(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5672134号
(P5672134)
(45)発行日　平成27年2月18日(2015.2.18)　(24)登録日　平成27年1月9日(2015.1.9)

(51)Int.Cl.　　F I
G06F 9/44 (2006.01)　　G06F 9/06 620A
G06F 3/048 (2013.01)　　G06F 3/048 654A
　　G06F 3/048 653A

請求項の数 10　(全 32 頁)

(21)出願番号　特願2011-101835 (P2011-101835)
(22)出願日　平成23年4月28日 (2011.4.28)
(65)公開番号　特開2011-238223 (P2011-238223A)
(43)公開日　平成23年11月24日 (2011.11.24)
審査請求日　平成26年2月4日 (2014.2.4)
(31)優先権主張番号　10161690.2
(32)優先日　平成22年4月30日 (2010.4.30)
(33)優先権主張国　欧州特許庁 (EP)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100146776
弁理士　山口　昭則
(72)発明者　メンデイ　ロジャー
イギリス国，ジーユー１　２エイチエイチ，ギルフォード　サリー，ウォレン　ロード　６４番，ザ　スタジオ

審査官　塚田　肇

最終頁に続く

(54)【発明の名称】オントロジー文書を生成する方法および装置

(57)【特許請求の範囲】
【請求項１】
ドメイン内の項目の諸クラスの構造、どの項目がどのクラスに属するかおよびどのオペレーションを各クラスがサポートするかを指示するオントロジー文書を生成する、コンピュータ実装される方法であって：
ドメイン内の項目および項目間の関係についての情報を含むドメイン記述を処理エンジンに入力する段階と；
前記処理エンジンがデータ構造およびオペレーション構造にアクセスする段階であって、データ構造は前記ドメイン記述において記述される項目を諸クラスに分けて特徴づけるための項目および項目間関係のドメイン独立なモデルであり、データ構造は諸項目のクラスとして少なくとも親項目を含み、各親項目は該親項目がそれに対して呼び出されたオペレーション構造からのオペレーションを受け容れる結果として該親項目によって生成される子孫項目だけを含むことができ、オペレーション構造は、ドメイン記述に適用されたときに諸項目の諸クラスについての処理規則を定義するオペレーションの、ドメイン独立なリストを含む、段階と；
前記処理エンジンが前記データ構造および前記オペレーション構造を前記ドメイン記述に適用して、ソフトウェアを生成するのに使うオントロジー文書を生成する段階とを含む、
方法。
【請求項２】

親項目の各インスタンスは、項目の特定のクラスのインスタンスである子孫項目だけを含むことができる、請求項１記載のコンピュータ実装される方法。
【請求項３】
前記オントロジー文書に基づいてURIテンプレートを前記処理エンジンにおいて生成し、該処理エンジンから出力する段階をさらに含み、前記URIテンプレートは、ランタイムにおけるドメイン内のリソースの各インスタンスがHTTP　URIを介した参照により取得可能であることを保証するために使うことができ、リソースはフィールドおよび他のリソースへのリンクをもつ項目のクラスを総称する用語である、請求項１または２記載のコンピュータ実装される方法。
【請求項４】
前記オントロジー文書は、管理可能な包含者のクラスを定義し、該管理可能な包含者クラスのインスタンスは状態親項目を含み、状態親項目が含むことのできる子孫項目は、管理可能な包含者のクラスのオペレーティング状態のあらかじめ定義されたリストのうちの一つを指示するために使われる項目の状態クラスのインスタンスである子孫項目のみである、請求項１ないし３のうちいずれか一項記載のコンピュータ実装される方法。
【請求項５】
前記データ構造およびオペレーション構造が制約されたエディタによってアクセスされ、前記制約されたエディタは、ユーザーを、前記データ構造およびオペレーション構造が前記ドメイン記述に適用されるような仕方で前記ドメイン記述を入力することに制約する編集環境である、請求項１ないし４のうちいずれか一項記載のコンピュータ実装される方法。
【請求項６】
前記オントロジー文書を使って生成されるインターフェース明細は、RESTアーキテクチャ・スタイルにおけるインターフェースの原理と整合する、請求項１ないし５のうちいずれか一項記載のコンピュータ実装される方法。
【請求項７】
ソフトウェアを生成するコンピュータ実装される方法であって、請求項１ないし６のうちいずれか一項記載のオントロジー文書を生成する方法を含み、さらに：
前記処理エンジンにおいて、前記データ構造および前記オペレーション構造に基づく規則のセットを前記オントロジー文書に適用してソフトウェアを生成する段階を含む、
コンピュータ実装される方法。
【請求項８】
請求項７記載のコンピュータ実装される方法であって、前記オントロジー文書は、管理可能な包含者のクラスを定義し、前記管理可能な包含者クラスのインスタンスは状態親項目を含み、前記状態親項目が含むことのできる子孫項目は、管理可能な包含者のクラスのオペレーティング状態のあらかじめ定義されたリストのうちの一つを指示するために使われる項目の状態クラスのインスタンスである子孫項目のみであり、それにより、ランタイムにおける前記管理可能な包含者クラスのインスタンスのオペレーティング状態は、前記管理可能な包含者クラスの前記インスタンスに含まれる前記状態親項目に前記状態クラスのインスタンスを追加することによって管理されるよう動作可能であり、
ランタイムにおける前記管理可能な包含者クラスの前記インスタンスの状態は、前記管理可能な包含者クラスの前記インスタンスに含まれる前記状態親項目に対する前記状態クラスの最も最近追加されたインスタンスによって指示され、それにより、前記状態親項目に含まれる子孫項目は、前記管理可能な包含者クラスの前記インスタンスの状態履歴である、方法。
【請求項９】
前記管理可能な包含者クラスのインスタンスは、仮想コンピューティング・システムである、請求項８記載のコンピュータ実装される方法。
【請求項１０】
前記ソフトウェアが定義された受け容れ制約条件を含み、前記受け容れ制約条件は、ラ

ンタイムにおいて、親項目に対してオペレーションが呼び出された結果として該親項目のインスタンスによって生成されることのできる子孫項目の個別的なインスタンスを制約し、前記受け容れ制約条件はクライアント・ツールに対して利用可能である、請求項７ないし９のうちいずれか一項記載のコンピュータ実装される方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、オントロジー文書を生成する方法および装置に関する。

【背景技術】

【０００２】

「機械のためのウェブ（web for machines）」やweb 2.0という側面に向けた近年の前進は、多くのウェブサイトやドメインが、API（Application Programming Interface［アプリケーション・プログラミング・インターフェース］）のようなインターフェースを介して、公開されるデータおよびサービスへのアクセスを提供するということを意味している。結果として生じるウェブ・アプリケーションの消費者は、ウェブ・ブラウザーを通じてアプリケーションを駆動する人間ではなく、機械である。

【０００３】

APIおよびAPIサポートは、二つ以上のソースからのデータおよびサービスを組み合わせて派生サービス、マッシュアップおよびウェブ・アプリケーションのデスクトップ版を創り出すことをも可能にした。さらに、HTTPベースのAPIが、（主として、既存のウェブサイトを補完するためではなく）コンピューティング要件をサポートする手段として発達した。たとえば、サービスとしてのインフラストラクチャー（IaaS: Infrastructure-as-a-Service）クラウドAPIと称されるAPIである。サービスとしてのインフラストラクチャーという機構により、クライアント・コンピュータまたはクライアント・ネットワークは、たとえば処理および記憶タスクの実行のためにウェブを介して外部またはサードパーティーのコンピューティング・インフラストラクチャーにアクセスできる。

【０００４】

REST（Representational State Transfer［表現による状態転送］）は、機械のためのウェブにおける参加者の適正な振る舞いを支配するアーキテクチャ・スタイルである。RESTは、システム・アーキテクチャのための制約条件を述べるもので、それに従うことは「RESTフル」〔RESTful、REST準拠〕であると形容される。そうした制約条件の第一のものは、アーキテクチャがクライアント‐サーバー構成を有しており、クライアントはサーバーから一様なインターフェースによって隔てられているということである。クライアントとサーバーの間のインターフェースには四つの指導原理があり、これらの原理に基づいて開発されたインターフェースは「RESTフル」であると形容できる。たとえば、APIはインターフェースについてのREST指導原理に基づいて書かれることができ、よって「RESTフルなAPI」〔REST準拠API〕として記述されることになる。プロトコルとしてのHTTPはRESTフルな仕方で使用されることができ、RESTフルなHTTPは機械のためのウェブに好適である。RESTフルなAPIは、いくつかの主立った理由により人気がある：質実なアーキテクチャ原理をもって証明された基礎の上に構築された基本プロトコルの簡単さがあり、その結果はウェブ開発者によってアプローチ可能であり、使用可能である。

