパターン１０６からドラッグされ、エンティティ上のＥｎｔｉｔｙＲｏｌｅｓフォルダにドロップされる。エンティティロールノード１０５を右クリックすると、モデル内で使用可能なエンティティパターンからエンティティロールを選択するダイアログが実行される。

【００２１】

次に、図３は、本発明による、ビジネスエンティティデザイナ（Ｂｕｓｉｎｅｓｓ Ｅｎｔｉｔｙ Ｄｅｓｉｇｎｅｒ）１１２による、エンティティロールのアプリケーションの例を示すサンプルスクリーンショットである。エンティティロールは、ＢＥＤ １１２においてドラッグアンドドロップを行うことにより適用可能である。設計されたエンティティパターン１１３は、ｆｉｒｓｔ－ｃｌａｓｓ ｃｉｔｉｚｅｎとしてツールボックス１１４に表示される。エンティティロール１１５を、例えば、ツールボックス１１４からドラッグしてエンティティ１１６でドロップすると、エンティティロール１１５が適用される。このドロップ操作により、エンティティロールウィザード（Ｅｎｔｉｔｙ Ｒｏｌｅ Ｗｉｚａｒｄ）がトリガされる場合がある。このウィザードは、必要なすべてのプロパティ、アソシエーション、コンポジション、および他の、エンティティロールのために記述される要素をマッピングまたは作成するための、簡単なユーザエクスペリエンスを与える。

【００２２】

次に、図４ａは、本発明による、図１の、エンティティパターンロールを基準にして具体的なエンティティの妥当性を検証することに対応するブロックに含まれるプロセスの、より詳細なビューを示すフローチャートである。適用されたエンティティロールによるエンティティの設計は、設計時に、適用されたエンティティロールに対する適合性について検査される。エンティティロールの規範的な性質は、エンティティロールを基準にエンティティの妥当性を検証して（１１７）エンティティがロールを果たすかどうかを判定する（１１８）モデルバリデーションエンジン（Ｍｏｄｅｌ Ｖａｌｉｄａｔｉｏｎ Ｅｎｇｉｎｅ）を用いることによって実装される。エンティティがロールを果たさない場合（１１９）は、エラーメッセージを開発者に伝える。適用されたエンティティロールの妥当性検査が失敗した場合は、そのモデルをコンパイルすることができない。エンティティがロールを果たす場合（１２０）は、妥当性検査が完了し（１２１）、成功となる。エンティティパターンの設計者は、エンティティパターンに関する有用情報を開発者に伝えるために、説明文を書き、情報がある付加リソースへのハイパーリンクを埋め込むことも可能である。

【００２３】

エンティティロールは、そのエンティティロールに対するリレーション（アソシエーションおよびコンポジション）を指定することが可能である。モデルの妥当性検査時に、ホストエンティティ上の具体的なリレーションシップが、多重度が正しいかどうか、および正しいエンティティロールが適用されているかどうかを検査される。リレーションシップは両方向が検査される。Ｈｅａｄｅｒ／Ｌｉｎｅパターンでは、ＨｅａｄｅｒＲｏｌｅを適用されたエンティティが、ＬｉｎｅＲｏｌｅを適用されたエンティティのコンポジションを有していなければならないが、ＬｉｎｅＲｏｌｅを有するエンティティも、ＨｅａｄｅｒＲｏｌｅを適用されたエンティティによって構成されているかどうかを検査される。

【００２４】
次に、図４ｂは、図４ａの、エンティティがロールを果たすかどうかを判定することに対応するブロックに含まれるプロセスの、より詳細なビューを示すフローチャートである。プロパティは、エンティティロールプロパティ（Ｅｎｔｉｔｙ Ｒｏｌｅ Ｐｒｏｐｅｒｔｙ）と具体的なエンティティプロパティ（Ｅｎｔｉｔｙ Ｐｒｏｐｅｒｔｙ）との間のマップの存在に関して検査される。エンティティロールプロパティ上のプロパティセット（例えば、ＡｌｌｏｗＮｕｌｌ、Ｔｙｐｅなど）との適合性も検査される。最初に、ロールのプロパティをチェックする（１４１）。次に、エンティティロールのこれらのプロパティが実際に、具体的なエンティティプロパティと一致するかどうかを判定する。一致した場合は検査が完了する。一致しない場合は、エラーメッセージを、例えば、開発者に伝えることが可能である。

【００２５】
プログラミングモデル
すべてのエンティティロールは、定義済みのプロパティおよびリレーションを有する対応クラスを有する。これにより、開発者は、より高いエンティティロールレベルでエンティティデータを扱うことが可能である。エンティティロール上に書かれたコードは、自身の定義済みプロパティと、関連するエンティティロール上のプロパティとを直接扱うか、適用されたプロパティパターンを介して間接的に扱うことが可能である。ただし、エンティティパターンの設計と妥当性検査は、本明細書に記載の特定のプログラミングモデルがなくても行うことが可能である。

【００２６】
エンティティロールとインターフェースとに関しては、インターフェースがコード実装を隠すのに用いられるのに対して、エンティティロールは、別のデータのアプリケーションを隠す。エンティティロールを用いて、「マーカインターフェース（Ｍａｒｋｅｒ Ｉｎｔｅｒｆａｃｅ）」パターンを適用することが可能である。また、エンティティロールは、（多重）継承と非常によく似ているところがある。普通の継承を用いて解決可能なシナリオもあるが、エンティティロールは、さらに、多重継承のシナリオを扱い、エンティティロールと、ベースとなるエンティティとの間にある、特別なマッピングレイヤを有する。多重継承とは異なり、既にコンパイルされているエンティティにエンティティロールを宣言的に適用することが可能である。

【００２７】
エンティティロールは、具体的なアプリケーションから発せられるイベントを定期予約することが可能である。例えば、３つのプロパティ（Ｆｉｒｓｔｎａｍｅ、Ｌａｓｔｎａｍｅ、およびＦｕｌｌｎａｍｅ）を定義した顧客エンティティロールは、ＦｉｒｓｔｎａｍｅおよびＬａｓｔｎａｍｅに対する、変更されたイベントを定期予約することが可能である。定期予約しているイベントがホストエンティティ上で発せられると、エンティティロールクラス１２３に書かれているコードが実行される。

【００２８】
所与のエンティティロールが適用されているエンティティタイプについて、照会を行うことが可能である。例えば、リソースエンティティロール（Ｒｅｓｏｕｒｃｅ Ｅｎｔｉｔｙ Ｒｏｌｅ）が適用されているすべてのエンティティについて照会を行うことが可能である。これは、適用されたエンティティロールを実行時に見つけることを考慮したものである。適用されたロールのリストについて、あるいは、特定のロールの存在について、エンティティのインスタンスに照会することが可能である。このことは、ＣＬＲインターフェースに対するプログラミングモデルとよく似たプログラミングモデルを与える。