【０００５】

手短に言うと、RESTアーキテクチャ・スタイルは、システム・アーキテクチャに対する次のような六つの制約条件を述べている（六つのうち一つは任意的）。

・アーキテクチャはクライアント・サーバー式であるべきである；

・クライアントとサーバーは一様なインターフェースによって隔てられる；

・アーキテクチャはステートレス〔stateless、無状態〕である。これは、クライアントからの要求から次の要求にかけての間、クライアント・コンテキストがサーバー上に記憶されず、各要求が、該要求にサービスするのに必要な情報のすべてを含んでおり、状態情報はクライアント自身に保持されるということを意味する；

・クライアントは応答をキャッシュできる；
・（任意的）機能はサーバー転送論理（server transferring logic）によってクライアントに差し延べられてもよく、それをクライアントが実行できる。
【０００６】
一様な（uniform）インターフェースについての指導原理は、手短に言うと次のようにまとめられる：
・ドメイン内の個々のリソースがクライアントからの要求において同定されることができる。（これは、ウェブ・ベースのシステムではURI（Universal Resource Identifier［不変リソース識別子］）を介してになるだろう。）リソース自身はクライアントに返される表現とは別個のエンティティである；
・クライアントによって保持されるリソースの表現は、クライアントに、サーバー上のリソースを修正または削除する（パーミッションにより許容されるとして）ための十分な情報を与えるのに十分である；
・クライアントとサーバーの間の各メッセージは、受信側が該メッセージを処理するための十分な情報を含む；
・サーバーからクライアントに与えられるリソースの表現は、関係したリソースへのハイパーテキスト・リンクを含むべきである。
【０００７】
RESTアーキテクチャ・スタイルの肯定的な側面は、情報モデルとよくリンクするということである。情報モデルは、ドメイン内の項目および項目間の関係の形式化された（formalised）記述である。RESTフルなAPIにおいて許容される動作は制約（フィックス）されており、これは、そうでなければインターフェースを情報モデルとリンクする際の問題につながるであろう、貧弱なプログラミング挙動の望まれない副作用を回避する。
【０００８】
実際、特定のドメインについてのRESTフルなAPIは、純粋に、そのドメインについての情報モデルを用いて、そしてこのモデルが異なるデータ・フォーマット内部でどのように見えるかによって、定義されうる。データ・フォーマットは、情報モデルの、ワイヤ・レベル（低レベルまたは実装レベル）での発現である。残念ながら、現在使用されるAPIは、情報モデリングへのアプローチ、これがデータ・フォーマット内部でどのように見えるか、およびHTTPのセマンティクスが問題のAPI（単数または複数）の個別的なドメインにおいて使用されるに至るかに関して不一致を示す。こうした一貫性の欠如は、RESTフルなプロトコルの潜在的な恩恵、たとえば、再使用可能なツールキット（たとえば標準コード）および汎用クライアント・エージェント（ブラウザーと等価）の潜在的可能性が失われるので、問題である。
【０００９】
特定のドメインについての情報モデルは、たとえばOWL（Web Ontology Language［ウェブ・オントロジー言語］）フォーマットにおいて呈示されるオントロジー文書において取り込まれてもよい。オントロジー文書は次いで、APIまたはインターフェース明細の基礎として使われることができる。
【発明の概要】
【発明が解決しようとする課題】
【００１０】
現在のところ、APIを書き、潜在的可能性としてはそれに続いてそのAPIの個別的実装をするサービス開発者が取っているアプローチは、そのAPIが生成されるドメインに固有で、特定のデータ・フォーマットに好適な言語の修正または適応されたバージョンを定義することでありうる。たとえば、言語はJSONまたはXMLに基づくことがありうる。このデータ・フォーマット固有のアプローチは、異なるデータ・フォーマットは異なる処理規則に従って機能するので、問題である。各ドメインおよび各データ・フォーマットについてカスタム・コードが必要とされるので、共通の処理規則セットを使ってすべてのAPIにアクセスすることができる汎用クライアントの実現は不可能になる。

【課題を解決するための手段】
【００１１】
本発明の諸実施形態は、ドメイン内の項目の諸クラスの構造、どの項目がどのクラスに属するかおよびどのオペレーションを各クラスがサポートするかを指示するオントロジー文書を生成する、コンピュータ実装される方法であって、ドメイン内の項目および項目間の関係についての情報を含むドメイン記述を入力する段階と；データ構造およびオペレーション構造にアクセスする段階であって、データ構造は当該ドメイン記述において記述される項目を諸クラスに分けて特徴づけるための項目および項目間関係のドメイン独立なモデルであり、データ構造は諸項目のクラスとして少なくとも親項目を含み、各親項目は該親項目がそれに対して呼び出されたオペレーション構造からのオペレーションを受け容れる結果として該親項目によって生成される子孫項目だけを含むことができ、オペレーション構造は、ドメイン記述に適用されたときに諸項目の諸クラスについての処理規則を定義するオペレーションの、ドメイン独立なリストを含む、段階と；前記データ構造および前記オペレーション構造を前記ドメイン記述に適用して、ソフトウェア製品を生成するのに使うオントロジー文書を生成する段階とを含む、方法を提供する。
【００１２】
本発明の発明者は、クライアントとサーバーの間のインターフェースのランタイムでの信頼性を改善するには、プログラマーの活動を単に自動化するだけでは十分ではなく、この目的のために、ドメインをモデリングするオントロジー文書を生成することに向けたアプローチの根本的な再定式化を使うことができるという驚くべき認識に至った。本発明の方法は、この新しいアプローチを具現するものであり、コアとなるドメイン独立なデータ構造およびオペレーション構造をドメイン記述に適用してオントロジー文書を生成する。オントロジーの概念を、ドメイン従属およびドメイン独立な構造および記述に分解することにより、ドメインを横断してのアプローチの一貫性が保証され、任意のドメインについてのオントロジー文書に適用できる想定を、ドメイン独立な構造においてすることが許容される。
【００１３】
オントロジー文書は、フォーマットによって制約されず、ドメイン内の諸項目の諸クラスの構造、どの個々の項目がどのクラスに属するかおよびたとえば各クラスがどのオペレーションをサポートするかを指示する情報を含みうる。オントロジー文書はOWL（ウェブ・オントロジー言語）で書かれてもよい。あるいはまた、オントロジー文書は、単一の文書として形式的に（formally）呈示されなくてもよく、ドメインのオントロジーの表現に寄与する情報の任意の集合であってよい。
【００１４】
本発明を具現する方法によって生成されるオントロジー文書は、ソフトウェア製品（artefact）〔生成物〕を生成するのに使うのに好適である。ソフトウェア製品はコンピュータ・プログラムまたはコンピュータ・プログラムのコードまたはソフトウェア・コードまたはコンピュータ・プログラムを生成するために直接使うことのできるプロトコルまたは明細である。ソフトウェア製品は、次のうちの一つまたは複数を含んでいてもよい：クライアント・アプリケーション、API、クライアント実装、サーバー側実装、ウェブ・ページ、アプリケーション・スケルトンまたは好ましくはインターフェース明細。
【００１５】
「インターフェース明細（interface specification）」の用語は、ドメイン内のサービスおよび公開されているデータにアクセスするためのプロトコルを含みうる。データおよびサービスは、ドメイン内の項目およびリソースの表現がクライアント・アプリケーションまたはサーバー側マッシュアップに関してクライアントの役割を演じるピア・サーバー・アプリケーションに対して利用可能になっている限り公開されていることができる。その代わりまたはそれに加えて、ドメインまたは完全なポートフォリオ・サービスにおけるデータおよびサービスは、クライアント・アプリケーションによって管理可能である限り公開されていることができる。好ましくは、「インターフェース明細」の用語は、特定

のドメインとの関連付けを含むが、該インターフェースをいかなる特定のクライアント・アプリケーションに関連付けたり、該インターフェースを（APIのような）明細の特定の実装に制約したりするものではない。インターフェース明細は、たとえば、次のうちの任意のものまたは全部を含んでいてもよい：
・URIテンプレート；
・諸項目の諸クラスを含むデータ・フォーマット；
・含む、受け容れるおよびその他の型に範疇分けされる、クラス間のリンク；
・各クラスについての許容可能なHTTPメソッド；
・どのデータ・オブジェクトが各クラス上で編集可能か；
・インスタンスに対してHTTP GETオペレーションが実行されたときに返される、クラスのインスタンスの表現；
・各クラスの新しいインスタンスを生成できるエンティティ。生成は、POST要求の有効な受け手に対するPOSTオペレーションを介してであってもよい。これはまた、包含者内のどの子リソースが生成されることをクライアントが期待するかを含め、各クラスのインスタンスの生成オペレーションをどのように処理するかについてのサーバー側への汎用命令をも含んでいてもよい；
・どのクラスが管理可能（manageable）であるか。