【００２９】
一見すると、エンティティ上の適用されたエンティティロールを、ＣＬＲインターフェースを用いて表しそうであるが、これにはエンティティの再コンパイルが必要であり、コードは以下のようになるであろう。これは、カスタマイズのシナリオのためには好ましくない。

【００３０】
【表１】

```
Customer customer = Customer.Create();

if (customer is IParty)
{
        // Do IParty stuff
}
```

【００３１】
その代わりに、以下のコードで表されるように、メソッドを介してエンティティに照会できるプログラミングモデルを実装する。
【００３２】
【表２】

```
Customer customer = Customer.Create();

if (customer.IsEntityRole<PartyEntityRole>())
{
        // Do Party stuff
}
```

【００３３】
インターフェースの場合のａｓキーワードと同様に、以下のコードで表されるように、特定のエンティティロールのインスタンスについてエンティティに照会することが可能である。
【００３４】
【表３】

```
Customer customer = Customer.Create();

PartyEntityRole party = customer.GetEntityRole<PartyEntityRole>();

if (party != null )
{
        // Do Party
}
```

【００３５】
エンティティロールは複数回、適用することが可能であるため（各回ごとにマッピングが異なる）、特定のエンティティロールクラスのリストについてエンティティに照会することのサポートは、以下に示すようにサポートされる。
【００３６】

【表４】

```
Customer customer = Customer.Create();

PartyEntityRole[] parties =
        customer.GetEntityRoles<PartyEntityRole>();

foreach (PartyEntityRole party in parties)
{
        // Go Party
}
```

【００３７】

エンティティに対するすべてのエンティティロールのリストを取得することも、以下に示すようにサポートされる。

【００３８】

【表５】

```
Customer customer = Customer.Create();

EntityRole[] roles = customer.GetEntityRoles();

foreach (EntityRole role in roles)
{
        // Do stuff
}
```

【００３９】

生成コードと実行時フレームワーク

次に、図５は、各エンティティパターン１１３に対する生成コードのアーキテクチャを示すブロック図である。各エンティティパターン１１３内の各エンティティロール１２２に対し、２つの対応クラスが、フレームワークによって生成される。一方のクラスは、実行時プログラミングインターフェース１２３であって、これは、ロール（Ｒｏｌｅ）によって記述される構造を有し、同じパターン１１３の他のロール１２５へのナビゲート１２４を可能にする。もう一方のクラス１２６は、各ロールに関連付けられた（１２７）、実行時に使用可能なメタデータを含む。ある特定の実施形態においてフレームワーク内でクラスがどのように生成されるかについては前述のとおりであるが、メタデータクラス１２６は、（クラス１２３の一部としてではあるが）別の方法で実装可能である。さらに、前述のように、エンティティは、適用されたロールについて実行時に照会されることが可能であり、返されたロールクラスのインスタンスを有する。

【００４０】

例えば、クラス（＜ＥｎｔｉｔｙＲｏｌｅＮａｍｅ＞）は、メタデータに格納されているデコレーションの値にアクセスするために、定義済みのプロパティ、アソシエーション、およびコンポジションと、プロパティを含むクラス（＜ＥｎｔｉｔｙＲｏｌｅＮａｍｅ＞Ｉｎｆｏ）とをエクスポーズする。＜ＥｎｔｉｔｙＲｏｌｅＮａｍｅ＞Ｉｎｆｏクラスは、デコレーション値と、エンティティロールプロパティ、アソシエーション、およびコンポジションとそれぞれのエンティティ側の相手との間のマッピングのためのデータとで、メタデータサービスから埋められる実行時クラスである。エンティティロールクラス上で指定されるすべてのデコレーションについて、そのデコレーションに関連付けられたメタデータ情報にアクセスするために、ＣＬＲプロパティが生成される。

【００４１】

すべてのプロパティ、アソシエーション、およびコンポジションについて、接尾辞「M

ａｐ」を有する文字列型ＣＬＲプロパティがクラス内に生成される。これらの文字列プロパティは、ホストエンティティクラス上のターゲットプロパティを反映するために、フレームワークによって生成される。ＳｕｍＦｏｒｍａｔＳｔｒｉｎｇデコレーション、Ｓｕｍプロパティ、およびＬｉｎｅｓコンポジションを定義されたＨｅａｄｅｒエンティティロールのｉｎｆｏクラスに対して生成されるコードの例を以下に示す。

【００４２】

【表６】

```
//------------------------------------------------------------------------------
// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version:2.0.40607.42
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </autogenerated>
//------------------------------------------------------------------------------

namespace MyModel {
    using System;
    using System.BusinessFramework;
    using System.BusinessFramework.Entities;


    public class HeaderRoleInfo : EntityRoleInfo {

        private string sumFormatString;

        private string sum;

        private string lines;
```

【００４３】

【表７】

```
    public string SumFormatString {
      get {
        return this.sumFormatString;
      }
      set {
        this.sumFormatString = value;
      }
    }

    public string Sum {
      get {
        return this.sum;
      }
      set {
        this.sum = value;
      }
    }

    public string Lines {
      get {
        return this.lines;
      }
      set {
        this.lines = value;
      }
    }

    public override System.Type GetEntityRoleClass() {
      return typeof(HeaderRole);
    }
  }
}
```

【００４４】

定義済みのプロパティ、アソシエーション、およびコンポジションをエクスポーズする＜ＥｎｔｉｔｙＲｏｌｅＮａｍｅ＞クラスに関し、ＳｕｍＦｏｒｍａｔＳｔｒｉｎｇデコレーション、Ｓｕｍプロパティ、Ｌｉｎｅｓコンポジションを定義されたＨｅａｄｅｒエンティティロールの実行時クラスについて生成されたコードの例を以下に示す。

【００４５】

【表８】

```
//------------------------------------------------------------------------------
// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version:2.0.40607.42
```

【００４６】

【表９】

```
//
//    Changes to this file may cause incorrect behavior and will be lost if
//    the code is regenerated.
// </autogenerated>
//------------------------------------------------------------------------------


namespace MyModel {
   using System;
   using System.BusinessFramework;
   using System.BusinessFramework.Entities;


   public partial class HeaderRole : EntityRole {

      public HeaderRole(EntityRoleInfo info, Entity entity) :
            base(info, entity) {
      }

      public Decimal Sum {
         get {
            return ((Decimal)(GetEntityRoleHost().GetProperty(GetInfo().Sum).Value));
         }
         set {
            GetEntityRoleHost().GetProperty(GetInfo().Sum).Value = value;
         }
      }

      public EntityRoleCollection<LineRole> Lines {
         get {
            object value =
GetEntityRoleHost().GetType().GetProperty(GetInfo().Lines).GetValue(GetEntityRoleHost(), null);
            if ((value == null)) {
               return null;
            }
            else {
               return new EntityRoleCollection<LineRole>(((EntityCollection)(value)));
            }
         }
      }

      protected new HeaderRoleInfo GetInfo() {
         return ((HeaderRoleInfo)(base.GetInfo()));
      }
   }
}
```

【００４７】

実行時フレームワークは、３つのメソッド、エンティティクラス、およびいくつかのベースクラスとともに実装される。実行時フレームワークは、軽量であり、エンティティクラス上にメモリフットプリントを持たない。また、エンティティロールがイベントを定期予約している場合には、エンティティごとにエンティティロールのインスタンスを生成することが可能である。例えば、ＥｎｔｉｔｙＲｏｌｅクラスは、適用されたエンティティロールの実行時表現のベースクラスである。対応するＥｎｔｉｔｙＲｏｌｅＩｎｆｏオブジェクトおよびホストエンティティを取得するのは、２つのメソッドを有するアブストラクトである。ＥｎｔｉｔｙＲｏｌｅクラスについての例示的なコードリストを以下に示す。

【００４８】
【表１０】

```
namespace System.BusinessFramework.Entities
{
  public abstract class EntityRole
  {
    public EntityRole(System.BusinessFramework.Entities.EntityRoleInfo info,
System.BusinessFramework.Entities.Entity entity);
    public virtual System.BusinessFramework.Entities.Entity GetEntityRoleHost();
    public System.BusinessFramework.Entities.EntityRoleInfo GetInfo();
  }
}
```

【００４９】
例えば、ＥｎｔｉｔｙＲｏｌｅＩｎｆｏは、実行時メタデータ情報クラスのベースクラスであって、メタデータサービスから抽出された、マッピングに関するキャッシュ情報を含む。ＥｎｔｉｔｙＲｏｌｅＩｎｆｏクラスについての例示的なコードリストを以下に示す。
【００５０】
【表１１】

```
namespace System.BusinessFramework.Entities
{
  public abstract class EntityRoleInfo
  {
    protected EntityRoleInfo();
    public abstract System.Type GetEntityRoleClass();
  }
}
```

【００５１】
ＥｎｔｉｔｙＲｏｌｅＣｏｌｌｅｃｔｉｏｎクラスは、ＥｎｔｉｔｙＣｏｌｌｅｃｔｉｏｎｓをラップして、ＥｎｔｉｔｙＲｏｌｅレベルでのＥｎｔｉｔｙＣｏｌｌｅｃｔｉｏｎｓへのアクセスを可能にし、ＥｎｔｉｔｙＰａｔｔｅｒｎＭａｎａｇｅｒクラスは、適用されたエンティティについてメタデータサービスに照会し、必要であればインスタンスを生成する。ＥｎｔｉｔｙＰａｔｔｅｒｎＭａｎａｇｅｒクラスは、メタデータサービスに接続し、適用されたすべてのエンティティロールについて、必要なマッピング情報を有する対応ＥｎｔｉｔｙＲｏｌｅＩｎｆｏクラスと、対応ＥｎｔｉｔｙＲｏｌｅクラスとをハイドレート（ｈｙｄｒａｔｅ）する。また、エンティティベースクラスは、エンティティに適用されたエンティティロールにアクセスするメソッドを有する。ＥｎｔｉｔｙＲｏｌｅＩｎｆｏクラスが所定のレートでのみインスタンス生成されるように、キャッシュを用いることが可能である。
【００５２】
エンティティパターンに基づいて自動生成されるＵＩ
エンティティパターンがあることにより、ユーザインターフェース（ＵＩ）を自動生成する際に活用可能な重要情報が保持される。エンティティロールの一部としてモデル化されるプロパティおよびリレーションは、多くの場合、エンティティの重要な要素であり、ＵＩにおける支配的な地位を与えられることが可能である。
【００５３】
使用可能なスペースが限られている状態で、Ｃｕｓｔｏｍｅｒエンティティを、適用されたＰａｒｔｙＲｏｌｅとともにＵＩに表示する場合は、自動生成メカニズムがＰａｒｔ

ｙＲｏｌｅの存在を用いて、表示するプロパティを選択することが可能である。

【００５４】

ＵＩレイアウトをエンティティパターンにリンクして、具体的なエンティティに適用されるエンティティロールを検査することによって、一貫性のあるＵＩを具体的なエンティティに与えることが可能である。

【００５５】

例示的コンピューティング環境およびネットワーク環境

図６は、本発明の各種態様と組み合わせての使用に好適な例示的コンピューティング装置を表すブロック図である。例えば、ツリービュー内でのインラインプロパティ編集のプロセスおよびメソッドを実行する、コンピュータで実行可能な命令は、図６に示されるようなコンピューティング環境での常駐および／または実行が可能である。コンピューティングシステム環境２２０は、好適なコンピューティング環境の一例に過ぎず、本発明の使用または機能性の範囲を何ら限定するものではない。同様に、コンピューティング環境２２０は、例示的運用環境２２０に示されたコンポーネントの任意の１つまたは組合せに関連する何らかの依存性または要件を有するものとして解釈されてはならない。

【００５６】

本発明の態様は、他の様々な、汎用または専用のコンピューティングシステム環境または構成で実用可能である。本発明との使用に好適であると考えられる、よく知られたコンピューティングシステム、環境、および／または構成の例として、パーソナルコンピュータ、サーバコンピュータ、ハンドヘルドまたはラップトップ機器、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラム可能な家庭用電子機器、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、およびこれらのシステムまたは機器を任意に含む分散コンピューティング環境などがあり、これらに限定されない。

【００５７】

本発明の態様は、コンピュータによって実行される、コンピュータで実行可能な命令（プログラムモジュールなど）の一般的文脈で実施可能である。一般に、プログラムモジュールは、特定のタスクを実行したり、特定の抽象データ型を実装したりするルーチン、プログラム、オブジェクト、コンポーネント、データ構造体などを含む。また、本発明の態様は、通信ネットワークでリンクされたリモート処理装置でタスクを実行する分散コンピューティング環境で実施されることも可能である。分散コンピューティング環境では、ローカルとリモートの両方のコンピュータ記憶媒体（メモリ記憶装置を含む）にプログラムモジュールを配置することが可能である。

【００５８】

本発明の態様を実装する例示的システムは、コンピュータ２４１の形の汎用コンピューティング装置を含む。コンピュータ２４１のコンポーネントとして、処理ユニット２５９、システムメモリ２２２、およびシステムメモリを含む各種システムコンポーネントを処理ユニット２５９に接続するシステムバス２２１などがあるが、これらに限定されない。システムバス２２１は、メモリバスまたはメモリコントローラ、ペリフェラルバス、および様々なバスアーキテクチャを任意に用いるローカルバスを含む、任意の様々なタイプのバス構造であることが可能である。そのようなアーキテクチャとして、例えば、ＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＭＣＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、およびＭｅｚｚａｎｉｎｅバスとも呼ばれるＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バスなどがあり、これらに限定されない。