【００１６】

「インターフェース」の用語は、クライアントとサーバーの間のインターフェースを含んでいてもよい。

【００１７】

「クラスのインスタンス」の用語は、ランタイムにおいてドメインに存在する項目であって、そのクラスのテンプレートに従うものを含んでいてもよい。テンプレートは、データ構造およびオペレーション構造をドメイン記述に適用することによって定義される。よって、各インスタンスはクラスに「属する」。同様に、「サブクラスのインスタンス」は、そのサブクラスおよび該サブクラスが発生するもとになったクラスに属するインスタンスを含みうる。リソースは、他のリソースへのリンクをもち、フィールドをもち、任意的に削除されることができる項目の汎用クラスを含んでいてもよい。

【００１８】

ドメイン記述は、少なくとも、該ドメイン内の項目およびそれらの項目間の関係についての情報を含んでいてもよく、ドメイン・オントロジーを含んでいてもよい。好ましくは、ドメイン記述は、ドメイン内の利用可能なクラスおよびそれがサポートするメソッドの完全なまたは実質的に完全な表現を含むオントロジー文書が生成されることを許容する仕方で呈示される十分な情報を含む。したがって、オントロジー文書は、該オントロジー文書から、ソフトウェア製品として、少なくとも部分的インターフェース明細が導出される（帰納される、演繹されるまたは表面化する）ことを許容する。たとえば、ドメイン記述は、該ドメイン記述内に存在する各クラスおよびサブクラスに属するリソースの包含階層構造がオントロジー文書において表現されることができるような仕方で呈示されてもよい。

【００１９】

「データ構造」の用語は、処理エンジン内で任意的に与えられる、あるいはオントロジー文書を生成する際にアクセスされるデータ構造をいうが、項目および項目間関係のドメイン独立なモデルを含んでいてもよく、ドメイン独立なデータ・モデルまたはコア・データ・モデルを含むと考えられることができる。用語「データ構造」は「データ構造オントロジー」を含んでいてもよい。データ構造は、ポスト可能（Postable）、集合（Collection）、共有可能（Shareable）、状態（State）、削除可能（Deletable）および管理可能（Manageable）のようなコアとなるドメイン独立なクラスを提供してもよい。処理エンジンにおいてドメイン構造がデータ記述に適用されると、ドメイン内のリソースは、ドメイン独立なクラスまたはそれらのドメイン独立なクラスから発生し、よって該ドメイン独立なクラスの属性を継承するサブクラスのインスタンスとして特徴づけられることができる

。

【００２０】

「オペレーション構造」の用語は、処理エンジン内で与えられる、あるいはオントロジー文書を生成する際にアクセスされるオペレーション構造をいうが、ドメイン記述に適用されたときに、所定のクラスに属するリソースに対して実行されるオペレーションをどのように扱うかをサーバーに伝える処理規則を定義するオペレーションのドメイン独立なリストを含んでいてもよく、ここで、これらの処理規則がリソースの状態に基づいて変わることが可能である。用語「オペレーション構造」は「オペレーショナル・オントロジー」を含んでいてもよい。オペレーション構造は、オペレーション・モデルと考えられることができる。

【００２１】

オペレーション構造およびデータ構造は、ドメイン独立な情報モデルと称されることができる。ひとたびオペレーション構造およびデータ構造がドメイン記述に適用されると、ドメイン固有情報モデルまたはオントロジー文書が取得されることができ、これはインターフェース明細のようなソフトウェア製品の基礎をなすのに好適である。

【００２２】

「リソース（resource）」の用語は、クラス「リソース（Resource）」の項目を含んでいてもよい。クラス「リソース」は、他のリソースへのリンクをもち、フィールドをもち、任意的に削除されることができる項目の汎用クラスである。用語「包含者（container）」は、クラス「包含者（Container）」の項目を含んでいてもよい。クラス「包含者」は、クラス「リソース」を拡張し（クラス「リソース」のサブクラスであり）、他のリソースを含むことができるクラスである。包含者に対する削除オペレーションは、その包含者およびそれに含まれるすべてのリソースを除去する。用語「親（progenitor）」は、クラス「親（Progenitor）」の項目を含んでいてもよい。クラス「親」は、クラス「包含者」を拡張し（クラス「包含者」のサブクラスであり）、子孫（Progeny）項目を生成することができるクラスである。子孫項目自身も親のインスタンス内でのリソースのサブクラスであり、たとえば親クラスのインスタンスに対して呼び出された（実行された）「ポスト（post）」オペレーションの結果である。クラス「親」は、データ構造の一部であり、ドメイン独立な概念である。これは、アプリケーション明細を生成するときにドメイン内のどのリソースが親として分類されるかを推定または他の技法を使って識別するためにドメイン記述に適用されることができる。

【００２３】

有利には、親項目を含むデータ構造に基づいてオントロジー文書を生成することは、結果として、オペレーション上の用語の誤用により生じる問題が避けられるソフトウェア製品（アプリケーション明細のような）の基礎をなすことのできるオントロジー文書を生成する。たとえば、HTTPオペレーションGETおよびPUTの誤用によって生じうるAPIまたは他のソフトウェア製品は、共通の処理規則に従わない、よって特殊なクライアント・アプリケーションが、当該APIが生成された対象のドメイン内での公開されたデータおよびサービスにアクセスすることを必要とする。POSTオペレーション（オペレーション構造の一部）に応答しての親項目（データ構造の一部）の振る舞いは、親項目自身（生成によって新しい項目が追加される包含者である）の性質から自明である。今や親と分類され、POSTオペレーションに応答してのよく定義された振る舞いをもつ当該ドメインの諸部分は、当該ドメインのうち、当該ドメインのためのAPIを生成する開発者によるGETおよびPUTの誤用およびPUTオペレーションの誤用に影響を受けやすかったであろう部分である。本発明の方法により生成されるオントロジー文書に基づくインターフェース明細を実装するインターフェースのようなソフトウェア製品のランタイム信頼性は、従来のインターフェースに対して改善される。

【００２４】

任意的に、本発明を具現する方法において、ドメイン記述が処理エンジンに入力され、データ構造およびオペレーション構造が処理エンジンによりアクセスされ、データ構造お

よびオペレーション構造が処理エンジンにおいてドメイン記述に適用され、オントロジー文書が処理エンジンによって生成される。

【００２５】

処理エンジンは、プロセッサを有するコンピューティング装置で実行されたときに、該コンピューティング装置に、ドメイン記述を入力として受け容れる段階と、データ構造およびオペレーション構造をドメイン記述に適用する段階と、インターフェース明細を出力として生成する段階とを実行させるソフトウェアとして実現されてもよい。あるいはまた、処理エンジンは、ドメイン記述を入力として受け容れ、データ構造およびオペレーション構造をドメイン記述に適用してインターフェース明細を出力として生成するよう動作可能なコンピューティング装置であってもよい。あるいはまた、この側面における処理エンジンは、のちに詳述する制約されたエディタとして実行する機能があってもよい。

【００２６】

有利には、処理エンジンを使ってデータ構造およびオペレーション構造をドメイン記述に適用する方法を提供することにより、オントロジー文書を生成するアプローチの一貫性が改善できる。

【００２７】

好ましい諸実施形態では、親項目の各インスタンスは、項目の特定のクラスのインスタンスである子孫項目を含むことができるだけである。

【００２８】

親項目のメンバーシップはさらに所定のクラスの特定の諸サブクラスに、あるいはクラスまたはサブクラス内の個々のものに制約されてもよい。この意味での個々のものとは、所定のクラス内の項目の型である。有利には、このようにして親項目のメンバーシップを制約することは、親項目を扱うために必要とされる処理規則を単純化する。それにより、オントロジー文書が提供するドメインの表現においては、該オントロジー文書に基づいてインターフェース明細を実装するインターフェースのようなソフトウェア製品を走らせる際に使われるべき処理規則は単純であり、よく定義されている。

【００２９】

本発明を具現する好ましい諸実施形態は、さらに、オントロジー文書に基づいてURIテンプレートを処理エンジンにおいて生成し、処理エンジンから出力することを含んでいてもよい。ここで、URIテンプレートは、ランタイムにおけるドメイン内の各リソースがHTTP URIを介した参照により取得可能であることを保証するために使うことができる。