【００５９】

コンピュータ２４１は、一般に、様々なコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピュータ２４１からのアクセスが可能な任意の市販媒体であってよく、揮

発性および不揮発性媒体、リムーバブルおよび非リムーバブル媒体を含む。コンピュータ可読媒体として、例えば、コンピュータ記憶媒体および通信媒体があり、これらに限定されない。コンピュータ記憶媒体は、コンピュータ可読命令、データ構造体、プログラムモジュール、その他のデータなどの情報を記憶する任意の方法または技術に実装される、揮発性および不揮発性媒体の両方と、リムーバブルおよび非リムーバブル媒体の両方とを含む。コンピュータ記憶媒体として、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリ、または他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）、または他の光ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置、または他の磁気記憶装置、あるいは他の任意の、所望の情報を格納するために使用可能であってコンピュータ２４１からのアクセスが可能な媒体などがあり、これらに限定されない。通信媒体は、一般に、コンピュータ可読命令、データ構造体、プログラムモジュール、または他のデータを、搬送波などの変調データ信号や他の搬送メカニズムの形で具体化し、任意の情報配信媒体を含む。用語「変調データ信号」は、その信号の１つまたは複数の特性が、信号内の情報をエンコードするように設定または変更される信号を意味する。通信媒体として、有線ネットワークや直接有線接続などの有線媒体、および音響、ＲＦ、赤外線、その他の無線媒体があり、これらに限定されない。以上のものの任意の組合せも、コンピュータ可読媒体の範囲に含まれるものとする。

【００６０】

システムメモリ２２２は、読み出し専用メモリ（ＲＯＭ）２２３およびランダムアクセスメモリ（ＲＡＭ）２６０などの揮発性および／または不揮発性メモリの形でコンピュータ記憶媒体を含む。スタートアップ時などにコンピュータ２４１内の要素間情報転送を支援する基本ルーチンを含む基本入出力システム２２４（ＢＩＯＳ）は、一般に、ＲＯＭ２２３に格納される。ＲＡＭ　２６０は、一般に、処理ユニット２５９からのアクセスがただちに可能になるデータおよび／またはプログラムモジュール、および／または処理ユニット２５９によって現在稼働中であるデータおよび／またはプログラムモジュールを含む。限定ではなく例として、図６は、オペレーティングシステム２２５、アプリケーションプログラム２２６、他のプログラムモジュール２２７、およびプログラムデータ２２８を含む。

【００６１】

さらに、コンピュータ２４１は、他のリムーバブル／非リムーバブル、揮発性／不揮発性のコンピュータ記憶媒体も含む。あくまで例であるが、図６は、非リムーバブルで不揮発性の磁気媒体に対して読み出しまたは書き込みを行うハードディスクドライブ２３８と、リムーバブルで不揮発性の磁気ディスク２５４に対して読み出しまたは書き込みを行う磁気ディスクドライブ２３９と、リムーバブルで不揮発性の光ディスク２５３（ＣＤ－ＲＯＭまたは他の光ディスクなど）に対して読み出しまたは書き込みを行う光ディスクドライブ２４０とを示している。この例示的運用環境で使用可能な、他のリムーバブル／非リムーバブル、揮発性／不揮発性のコンピュータ記憶媒体として、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭなどがあるが、これらに限定されない。ハードディスクドライブ２３８は、一般に、非リムーバブルのメモリインターフェース（インターフェース２３４など）を介してシステムバス２２１に接続され、磁気ディスクドライブ２３９および光ディスクドライブ２４０は、一般に、リムーバブルメモリインターフェース（インターフェース２３５など）によってシステムバス２２１に接続される。

【００６２】

前述の、図６に示された各ドライブとそれぞれに関連付けられたコンピュータ記憶媒体は、コンピュータ２４１に用いられるコンピュータ可読命令、データ構造体、プログラムモジュール、およびその他のデータの記憶領域を提供する。図６では、例えば、ハードディスクドライブ２３８は、オペレーティングシステム２５８、アプリケーションプログラム２５７、他のプログラムモジュール２５６、およびプログラムデータ２５５を格納するものとして示されている。これらのコンポーネントは、オペレーティングシステム２２５

、アプリケーションプログラム２２６、他のプログラムモジュール２２７、およびプログラムデータ２２８と同じであっても、異なっていてもよいことに注意されたい。オペレーティングシステム２５８、アプリケーションプログラム２５７、他のプログラムモジュール２５６、およびプログラムデータ２５５は、ここでは異なる番号を与えられているが、これは、少なくともこれらが異なるコピーであることを示すためである。ユーザは、キーボード２５１、ポインティングデバイス２５２（一般にマウス、トラックボール、タッチパッドなどと呼ばれる）などの入力装置を用いて、コンピュータ２４１にコマンドおよび情報を入力することが可能である。他の入力装置（図示せず）として、マイク、ジョイスティック、ゲームパッド、パラボラアンテナ、スキャナなども可能である。これらおよびその他の入力装置は、多くの場合、システムバスに接続されているユーザ入力インターフェース２３６を介して処理ユニット２５９に接続されるが、他のインターフェースやバス構造（パラレルポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）など）で接続されることも可能である。モニタ２４２または他のタイプのディスプレイ装置も、ビデオインターフェース２３２などのインターフェースを介して、システムバス２２１に接続される。コンピュータは、モニタに加えて、出力ペリフェラルインターフェース２３３を介しての接続が可能な、他のペリフェラル出力装置（スピーカ２４４、プリンタ２４３など）も含むことが可能である。

【００６３】

コンピュータ２４１は、ネットワーク環境で、リモートコンピュータ２４６など、１つまたは複数のリモートコンピュータとの論理接続を用いて稼動することも可能である。リモートコンピュータ２４６は、パーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピア装置、または他のコモンネットワークノードであることが可能であり、一般に、コンピュータ２４１に関連して前述した要素のほとんどまたはすべてを含むが、図６ではメモリ記憶装置２４７だけを示している。図６に示した論理接続は、ローカルエリアネットワーク（ＬＡＮ）２４５およびワイドエリアネットワーク（ＷＡＮ）２４９を含むが、他のネットワークを含むことも可能である。そのようなネットワーク環境は、オフィス、企業規模コンピュータネットワーク、イントラネット、インターネットなどでは、ごく一般的である。

【００６４】

コンピュータ２４１は、ＬＡＮネットワーキング環境で使用される場合、ネットワークインターフェースまたはアダプタ２３７を介してＬＡＮ　２４５に接続される。ＷＡＮネットワーキング環境で使用される場合、コンピュータ２４１は、一般に、インターネットなどのＷＡＮ　２４９との通信を確立するためにモデム２５０または他の手段を含む。モデム２５０は、内蔵であれ、外付けであれ、ユーザ入力インターフェース２３６、または他の適切なメカニズムによってシステムバス２２１に接続されることが可能である。ネットワーク環境では、コンピュータ２４１に関連して示したプログラムモジュールまたはその一部を、リモートメモリ記憶装置に格納することが可能である。限定ではなく例として、図６は、メモリ装置２４７に常駐しているリモートアプリケーションプログラム２４８を示している。図示したネットワーク接続は例示的であって、コンピュータ間の通信リンクを確立する他の手段も使用可能であることを理解されたい。

【００６５】

本明細書で説明した様々な手法は、ハードウェアまたはソフトウェア、あるいは必要に応じてそれらの組合せとともに実装可能であることを理解されたい。したがって、本発明の方法および装置、あるいは、それらの特定の態様または部分は、フロッピー（登録商標）ディスク、ＣＤ－ＲＯＭ、ハードドライブ、または他の任意のマシン可読記憶媒体のような有形の媒体において実施されるプログラムコード（すなわち、命令）の形をとることが可能であり、そのプログラムコードがコンピュータなどのマシンにロードされ、そのマシンで実行された場合に、そのマシンが、本発明を実施する装置となることを理解されたい。プログラム可能なコンピュータでプログラムコードを実行する場合、コンピューティング装置は、一般に、プロセッサ、プロセッサ可読な記憶媒体（揮発性および不揮発性の

メモリおよび／または記憶要素を含む）、少なくとも１つの入力装置、および少なくとも１つの出力装置を含む。１つまたは複数のプログラムが、例えば、ＡＰＩ、再利用可能なコントロール、その他を用いて、本発明と関連して説明されたプロセスを実施または利用することが可能である。そのようなプログラムは、コンピュータシステムと通信するために、高レベル手続き言語、またはオブジェクト指向プログラミング言語で実装されることが好ましい。ただし、必要であれば、プログラムをアセンブリ言語やマシン言語で実装することも可能である。どのような場合でも、言語は、コンパイルまたは解釈された言語であることが可能であり、ハードウェア実装と組み合わせられることが可能である。

【００６６】

例示的実施形態は、１つまたは複数のスタンドアロンコンピュータシステムの文脈で本発明の態様を利用することを示しているが、本発明はそのようには限定されず、ネットワーク（または分散）コンピューティング環境など、任意のコンピューティング環境とともに実装されることが可能である。さらに、本発明の態様は、複数の処理チップまたは処理装置に（またはその全体に）実装されることが可能であり、記憶も同様に、複数の装置の全体で行われることが可能である。そのような装置として、パーソナルコンピュータ、ネットワークサーバ、ハンドヘルド装置、スーパーコンピュータ、または他のシステム（自動車や飛行機など）に組み込まれたコンピュータが考えられる。

【００６７】

例示的なネットワークコンピューティング環境を図７に示す。当業者であれば、ネットワークは、任意のコンピュータまたは他のクライアントまたはサーバ装置、あるいは分散コンピューティング環境に接続可能であることを理解されよう。この点において、任意の数の処理ユニット、メモリユニット、または記憶ユニット、および任意の数のアプリケーションおよび同時進行中のプロセスを有する任意のコンピュータシステムまたは環境が、本発明のシステムおよび方法との使用に好適であると見なされる。

【００６８】

分散コンピューティングは、コンピュータ装置およびコンピュータシステムの間での交換によってコンピュータリソースおよびサービスを共有することを可能にする。これらのリソースおよびサービスは、情報、キャッシュ記憶領域、およびファイル用ディスク記憶装置の交換を含む。分散コンピューティングは、ネットワーク接続性を利用して、クライアントがそれぞれの能力を集合させて活用することによって企業全体に資することを可能にする。この点において、様々な装置が、本明細書で説明されたプロセスを関与させうるアプリケーション、オブジェクト、またはリソースを有することが可能である。

【００６９】

図７は、例示的なネットワーク（または分散）コンピューティング環境の概略図である。この環境は、コンピューティング装置２７１、２７２、２７６、および２７７と、オブジェクト２７３、２７４、および２７５と、データベース２７８とを含む。これらのエンティティ２７１、２７２、２７３、２７４、２７５、２７６、２７７、および２７８のそれぞれは、プログラム、メソッド、データストア、プログラマブルロジックなどを含むか、利用することが可能である。エンティティ２７１、２７２、２７３、２７４、２７５、２７６、２７７、および２７８は、ＰＤＡ、オーディオ／ビデオプレーヤ、ＭＰ３プレーヤ、パーソナルコンピュータなどの同じまたは異なる装置の部分にまたがることが可能である。各エンティティ２７１、２７２、２７３、２７４、２７５、２７６、２７７、および２７８は、通信ネットワーク２７０によって、別のエンティティ２７１、２７２、２７３、２７４、２７５、２７６、２７７、および２７８と通信することが可能である。この点において、データベース２７８または他の記憶要素の保守および更新を、任意のエンティティが担当することが可能である。

【００７０】

このネットワーク２７０は、それ自体で、図７のシステムにサービスを提供するコンピューティングエンティティを含むことが可能であり、それ自体で、相互接続された複数のネットワークを表すことが可能である。本発明の態様によれば、各エンティティ２７１、

２７２、２７３、２７４、２７５、２７６、２７７、および２７８は、他のエンティティ２７１、２７２、２７３、２７４、２７５、２７６、２７７、および２７８のうちの１つまたは複数のエンティティのサービスを要求するためにＡＰＩ、または他のオブジェクト、ソフトウェア、ファームウェアおよび／またはハードウェアを利用することが可能な離散的機能プログラムモジュールを含むことが可能である。

【００７１】

２７５などのオブジェクトが別のコンピューティング装置２７６でホストされることが可能であることも理解されよう。したがって、図示された物理環境では、接続された装置がコンピュータとして示されていても、そのような図は例示的に過ぎず、代替として、ＰＤＡ、テレビ、ＭＰ３プレーヤなどの各種デジタル装置や、インターフェース、ＣＯＭオブジェクトなどのソフトウェアオブジェクトを含むように物理環境を図示または説明することも可能である。

【００７２】

分散コンピューティング環境をサポートするシステム、コンポーネント、およびネットワーク構成は様々にある。例えば、複数のコンピューティングシステムを、有線または無線システムによって、あるいは、ローカルネットワークや広域分散ネットワークによって、まとめて接続することが可能である。現在、多数のネットワークがインターネットに接続されていて、これらが、広域分散コンピューティングのインフラストラクチャを提供し、多数の異なるネットワークを包括している。任意のそのようなインフラストラクチャは、インターネットに接続されていてもいなくても、本発明のシステムおよび方法と組み合わせて用いることが可能である。