【００３０】

形式的には不透明であるが、URI（普遍的リソース識別子）テンプレートは、オントロジー文書に基づくインターフェース明細の実装のようなソフトウェア製品において使用されるURIがオントロジー文書において取り込まれ、ドメイン記述から導出される包含階層構造を反映することを保証するので、有用である。処理エンジンは、先述したのと同じ処理エンジンであってもよい。処理エンジンは、のちに詳述する制約されたエディタおよび変換エンジンの組み合わせとして機能できる追加的な機能をもつ制約されたエディタであってもよい。

【００３１】

好ましくは、URIテンプレートのフル・セットが生成される。それにより、ドメイン内の項目の各クラスについてのURIテンプレートが利用可能にされる。

【００３２】

好ましい諸実施形態では、オントロジー文書は、管理可能な包含者（manageable container）のクラスを定義する。該管理可能な包含者クラスのインスタンスは状態親項目（states progenitor item）を含み、状態親項目が含むことのできる子孫項目は、管理可能な包含者のクラスのオペレーティング状態のあらかじめ定義されたリストのうちの一つを指示するために使われる項目の状態（state）クラスのインスタンスである子孫項目（progeny item）のみである。

【００３３】

このようにして管理可能な包含者を定義するオントロジー文書が提供する機構によって、クライアントは、簡単なデータ・オブジェクト（新しい状態項目）を状態親項目にポストすることによって、ドメイン内の諸項目の状態を管理できる。

【００３４】

たとえば、仮想システム・ドメイン内の親「サーバー（servers）」は、クライアントがPOSTオペレーションを実行する結果として子孫項目「サーバー（server）」を生成できてもよい。この場合、受け容れ制約条件は、オペレーションPOSTについては、子孫項目サーバーが生成されるというものであり、これは、環境変数に関わりなく真であり続ける。別の例では、管理可能なリソースは親「状態（states）」を有していてもよい。親「状態」は、それに対してクラス「状態（State）」のサブクラスを指定するPOSTオペレーションが呼び出されるときに、子孫項目として新状態リソースを生成できる。状態親（states progenitor）が状態「オフ」を含むとき、クラス「状態」のサブクラスであるリソース「オン」を指定するPOSTオペレーションだけを受け容れるのでもよい。これらの制約条件をクライアント・ツールに利用可能にすることによって、インターフェース明細は事実上、これらの受け容れ制約条件がクライアント・ツールに広告されることのできる手段を提供する。広告は、問い合わせに応答してであってもよいし、それ以外であってもよい。たとえば、クライアント・ツールは、APIを介してアクセスされているドメイン内のリソースが管理可能であることを識別し、該管理可能なリソースの状態を修正するためにどの状態項目が状態親において生成されることができるかを見出すべくサーバーに問い合わせを提出してもよい。

【００３５】

上述したように、制約されたエディタは代替的には、処理エンジンの機能として自動化されてもよい。

【００３６】

任意的に、データ構造およびオペレーション構造は制約されたエディタによってアクセスされる。制約されたエディタは、ユーザーを、データ構造およびオペレーション構造がドメイン記述に適用されるような仕方でドメイン記述を入力することに制約する編集環境である。

【００３７】

制約されたエディタは、オントロジー文書（ontology document）を生成するために必要とされる専門知識を減らすためにグラフィカル・インターフェースを含んでいてもよい。制約されたエディタは、ソフトウェアとして実現されるとき、それ自身が本稿に記載される方法を使って導出されたものでもよく、オントロジー表現（ontology representation）がグラフまたは図またはマップといったグラフィカルなフォーマットで構築されることを許容することによってユーザーの行動を制約してもよく、そうしたグラフィカルなフォーマットにおいて、どの段階においても、オントロジー表現に対する許容される追加はデータ構造およびオペレーション構造と整合するものだけである。さらに、制約されたエディタは、連続的にまたは間欠的に、あるいはユーザーの要求の際に、データ構造およびオペレーション構造との整合性を保証するためにオントロジー表現を見直してもよい。入力は、ドメイン記述情報を、制約されたエディタをホストするコンピュータ端末（あるいは制約されたエディタはウェブ・アプリケーションまたはデバイスであってもよい）に入力することを含んでいてもよく、ドメイン記述を生成することを含んでいてもよい。

【００３８】

好ましい諸実施形態では、オントロジー文書を使って生成されるソフトウェア製品は、RESTアーキテクチャ・スタイルにおけるインターフェースの原理と整合する。分散式ハイパーメディア・システムのためのRESTアーキテクチャ・スタイルは、ウェブにおける参加者の適正な振る舞いを支配する、クライアントとサーバーの役割の明確な分離のために、ウェブ開発において好まれている。RESTアーキテクチャ・スタイルは望ましいが、該スタイルと整合するクライアントとサーバーの間のインターフェースを生成することは時間がかかり、誤りを起こしやすい。本発明のオントロジー文書を導出することに向けたアプロ

ーチの再定式化は、オントロジー文書がインターフェース明細を生成する際に使うのに好適であり、RESTアーキテクチャ・スタイルまたは他のRESTフルなソフトウェア製品に準拠するインターフェースを生じることを、データ構造およびオペレーション構造自身が保証する方法を提供する。ユーザーはREST制約およびスタイルを意識する必要はなく、単にデータ構造およびオペレーション構造に従うことによってREST準拠の明細を生成する際に使うためのオントロジー文書を生成できる。

【００３９】

データ構造およびオペレーション構造におけるドメイン独立なレベルでのある種の想定をすることによって、それらの想定（項目のある種のクラスに対して実行されるある種のオペレーションの組み合わせについての処理規則のような）は、本発明の方法を使って生成されたすべてのソフトウェア製品を横断して伝わる。データ構造およびオペレーション構造は、許容されるオペレーションと、RESTアーキテクチャ・スタイルと整合し、よってRESTフルなAPIの生成につながるドメイン内のリソースの表現の仕方とを定義する。本発明に基づく方法は、RESTフルなAPI、RESTフルなソフトウェア製品およびインターフェース明細を開発し、生成するプロセスに対して、データ・フォーマットおよび対話セマンティクスの使用に関して一貫性を保証する系統的なアプローチを提供する。

【００４０】

好ましい諸実施形態はさらに、処理エンジンにおいて、データ構造およびオペレーション構造に基づく規則のセットをオントロジー文書に適用してソフトウェア製品を生成することを含んでいてもよい。本発明の方法を使って生成されたオントロジー文書の内容に基づいて、インターフェース明細を生成するために、オントロジー文書において明示的に述べられていない項目間の関係のような、追加的な属性および情報を推定し、導出するために、規則のセットが適用できる。データ構造およびオペレーション構造をドメイン記述に適用するプロセス全体も処理エンジン内で行われるのであってもよい。それにより、シングルクリック・プロセスによってユーザーはドメイン記述からソフトウェア製品を生成できる。この場合のオントロジー文書は、複数のオントロジー情報として実現されてもよく、このオントロジー情報に対して、ソフトウェア製品を生成する際に規則が適用される。あるいはまた、データ構造およびオペレーション構造を取り込むオントロジー文書が、ユーザーにより記憶可能でありアクセス可能である単一のエンティティであり、ソフトウェア製品を生成するために処理エンジンに入力するのであってもよい。この点に関して、処理エンジンは、先に定義された処理エンジンの一部として、変換エンジンを含んでいてもよい。

【００４１】

任意的に、本発明を具現する方法は、処理エンジンにおいて、データ構造およびオペレーション構造に基づく規則のセットを入力オントロジー文書に適用してソフトウェア製品を生成することを含む方法であってもよい。オントロジー文書は、形式的なOWL文書として入力されてもよいし、あるいはドメインのオントロジーを表す複数の情報として処理エンジンに供給されてもよい。オントロジー文書は、本発明を具現する方法に基づいて生成されたものであってもよい。

【００４２】

さらに、本発明を具現する方法によって生成されたオントロジー文書およびインターフェース明細は、ドメイン独立な構造が適用されるため、異なる型のメディアを扱うことに向けたアプローチの一貫性を保証できる。それにより、リソースの表現を扱うためのXMLスキーマは必要とされない。汎用クライアント・ツールを任意のドメインとのインターフェースをもつために使うことができることは、本発明を具現する方法の追加的な効果である。すなわち、諸ドメインの範囲を横断して公開されるデータおよびサービスにアクセスするためのクライアント・ツールであって、各ドメイン内のリソースの表現を扱うための個別の処理規則を必要としないものが、実現できる。ドメイン独立なデータ構造およびオペレーション構造は、クライアント・ツールが、ドメインから情報を得るために汎用の言語、問い合わせおよび要求を使うことができ、ドメイン内のリソースの表現を扱うために