【００７３】

ネットワークインフラストラクチャは、クライアント／サーバ、ピアツーピア、ハイブリッドアーキテクチャなどのネットワークトポロジのホストを有効にすることが可能である。「クライアント」は、関連がない別のクラスまたはグループのサービスを使用する、クラスまたはグループのメンバである。コンピューティングにおいては、クライアントは、別のプログラムによって提供されるサービスを要求するプロセス（すなわち、大まかには、命令またはタスクのセット）である。クライアントプロセスは、他のプログラムまたはサービス自体の動作詳細をまったく「認識」せずに、要求したサービスを利用する。クライアント／サーバアーキテクチャ（特にネットワークシステム）では、通常、クライアントは、別のコンピュータ（例えば、サーバ）によって提供される共有ネットワークリソースにアクセスするコンピュータである。図７の例では、任意のエンティティ２７１、２７２、２７３、２７４、２７５、２７６、２７７、および２７８を、状況に応じて、クライアント、サーバ、または両方と見なすことが可能である。

【００７４】

サーバは、（必ずしもそうとは限らないが）一般に、インターネットなどのリモートまたはローカルのネットワークからのアクセスが可能なリモートコンピュータシステムである。クライアントプロセスは第１のコンピュータシステムでアクティブであることが可能であり、サーバプロセスは第２のコンピュータシステムでアクティブであることが可能であって、これらは通信媒体を介して互いに通信していることが可能である。これによって、機能の分散が実現され、複数のクライアントがサーバの情報収集機能を利用することが可能になる。ソフトウェアオブジェクトは、複数のコンピューティング装置またはオブジェクトにわたって分散可能である。

【００７５】

クライアントとサーバは、互いに通信して、プロトコルレイヤで提供される機能を利用する。例えば、ハイパーテキスト転送プロトコル（ＨＴＴＰ）は、ワールドワイドウェブ（ＷＷＷ）（「Ｗｅｂ」）とともに用いられる共通プロトコルである。一般に、インターネットプロトコル（ＩＰ）アドレスなどのコンピュータネットワークアドレスや、ユニバーサルリソースロケータ（ＵＲＬ）などの他のリファレンスは、サーバコンピュータまたはクライアントコンピュータが互いを識別するために使用可能である。ネットワークアド

レスを、ＵＲＬと呼ぶことも可能である。通信は、通信媒体を介して提供可能である。例えば、クライアントとサーバは、大容量通信のために、ＴＣＰ／ＩＰ接続を介して互いに接続されることが可能である。

【００７６】

図６に示された一般的フレームワークに従って構築可能なコンピューティング環境が多種多様であることと、図７に示されるようなネットワーク環境でのコンピューティングにおいてはさらなる多様化が可能であることとを考慮すると、本明細書で提供されるシステムおよび方法は、何らかの形で特定のコンピューティングアーキテクチャに限定されるようには解釈されるべきではない。むしろ、本発明は、１つの実施形態に限定されるべきでなく、添付の特許請求項に従う広さおよび範囲において解釈されるべきである。

【００７７】

結論

本明細書で説明した各種のシステム、方法、および手法は、ハードウェアまたはソフトウェア、あるいは必要に応じてそれらの組合せによって実装されることが可能である。したがって、本発明の方法および装置、あるいは、それらの特定の態様または部分は、フロッピー（登録商標）ディスク、ＣＤ－ＲＯＭ、ハードドライブ、または他の任意のマシン可読記憶媒体のような有形の媒体において実施されるプログラムコード（すなわち、命令）の形をとることが可能であり、そのプログラムコードがコンピュータなどのマシンにロードされ、そのマシンで実行された場合に、そのマシンが、本発明を実施する装置となることを理解されたい。プログラム可能なコンピュータでプログラムコードを実行する場合、コンピュータは、一般に、プロセッサ、プロセッサ可読な記憶媒体（揮発性および不揮発性のメモリおよび／または記憶要素を含む）、少なくとも１つの入力装置、および少なくとも１つの出力装置を含む。コンピュータシステムと通信するために、１つまたは複数のプログラムが、高レベル手続き言語、またはオブジェクト指向プログラミング言語で実装されることが好ましい。ただし、必要であれば、プログラムをアセンブリ言語やマシン言語で実装することも可能である。どのような場合でも、言語は、コンパイルまたは解釈された言語であることが可能であり、ハードウェア実装と組み合わせられることが可能である。

【００７８】

本発明の方法および装置は、何らかの伝送媒体（電気的ワイヤリングまたはケーブリング、光ファイバ、または他の任意の伝送形態）によって伝送されるプログラムコードの形で具体化されることも可能であり、このプログラムコードが、ＥＰＲＯＭ、ゲートアレイ、プログラマブルロジックデバイス（ＰＬＤ）、クライアントコンピュータ、ビデオレコーダなどのマシンで受信およびロードされ、実行されると、そのマシンが、本発明を実施する装置になる。プログラムコードは、汎用プロセッサに実装された場合には、そのプロセッサと結合して、本発明のインデックス作成機能を実施するよう動作する、固有の装置を提供する。

【００７９】

様々な図面を用いて好ましい実施形態を示しながら本発明について説明してきたが、本発明から逸脱することなく本発明と同じ機能を実施するために、他の同様な実施形態も使用可能であること、あるいは、説明した実施形態に変更や追加を行うことが可能であることを理解されたい。例えば、パーソナルコンピュータの機能性をエミュレートするデジタル装置の文脈で本発明の例示的実施形態を説明したが、当業者であれば、本発明が、そのようなデジタル装置に限定されないこと、本明細書で説明したように、任意の数の既存の、または今後現れるコンピューティング装置またはコンピューティング環境（ゲームコンソール、ハンドヘルドコンピュータ、ポータブルコンピュータなど（有線か無線かを問わず））に適用可能であること、および、通信ネットワークを介して接続され、ネットワーク経由で対話する、任意の数のそのようなコンピューティング装置に適用可能であることを理解されよう。さらに、特に、無線ネットワーク装置の数が急増し続けていることから、本明細書では、ハンドヘルド装置オペレーティングシステムおよび他のアプリケーショ

ン固有のハードウェア／ソフトウェアインターフェースシステムを含む様々なコンピュータプラットフォームを考慮していることを強調しておきたい。したがって、本発明は、１つの実施形態に限定されるべきでなく、添付の特許請求項に従う広さおよび範囲において解釈されるべきである。

【００８０】

最後に、本明細書で説明した、開示の実施形態は、他のプロセッサアーキテクチャ、コンピュータベースのシステム、またはシステム仮想化で使用されるように適合されることが可能であり、そのような実施形態は、本明細書での開示から明確に予想されるため、本発明は、本明細書で説明した特定の実施形態に限定されるのではなく、最大限に広く解釈されるべきである。同様に、プロセッサ仮想化以外の目的での合成命令（ｓｙｎｔｈｅｔｉｃ　ｉｎｓｔｒｕｃｔｉｏｎ）の使用も、本明細書での開示から予想され、プロセッサ仮想化以外の文脈での合成命令のそのような任意の利用は、本明細書での開示に最大限広く読み込まれなければならない。

【００８１】

付録Ａ：定義と用語

【００８２】

【表１２】

REA	リソース、イベント、およびエージェント－ビジネス領域に対する確立されたオントロジー。
エンティティパターン	１つまたは複数のエンティティロールからなる概念パターン。本明細書で説明したHeader-Lineパターンは、エンティティパターンの例である。
エンティティロールクラス	他のエンティティロールに対する、必須のプロパティおよびリレーションを記述する。Header-Lineパターンの場合は、Lineエンティティロールが適用されたエンティティに対するコンポジションを要求するHeaderエンティティロールからなる。エンティティロールは、多くの用途では、エンティティロールクラスと区別なく用いられる。
エンティティロール	適用されたエンティティロールクラスのインスタンス。
BED	ビジネスエンティティデザイナ－エンティティの設計に用いられるビジュアル設計サーフェス。
ホストエンティティ	エンティティロールの適用先のエンティティ。
エンティティタイプカテゴリ（Entity Type Category）	エンティティロールに対して用いられていたオリジナルの（前の）用語。
プロパティパターン	PARAMETERIZED AND REUSABLE IMPLEMENTATIONS OF BUSINESS LOGIC PATTERNS特許のMBF実装。

【図面の簡単な説明】

【００８３】

【図１】本発明による、エンティティパターンを作成および使用するプロセスの高レベルビューのフローチャートである。

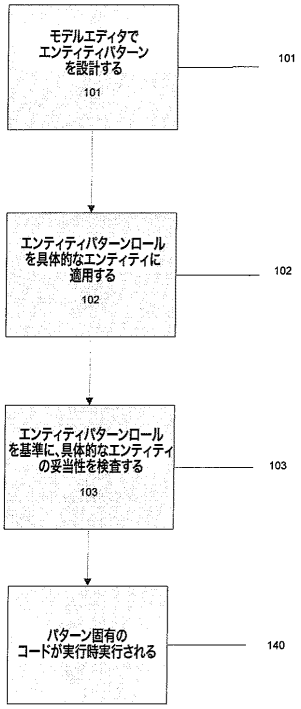
【図２】本発明の可能な実施形態による、モデルエディタ（Ｍｏｄｅｌ　Ｅｄｉｔｏｒ）に表示された例示的エンティティパターンを示すサンプルスクリーンショットである。

【図３】本発明の可能な実施形態による、ビジネスエンティティデザイナ（Ｂｕｓｉｎｅｓｓ　Ｅｎｔｉｔｙ　Ｄｅｓｉｇｎｅｒ）による、エンティティロールの適用の例を示すサンプルスクリーンショットである。

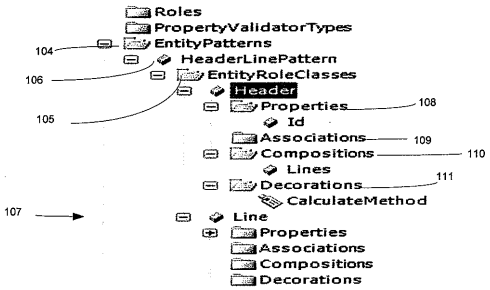
【図４ａ】本発明による、図１の、エンティティパターンロールを基準にして具体的なエ

ンティティの妥当性を検証することに対応するブロックに含まれるプロセスの、より詳細なビューを示すフローチャートである。
【図４ｂ】本発明による、図４ａの、エンティティがロールを果たすかどうかを判定することに対応するブロックに含まれるプロセスの、より詳細なビューを示すフローチャートである。
【図５】可能な実施形態における、各エンティティパターンに対する生成コードのアーキテクチャを示すブロック図である。
【図６】本発明の各種態様と組み合わせての使用に好適な例示的コンピューティング装置を表すブロック図である。
【図７】多くのコンピュータ化されたプロセスを実施することが可能な例示的ネットワークコンピューティング環境を示す図である。
【符号の説明】
【００８４】
２２０　コンピューティング環境
２２１　システムバス
２２２　システムメモリ
２２５　オペレーティングシステム
２２６　アプリケーションプログラム
２２７　他のプログラムモジュール
２２８　プログラムデータ
２３０　ビデオメモリ
２３１　グラフィックスインターフェース
２３２　ビデオインターフェース
２３３　出力ペリフェラルインターフェース
２３４　非リムーバブル不揮発性メモリインターフェース
２３５　リムーバブル不揮発性メモリインターフェース
２３６　ユーザ入力インターフェース
２３７　ネットワークインターフェース
２４２　モニタ
２４３　プリンタ
２４４　スピーカ
２４５　ローカルエリアネットワーク
２４６　リモートコンピュータ
２４８　リモートアプリケーションプログラム
２４９　ワイドエリアネットワーク
２５０　モデム
２５１　キーボード
２５２　ポインティングデバイス
２５５　プログラムデータ
２５６　他のプログラムモジュール
２５７　アプリケーションプログラム
２５８　オペレーティングシステム
２５９　処理ユニット
２７０　通信ネットワーク／バス
２７１　コンピューティング装置
２７２　コンピューティング装置
２７３　オブジェクト
２７６　コンピューティング装置
２７７　コンピューティング装置
２７８　データベース