汎用の処理規則を使うことができるというものである。

【００４３】

生成されたソフトウェア製品がインターフェース明細であるとき、インターフェース明細はRESTアーキテクチャ・スタイルにおけるインターフェースのための原理に従ってインターフェースを定義する。

【００４４】

ある好ましい実施形態では、ドメイン記述はユーザーによってコンピュータ端末に入力されてもよく、データ構造およびオペレーション構造はコンピュータ上で走っているプログラムによってアクセスされ（あるいはコンピュータによってリモート・アクセスされる）、制約された編集環境において入力されたドメイン記述に適用されてもよい。次いで、指定されたフォーマットにおける中間オントロジー文書が生成されてもよく、これ自身も、オントロジー文書に規則のセット（推定オペレーションおよび導出オペレーション）を適用してインターフェース明細または他のソフトウェア製品を生成するよう動作可能な変換エンジン（任意的にはコンピュータ上で実行されるプログラム）に入力されることができる。形式的なオントロジー文書の中間生成は省略されることができ、それによりオントロジー文書は形式的な文書ではなく、むしろ制約された編集環境においてデータ構造およびオペレーション構造をドメイン記述に適用した産物となる。インターフェース明細および他の出力が、制約された編集環境内で利用可能にされる。

【００４５】

任意的に、本発明の諸実施形態において、データ構造およびオペレーション構造は、生成されたソフトウェア製品がRESTアーキテクチャ・スタイルにおけるインターフェースのための原理と整合することを保証する。

【００４６】

RESTアーキテクチャ・スタイルの恩恵は十分に文献に示されているが、RESTの制約条件および原理に従うインターフェースおよびインターフェース明細のコーディングは時間がかかり、誤りが起こりやすいものである。本発明の諸実施形態は、ユーザーが、ドメイン記述を生成するためのドメイン知識だけを使ってインターフェース明細を生成する際に使うためのインターフェース明細または文書を生成することを可能にする。本発明のアプローチは、それを、インターフェースについてのオペレーション上の詳細におけるユーザー・コントロールおよび柔軟性も最適化されることができるような仕方で行う。ドメイン独立なデータ構造およびオペレーション構造は、それらの構造と整合するオントロジー文書またはインターフェース明細が生成され、それらの構造の恩恵がオントロジー文書において表されている情報モデルに内在的であるようにするために、ドメイン記述に適用される。したがって、ユーザーは、基本的なドメイン独立なデータ構造およびオペレーション構造に内在する恩恵を損なうことなく、一連のアプリケーションにおいてオントロジー文書またはインターフェース明細を自由に利用できる。

【００４７】

好ましい諸実施形態では、オントロジー文書は、管理可能な包含者のクラスを定義する。管理可能な包含者クラスのインスタンスは状態親項目を含み、状態親項目が含むことのできる子孫項目は、管理可能な包含者のクラスのオペレーティング状態のあらかじめ定義されたリストのうちの一つを指示するために使われる項目の状態クラスのインスタンスである子孫項目のみである。それにより、ソフトウェア製品またはソフトウェア製品の実装のランタイムにおける管理可能な包含者クラスのインスタンスの状態は、管理可能な包含者クラスのインスタンスに含まれる状態親項目に状態クラスのインスタンスを追加することによって管理されるよう動作可能である。

【００４８】

有利には、オントロジー文書が上に定義されたように管理可能な包含者を定義する本発明の諸実施形態は、クライアント・ツールまたはアプリケーションが、単純なHTTPまたは等価なオペレーションを使ってサーバー側で項目の状態を管理することを許容する。このアプローチは、RESTアーキテクチャ・スタイルにおけるインターフェースのための原理に

準拠する。

【００４９】

好ましい諸実施形態では、ランタイムにおける管理可能な包含者クラスのインスタンスの状態は、管理可能な包含者クラスのインスタンスに含まれる状態親項目に対する状態クラスの最も最近追加されたインスタンスによって指示される。それにより、状態親項目に含まれる子孫項目は、管理可能な包含者クラスのインスタンスの状態履歴である。

【００５０】

このようにして状態を管理することは、何ら追加的な処理や記憶を必要とすることなく、状態履歴が利用可能になるという驚くべき、望ましい効果をもつ。状態親（states progenitor）における状態のリストがHTTP GETオペレーションが状態親項目（states progenitor item）上で実行されるのに応答して返されるのであってもよい。状態親項目内に保持される状態の数は制約されてもよい。

【００５１】

任意的に、管理可能な包含者クラスのインスタンスは、仮想コンピューティング・システムである。仮想ネットワークおよび仮想コンピューティング・システムは、状態管理を必要とし、これはクライアント・ツールまたはクライアント・アプリケーションからであってもよい。有利には、仮想コンピュータ・システムが管理可能な包含者クラスとして扱われ、このようにしてオントロジー文書および結果として得られるインターフェース明細において分類される本発明の諸実施形態では、クライアントは、仮想コンピューティング・システムを、RESTアーキテクチャ・スタイルに準拠する仕方で管理できる。

【００５２】

好ましくは、ソフトウェア製品は、定義された受け容れ制約条件を含む。該受け容れ制約条件は、該ソフトウェア製品または該ソフトウェア製品の実装のランタイムにおいて、それに対してオペレーションが呼び出された結果として親項目のインスタンスによって生成されることのできる子孫項目の個別的なインスタンスを制約する。受け容れ制約条件は汎用クライアント・ツールに対して利用可能である。

【００５３】

親項目がクライアント・アプリケーションに対して広告できる動的受け容れ制約を定義することが望ましいことがありうる（クライアント要求に応答してサーバーが一部またはすべての受け容れ制約条件の詳細を返すことになる）。たとえば、状態親項目の次の状態は現在状態に依存することがあり、よって、ある種の状態がある種の他の状態の次にくることがあり、ある種の状態は排除されることがありうる。

【００５４】

任意的に、ドメインはクラウド・コンピューティング環境であり、ソフトウェア製品はサービスとしてのインフラストラクチャーAPIとして実装されることのできるインターフェース明細であり、あるいはソフトウェア製品はサービスとしてのインフラストラクチャーAPIである。クラウド・コンピューティングはインターネット・ベースのコンピューティングである。デバイスは、互いにリソースを共有し、サービスを提供し合うことができる。デバイスおよびサービスは、公共施設のようにオンデマンド式にアクセスされることができる。実際、本発明の諸実施形態は、サービスとしてのソフトウェアAPI、サービスとしてのプラットフォームAPI、サービスとしてのソフトウェアAPIを生成するためにも使われることができる。具体的に、サービスとしてのインフラストラクチャーでは、コンピュータ・インフラストラクチャーはインターネット上で利用可能なリソースとして提供される。

【００５５】

有利には、本発明を具現する方法によって生成されるインターフェース明細およびオントロジー文書は、RDF（Resource Description Framework［リソース記述枠組み］）に基づいてデータをモデル化するために作られることができる。RDFは、多様なシンタックス・フォーマットを使う、ウェブ・リソースにおいて実装される情報の概念的な記述またはモデリングのための一般的な方法である。本発明の実施形態によって生成されるオント

ロジー文書およびソフトウェア製品のRDFバックグラウンドのおかげで、汎用問い合わせ機能がクラウドIaaS API（またはクラウドPaaS API）に追加されることができ、クラウド内の環境条件およびオペレーティング条件に関するライブの情報が、ランタイムにおいて前記汎用問い合わせ機能を用いるクライアント・アプリケーションによってアクセスされることを可能にする。

【００５６】

本発明は、ドメイン内の項目の諸クラスの構造、どの項目がどのクラスに属するかおよびどのオペレーションを各クラスがサポートするかを指示するオントロジー文書を生成するよう構成されたコンピューティング装置であって、ドメイン内の項目および項目間の関係についての情報を含むドメイン記述を入力する入力手段と；データ構造およびオペレーション構造にアクセスするアクセス手段であって、データ構造は当該ドメイン記述において記述される項目を諸クラスに分けて特徴づけるための項目および項目間関係のドメイン独立なモデルであり、データ構造は諸項目のクラスとして少なくとも親項目を含み、各親項目は該親項目がそれに対して呼び出されたオペレーション構造からのオペレーションを受け容れる結果として該親項目によって生成される子孫項目だけを含むことができ、オペレーション構造は、ドメイン記述に適用されたときに諸項目の諸クラスについての処理規則を定義するオペレーションの、ドメイン独立なリストを含む、手段と；前記データ構造および前記オペレーション構造を前記ドメイン記述に適用して、ソフトウェア製品を生成するのに使うオントロジー文書を生成する生成手段とを有する装置として具現されてもよい。入力手段は、たとえば別のプログラムまたは装置からまたはユーザーから入力を受け容れるための手段を含んでいてもよい。コンピューティング装置は、本稿の他所で記述される制約されたエディタであってもよい。