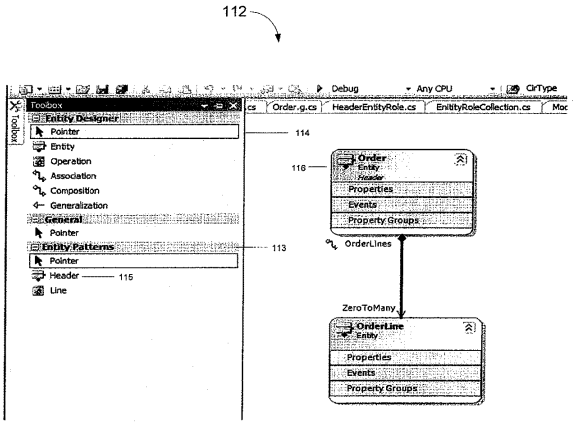
【図１】



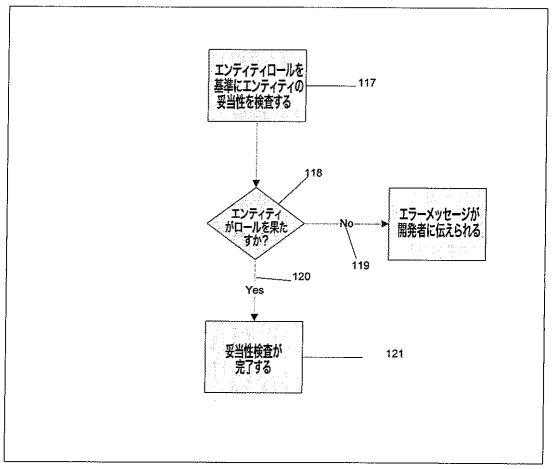
【図２】



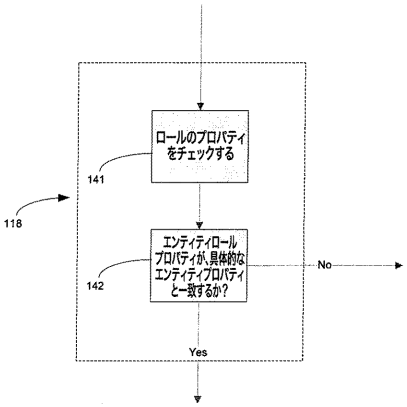
【図３】



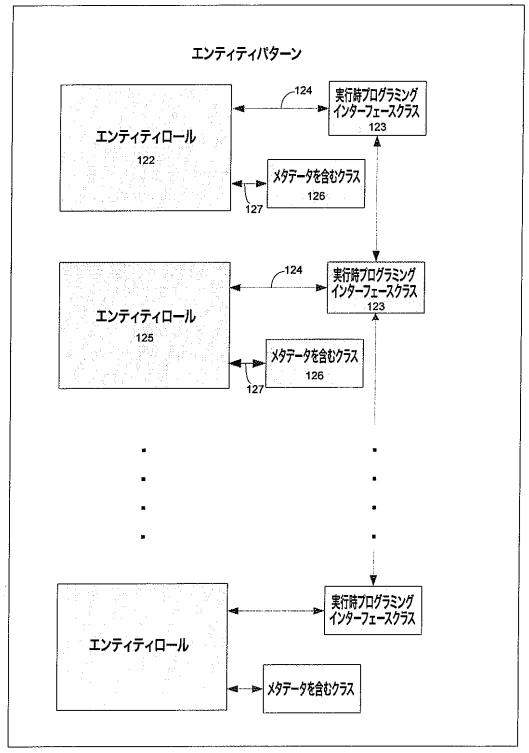
【図４ａ】



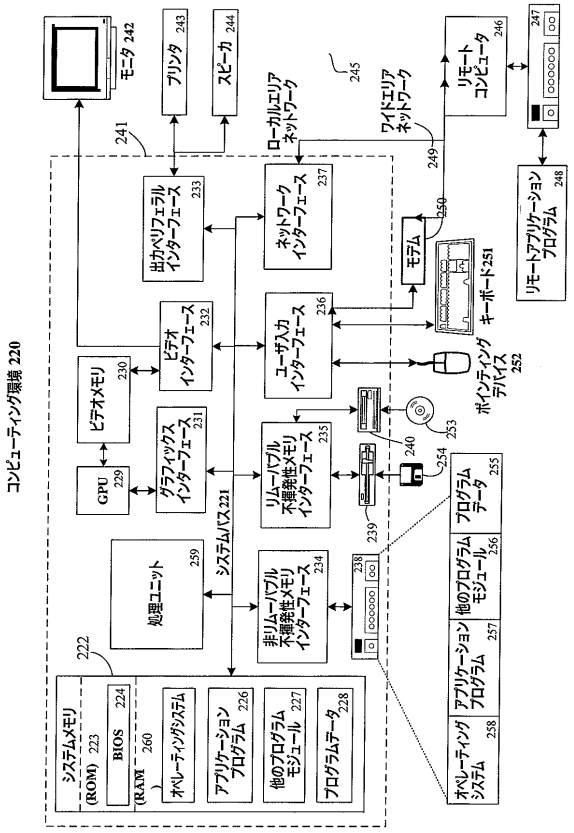
【図４ｂ】



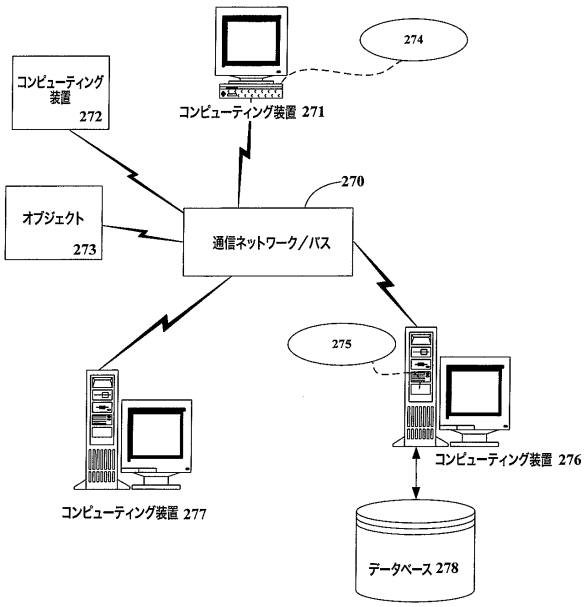
【図５】


エンティティパターン
エンティティロール
122
124
実行時プログラミング
インターフェースクラス
123
メタデータを含むクラス
126
127
エンティティロール
125
124
実行時プログラミング
インターフェースクラス
123
メタデータを含むクラス
126
127
エンティティロール
実行時プログラミング
インターフェースクラス
メタデータを含むクラス

【図６】


コンピューティング環境 220
モニタ 242
プリンタ 243
スピーカ 244
ローカルエリア
ネットワーク
245
ワイドエリア
ネットワーク
リモート
コンピュータ 246
247
リモートアプリケーション
プログラム 248
249
241
出力ペリフェラル
インターフェース 233
ネットワーク
インターフェース 237
モデム 250
キーボード 251
ポインティング
デバイス 252
ビデオ
インターフェース 232
ユーザ入力
インターフェース 236
ビデオメモリ 230
GPU 229
グラフィックス
インターフェース 231
リムーバブル
不揮発性メモリ
インターフェース 235
253
240
254
239
システムバス 221
処理ユニット 259
非リムーバブル
不揮発性メモリ
インターフェース 234
238
プログラム
データ 255
他のプログラム
モジュール 256
アプリケーション
プログラム 257
オペレーティング
システム 258
222
システムメモリ
(ROM) 223
BIOS 224
(RAM) 260
オペレーティングシステム
アプリケーション
プログラム 226
他のプログラム
モジュール 227
プログラムデータ 228

【図７】


274
コンピューティング
装置
272
コンピューティング装置 271
270
オブジェクト
273
通信ネットワーク／バス
275
コンピューティング装置 276
コンピューティング装置 277
データベース 278

フロントページの続き

(72)発明者　ラース　ハマー
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　マイケル　リッダーショルム　ピーターセン
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　ハインリッヒ　ホフマン　クラウゼン
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　トーマス　ヘルスバーグ
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

審査官　川崎　優

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ　　９／４４