【図面の簡単な説明】

【００５７】

【図１】本発明を具現するAPIを生成することに向けたアプローチの概念的マップを示す図である。

【図２】オペレーション、クラスの型およびそれらの間のリンクの図的な表現を示す図である。

【図３】本発明を具現するプロセスを示す図である。

【図４】Ａ～Ｃは、ドメイン・オントロジーにおいて普通に観察されるパターンを示す図である。

【図５Ａ】本発明が仮想コンピューティング・ドメインに適用される実施形態における仮想システムのグループの概略図である。

【図５Ｂ】本発明が仮想コンピューティング・ドメインに適用される実施形態における仮想システムのグループに対して実行されるpost〔ポスト〕オペレーションの概略図である。

【図５Ｃ】本発明が仮想コンピューティング・ドメインに適用される実施形態における単一の仮想システム・リソースの概略図である。

【図５Ｄ】本発明が仮想コンピューティング・ドメインに適用される実施形態において新しい状態が仮想システム・リソース内の「状態」親項目（States progenitor item）にpostされる概略図である。

【図６】本発明の諸実施形態におけるデータ構造を例示する「システム」リソースの概略図である。

【図７Ａ】本発明を具現するコンピュータ・プログラムによって出力されるドメイン全体表示の例を示す図である。

【図７Ｂ】本発明を具現するコンピュータ・プログラムによって出力される子孫クラス表示の例を示す図である。

【図７Ｃ】本発明を具現するコンピュータ・プログラムによって出力される親クラス表示の例を示す図である。

【図８】Ａは子孫クラス表示の明細セクションの詳細図である。Ｂは親クラス表示の明細

セクションの詳細図である。
【発明を実施するための形態】
【００５８】
図１は、本発明を具現する方法において使用されるアプローチの概念マップである。抽象的なオントロジー１０はいかなるアドレッシング可能な形において存在するのでもなく、オントロジーの一般化された概念である（オントロジーとは「存在するもの」である）。抽象的なオントロジー１０から延びる矢印は、オントロジーの抽象化から実際のオントロジー（これはたとえばドメインを定義できる）に移行するためには、ドメイン・オントロジー、データ構造オントロジーおよびオペレーショナル・オントロジーの三つの側面が考慮されることができるという発想を表している。換言すれば、何らかのドメインに存在しているものは、本発明を具現する方法によって拡張され、ドメイン・オントロジー、データ構造オントロジーおよびオペレーショナル・オントロジーを用いて考えられることができる。ドメイン・オントロジーはドメイン固有言語、項目および関係であり、これらはドメイン記述によって何らかの可読なまたは使用可能な形で記述されてもよい。好ましくは、それはユーザーによって入力されるドメイン固有情報である。特に、ドメイン・オントロジーは、ドメイン内の項目を、項目のクラスおよび項目のサブクラスならびにそれらのクラスおよびサブクラスが関係している関係の性質として記述してもよい。
【００５９】
オペレーショナル・オントロジーおよびデータ構造オントロジーは、すべてのドメインについての汎用の関係、規則および想定を定義するドメイン独立な側面である。データ構造オントロジーおよびオペレーショナル・オントロジーのコンポーネントは、拡張されたドメインについての完全な情報モデルを与えるために使われることのできる構成ブロックのためのテンプレートと考えられることができる。
【００６０】
データ構造オントロジーは、項目のクラスどうしの間の関係（リンク）の、「含む」「受け容れる」および「その他」への範疇分けを含んでいてもよい。すなわち、項目のクラスAと項目のクラスBとの間の関係は、「含む」と範疇分けされうる。これはすなわちAがBを含むことを意味する。ドメインにおいて、項目a（クラスAに属する）および項目b（クラスBに属する）は、リンクされていれば、「aがbを含む」という関係を介してリンクされることができる。換言すれば、クラスAはクラスBの項目を含みうる包含者（container）である。項目のクラスAが「受け容れる」と範疇分けされる関係を介して項目のクラスBにリンクされ、したがって、そのドメインにおいて、項目aが項目bを受け容れることができるということが真であることもある。換言すれば、クラスAの項目はクラスBの項目を受け容れることができる。
【００６１】
データ構造オントロジーのコンポーネントは、「スロット」と称される項目のクラス上で公開される単純なデータ項目を含む属性であってもよい。この意味における公開とは、スロット内にあるデータ項目が、編集するために（「編集可能なスロット」）あるいは単に表現を取得するために（「読み出し専用スロット」）クライアントによってアクセス可能であることを意味する。
【００６２】
データ構造オントロジーの他のコンポーネントは次のものであってもよい：
・項目のクラス上でどのHTTPメソッドが許容されるかのリスト；
・リソース上でHTTP　GETオペレーションが実行されるときに返される該リソースの表現；
・特定のリソースの新しいインスタンスを生成する役割を担うエンティティ。これはそのリソースが生成されるPOSTオペレーション要求を有効に受領することのできるリソースであってもよい（有効な受領者はクラス「親（Progenitor）」および「集合（Collection）」のものである。推定により、そのような有効な受領者がデータ構造オントロジーに「親」としてフィットすることがわかる）；

・一般に、包含者内に新しいリソースを生成する効果をもつオペレーションをサーバーがどのように処理するべきか；
・どのリソースが削除可能か；
・どのリソースが管理可能（manageable）か。
【００６３】
オペレーショナル・オントロジーは、GET、POST、DELETEおよびPUTのような、メソッドの、コアのドメイン独立なリストと、これらがサーバーによってどのように扱われるべきかについての処理規則とを提供する。任意的に、オペレーショナル・オントロジーは、当然のこととしてはドメイン記述に適用されないが特定のドメインについて必要とされる場合には使用されうる追加的なメソッドまたはオペレーションを含んでいてもよい。しかしながら、RESTアーキテクチャ・スタイルの原理と整合する本発明の諸実施形態では、オペレーションGET、POST、DELETEおよびPUTはオペレーショナル・オントロジーに含まれることが含意されている。オペレーショナル・オントロジーはより多くのオペレーションをもって拡張されてもよいが、異なるオペレーションの数を最小限に保つことが望ましい。
【００６４】
データ構造オントロジーおよびオペレーショナル・オントロジーはブラックボックス中でドメイン・オントロジーに適用される。ブラックボックスは、処理エンジンの役割を実行し、オントロジーの諸側面を取り入れて、インターフェース明細（抽象API　３０）を生成する。ブラックボックス内の手順は自動化されることができるが、ある種の段階ではユーザー入力が必要とされることも可能である。ドメイン独立なデータ構造オントロジーおよびオペレーショナル・オントロジーをドメイン・オントロジーに適用する、ブラックボックス内で生起する機構は、次のうちの一つまたは複数を含む技術的プロセスである：
・ドメイン・オントロジーならびにドメイン独立なデータ構造オントロジーおよびオペレーショナル・オントロジーから導出可能な推定規則に基づく、ドメインについての追加的情報を推定するための推定オペレーション――追加的情報はドメイン・オントロジーにおいて明示的に与えられていないがそれから導出可能である；
・集合として表された項目のクラスのメンバーシップ基準を計算するための、ドメイン・オントロジーにおいて表されている属性からの推論；
・クラス等価属性を使って、ブラックボックスはドメイン内の項目の各クラスについてクラス等価を計算する。これは、ブラックボックスが、たとえば、あるクラスが「親」クラスであると推定することを許容する。すると、ブラックボックスは、インターフェース明細および他のコード製品のような出力を生成するときに特定のパターンをたどることを知る；
項目の各クラスにどの「スロット」が存在するかを計算することを含む、各クラスのビューの構築。項目の各クラスについて、関連付けられた属性の列挙が計算される。スーパークラス属性から継承された属性について（すなわち、クラスAが属性Xをもち、クラスA1がAのサブクラスであり、よってクラスA1は属性Xを継承する）、最も導出された属性だけが選択される。
【００６５】
ブラックボックス内でドメイン独立なオペレーショナル・オントロジーおよびデータ構造オントロジーをドメイン・オントロジーに適用するプロセスの一部は、再帰的なプロセスとして記述できる。該再帰的プロセスにおいては：
・ドメイン・オントロジー内の項目の各クラスについて、そのクラスが「親」クラスであれば、生成側面をもつオペレーション（Post［ポスト］、Copy［コピー］、Move［移動］）をそのクラスに関連付け、あるいは非生成オペレーション（Get［ゲット］、Put［プット］およびDelete［削除］）を非親項目に関連付ける；
・項目のクラスが「親」クラスであれば、生成側面をもつ各オペレーションについて、その「親」がいつオペレーションを受け容れるかに関連付けられた何らかの基準を同定する；問題の「親」の子孫項目を生成するために必要とされる情報を含む、そのオペレーションに関連付けられた引数型を定義する。

【００６６】
コード生成器４０は本発明の実施形態に任意的に含まれるもので、ブラックボックス２０によって生成されるインターフェース明細（抽象API　３０）に基づいてXMLコード、APIまたは他のソフトウェア製品のうちの一つまたは複数を生成できる。
【００６７】
図２は、クラスの型どうしの間の関係とそれらがサポートするオペレーションを示す図である。図２では、クラスは楕円の外形を、インスタンスは長方形の外形を与えられている。矢印はクラス間またはクラスとオペレーションの間のリンクを表し、単語は関係の型を範疇分けするものである。
【００６８】
図２から、クラス「オペレーション」があることが見て取れる。オペレーションの型は「get」、「post」、「put」および「delete」である。クラス「Getable」は型「get」のオペレーションをサポートする。つまり、クラス「Getable」に属する項目は、「get」オペレーションが実行されることができる。クラス「Getable」が「get」オペレーションをサポートするのと等価な仕方で「post」オペレーションをサポートするクラス「Postable」がある。同様に、クラス「Putable」は「put」オペレーションをサポートし、「Deletable」は「delete」オペレーションをサポートする。
【００６９】
「Resource」〔リソース〕は「Getable」のサブクラスである。すなわち、クラス「Resource」に属する項目「resource」は、上位クラス「Getable」からオペレーション「get」をサポートする属性を継承しているので、本来的に「Getable」である。Resourceのサブクラスであるクラス「Container」〔包含者〕が与えられており、よってその上位クラスResourceからオペレーション「get」をサポートする属性を継承する。クラス「Container」に属する項目は、「含む」関係によってクラス「Resource」に属する項目にリンクされることができる。すなわち、container項目はresource項目を含むことができる。
【００７０】
クラス「Progenitor」〔親〕は、クラス「Container」のサブクラスであり、よってやはりオペレーション「get」をサポートする属性を継承する。クラス「Progenitor」はクラス「Postable」のサブクラスでもあり、よってオペレーション「post」をサポートする属性を継承する。クラスProgenitorのこれら二つのサブクラス関係は、クラス「Progenitor」の項目が「Postable　Container」〔ポスト可能な包含者〕とも呼ばれることができることを意味する。「Progenitor」クラスは、resource項目を含むことができるのは、resource項目が、そのprogenitor項目についての子孫項目として定義されるresourceの型またはresourceのサブクラスである場合にのみであるという点で、「Resource」クラスと特別な関係をもつ。その関係は、has_progeny〔子孫をもつ〕関係であり、よってprogenitorはresourceを子孫（progeny）としてもつ。この関係を支配する処理規則は、含むリンクを支配する処理規則とは異なる。さらに、クラス「Progenitor」はクラス「Resource」に「accepts_progeny」〔子孫を受け容れ〕と記される関係を介してリンクされている。この関係は、progenitor項目が、（特定の型またはサブクラスの）resource項目がprogenitor項目にポストされることを要求するpostオペレーションを受け容れることができることを表し、それに応答してprogenitor項目は子孫項目として問題のresource項目の型のインスタンスを生成できる。
【００７１】
クラスResourceは、クラスContainerと同様、HTTPオペレーションGET、DELETEおよびPUTをサポートする。クラスProgenitorはHTTPオペレーションGET、DELETE、PUTおよびPOSTをサポートする。
【００７２】
図３は、ドメイン固有知識１をインターフェース明細６に転換するプロセスおよび中間段階を表す図である。最初、ドメイン知識１が操作されて諸オントロジー３にされる。ドメイン知識１と諸オントロジー３の間の相違は、事実上、オントロジー３はドメイン知識

１を取り込み、それをドメイン独立なデータ構造およびオペレーション構造と整合する仕方で呈示するということである。変換エンジン４（これは図１のブラックボックス２０およびコード生成４０を組み込んでいる）は、データ構造オントロジーおよびオペレーショナル・オントロジーから導出された規則のセットをオントロジー３に適用してアプリケーション・スケルトン５およびインターフェース明細６を生成することができる。ドメイン知識１からオントロジー３を形成するプロセスは、インターフェース明細６を生成するために十分なドメイン知識１があることを保証するためにも使われることができる。

【００７３】

ドメイン知識１からオントロジー３を生成するプロセスは、制約されたエディタ２を使って実行されてもよい。ドメイン知識は、クラス定義およびクラス間のリンクを記述する仕方を含んでいてもよい。クラス間のリンクを記述する仕方はリンクの属性を含む。オントロジー３はドメイン独立データ構造およびオペレーション構造と整合する。実際、オントロジー３は、ドメイン・オントロジー、データ構造オントロジーおよびオペレーショナル・オントロジーであってもよい（図１に示されるように）。すなわち、ドメイン内の諸項目の諸クラスおよびそれらの間の関係およびそれらがサポートするオペレーションは、図２に例示したデータ構造およびオペレーション構造および／または図１に関係してデータ構造オントロジーおよびオペレーショナル・オントロジーとして記述された構造に基づいている。

【００７４】

形式的に、諸オントロジー３は統一され、OWL（ウェブ・オントロジー言語）で呈示されてもよい。制約されたエディタ２は、ユーザーが、オントロジー文書をどのようにして形式的に構築すべきかを知る必要なしに、何らかの文献記載されたフォーマットで、たとえばOWL文書としてオントロジー３を生成することを許容する編集環境である。制約されたエディタ２はグラフィカル・インターフェースを含んでいてもよく、該グラフィカル・インターフェースにおいて、ドメイン知識が、事実上構築ブロックを形成する基本的クラスのセットに適用されることができる。制約されたエディタは、ユーザーを、ドメイン独立なデータ構造および／またはオペレーション構造と整合するオントロジー３を生成することに拘束することができ、あるいはオントロジー３がそのように生成されることを強制してもよい。諸オントロジー３は概念的に異なっている（ドメイン独立なオントロジーとドメイン従属なオントロジーがある）が、これらのオントロジーを統合する単一の文書として、あるいはドメイン独立なオントロジーおよびドメイン従属なオントロジーの組み合わせから帰結する多数の情報として呈示されてもよい。

【００７５】

普遍的な対話セマンティクスおよびモデリング・パターンがドメイン独立なデータ構造およびオペレーション構造（これらはまとめてコア・オントロジーまたは一般オントロジーと称されることもできる）に取り込まれる。特定のドメインについてのオントロジー３を生成することは、データ構造およびオペレーション構造をインポートし、参照することを含んでいてもよい。オントロジー３から実装可能なインターフェース明細６および他の製品５（サーバー側実装コードの諸ビット）へのステップは、変換エンジン４内で行われる自動化されたプロセスであってもよい。

【００７６】

対話セマンティクス（interaction semantics）は、ドメインの全体的なランタイム状態を（インターフェース明細を実装するAPIのインターフェースを介して）いかにして変更できるかを指定する規則と考えることができる。ドメインのランタイム状態はグラフ（たとえばリソースおよびリソース間のリンクのマップ）によって表現できる。普遍的な対話セマンティクスは、一般的な用語において、グラフの形を変えるためにクライアントができることを取り込み、これらはその後、たとえば制約されたエディタ２によって、特定のドメインに適用されることができる。対話セマンティクスの例は、どのクラスが削除されることができるかおよびどのようにして新しいリソースが追加できるかを指示することを含む。

【００７７】
制約されたエディタ２は、ドメイン独立なデータ構造およびオペレーション構造の制約およびパターンをエンコードおよび表現できるグラフィカル・エディタとして実現されてもよい。その際、オントロジー３において取り込まれるユーザーのドメイン知識が、データ構造およびオペレーション構造の規約に従う。制約されたエディタ２を用いるユーザーは、OWLのような言語を使ってドメインを表現するオントロジー文書を生成する複雑さから効果的に隔離される。制約されたエディタをバイパスする場合、ユーザーは自分で、データ構造およびオペレーション構造の制約条件が従われていることを保証する必要がある。

【００７８】
変換エンジン４による変換のプロセスは、アプリケーション・スケルトンのコードおよびインターフェース明細を生成してもよい。アプリケーション・スケルトンは、インターフェース明細の機能する実装に向けての出発点として使用できる。そのような実装のプログラミング・スタイルはイベント駆動であり、オントロジー３のクラスにイベント・ハンドラが取り付けられる。

【００７９】
ここで、本願では、ドメイン（domain）は環境（environment）の一般的な型を表すために使われていることを注意しておくことが有用であることがありうる。特定のプロバイダーのドメインは環境と称されることになる。よって、ドメインは環境の抽象化である。

【００８０】
ドメイン独立なデータ構造およびオペレーション構造の確立は、異なるドメインを横断して繰り返し生起する観察から、構造中に組み込まれるべきパターンを抽出することによって可能にされる。データ構造およびオペレーション構造はこれらのパターンを形式化する。データ構造およびオペレーション構造において形式化されるいくつかの有意なパターンについてこれから述べる。

【００８１】
図４のＡでは、クラスおよびサブクラスの性質が示されており、属性「受け容れる」および「含む」が例示されている。図４のＡないしＣにおいて、長方形はクラスを示し、角丸のボックスはクラスのインスタンスを表す。図４のＡにおいて、Bはクラスであり、サブクラスBxおよびByはクラスBに属する。Aは図４のＡには示されていないクラスであるが、項目aはクラスAのインスタンスである。項目b1およびb2はクラスByのインスタンスである。図４のＡのオブジェクト間の関係は、矢印および付随するテキストによって表されている。先述したように、クラスBxおよびByはクラスBのサブクラスである。クラスAのインスタンスは、この時点で、クラスBx内の項目を受け容れる。しかしながら、それより前の時点では、AはクラスByを受け容れた。項目aはクラスAのインスタンスであり、オペレーション（メソッド）GETおよびPOSTをサポートする。つまり、クラスAはゲット可能（getable）かつポスト可能（postable）なサブクラスである。項目aは項目b1およびb2を含む包含者である。b1およびb2はいずれもクラスByのインスタンスである。項目b1およびb2はオペレーションGETをサポートし、リソースもそうである。

【００８２】
「含む」は属性であり、二つの項目間の関係の型である。包含の原理は、所与のドメインに内在する階層構造を画定する。属性「含む」は、同じドメインのサブクラスに適用されるべきであり、どの項目が何を含むことができるかの範囲は、問題のドメインに依存してさらに制約されてもよい。属性としての「包含者」の使用は、二つのクラスの間の関係の包含特性を示す。含まれる項目（要素）へのリンクのセットをホストするリソースは、暗黙的に包含者である。図４のＡにおいて、項目aはクラスAのリソースである（それはオペレーションGETをサポートする）。ドメインのオントロジー（図４のＡにおけるドメインは抽象的である）は、クラスAとそれが含むことのできるリソースの型との間の「含む」関係を宣言する。この場合、クラスAは、クラスBおよびそのサブクラスをもつリソースを含むことができる（このことは図４のＡだけから導出できるものではないが）。項目a

はGETおよびPUTの両方をサポートするので、親項目である。

【００８３】

項目の集合（collection）であるほか、包含者は、該集合に新しい要素を受け容れることもできてもよい。包含者aがPOSTをサポートし、よってクラス「親」に属すると分類することは、クラスAのインスタンスが諸項目を受け容れることができると指定する。要求はエンティティであり、要求はPOSTオペレーションを介して親によって受領される。要求エンティティは、親aにおける集合中に入れられるべきリソースを記述する。受け容れられる場合（「受け容れる」関係に適用される基準があってもよい）、新しい実際のリソースが生成され、集合中に入れられる。

【００８４】

いくつかの場合には、受け容れポリシー（要求エンティティに適用される基準）がランタイムにおいて、たとえば環境変数およびリソースの状態に基づいて、さらに制約されることができる。しかしながら、「含む」ポリシーは、何が集合内のメンバーとして現れるかについての設計時の制約である。たとえば、「管理可能な」リソースのライフサイクルがリソースに関連付けられた状態の集合を通じて制御されるとき、「状態」集合は一般に状態（state）リソース（クラス「状態（State）」のリソース）を含むことができ、そのライフサイクルの特定の時点において、それを「開始する（start）」ためのStartStateリソース（Stateのサブクラス）を受け容れることができる。属性「受け容れる」は、所与の「ポスト可能な包含者（postable　container）」（親）リソースについて、このランタイムのメンバーシップ情報を宣伝するために使われる。

【００８５】

図４のＢは、リンクするクラスの概念を示している。図４のＢにおいて、項目aはクラスAのインスタンスであり、オペレーションGETおよびPOSTをサポートする。項目aは、オペレーションGETおよびPOSTをサポートするので親項目である。項目blink〔点滅〕はリンクするクラスのインスタンスである。項目bはクラスBのインスタンスである。項目aは項目blinkを含み、blinkは項目bにリンクされている。

【００８６】

包含は、含まれるリソースの観点からは、二つの項目の間の関係に、排他性があることを含意する。二つの別個のリソースが論理的に同じ特定のリソースを含みうるように見える場合には、モデルは因子分解され直す必要がある。この状況を扱うためのアプローチが、「リンクする」クラスを導入することである。図４のＢに示される例では、各blinkリソースは（クラスAをもつ）単一のリソース内に排他的に含まれる。その結果、（クラスBをもつ）リソースbはクラスAの複数のインスタンスに関連付けられることができる。

【００８７】

図４のＣは、非均一な要素の包含を宣言するクラスを扱うために包含の追加的な層を使う概念を示している。図４のＣにおいて、項目aはクラスAのインスタンスであり、オペレーションPOSTをサポートする。項目bはクラスBのインスタンスであり、オペレーションPOSTをサポートする。項目cはクラスCのインスタンスであり、オペレーションPOSTをサポートする。項目aは項目bおよびcを含む包含者である。項目cは項目c2を含む包含者であり、項目bは項目d1およびd2を含む包含者である。

【００８８】

非均一な項目の包含を宣言するクラスについては、本発明のデータ構造およびオペレーション構造はオントロジーを、その階層構造中に、包含者クラスの追加的なセットを含むよう制約してもよい。包含者内部の要素が均一であることを保証するためである。これは、均一なクラスのインスタンスが、ランタイムにおいて共通の処理規則を使って扱うことができるので、有利である。包含者において要求すべき均一性のレベルに関する決定は、データ構造中に変更できないよう組み込まれてもよいし、あるいはユーザー（アプリケーション開発者）に任されてもよい。しばしば、包含者中への有効なエントリーを指定するために、有効なスーパークラスが使われる。

【００８９】

データ構造およびオペレーション構造は、情報モデリング空間においてこれらのパターンを使うこと（オントロジー生成プロセス）に向けたアプローチを形式化し、それによりオペレーション・レベルでのソフトウェア製品（たとえばHTTPベースのAPI）が許容される。オントロジーからインターフェース明細および他の製品への変換は、インターフェース明細を構築するために、推定（inferencing）および推論（reasoning）といったいくつかの内部プロセスを必要とする（規則のセットがオントロジーに適用される）。

【００９０】

図５Ａないし図５Ｄは、オントロジー生成へのアプローチが、仮想システムのグループについてオントロジーを生成する場合にどのように適用されうるかを例示している。

【００９１】

VSYSs　１００は親項目であり、その子孫項目はVSYS　１１０ａ～ｃの項目である。VSYSs　１００の表現を受け取るHTTP　GETオペレーションは、たとえば、VSYSs　１００に含まれるVSYS子孫項目１１０ａ～ｃのリストを返してもよい。VSYSs項目１００に対して実行されるHTTP削除オペレーションは、含まれるVSYS項目１１０ａ～ｃのすべてを再帰的に除去し、VSYSs　１００項目を除去してもよい。

【００９２】

図５Ｂは、VSYSs項目１００に対して実行されるHTTP　POSTオペレーションを示している。クライアント・アプリケーションが、HTTP　POSTオペレーションを使った要求エンティティ１１１をVSYSs項目１００に提出する。要求エンティティ１１１のボディは、生成されVSYSs項目１００内に含まれることを望む子孫項目（この場合VSYS項目）を記述する。次いでサーバーは、親VSYSs　１００内に新しいVSYS項目１００ｄを生成し、その新しい項目にURLを割り当てる。VSYSs　１００は、その子孫項目はクラスVSYSの項目であるという制約条件をもつ親項目である。

【００９３】

図５Ａないし図５Ｄに表されるオントロジーが生成される仮想コンピューティング・ドメインのコンテキストでは、新しいVSYS子孫項目をポストすることは、現実には、コンピューティング・リソースの既存のグループ内に仮想システムを創り出してもよく、該仮想システムは、クライアント・アプリケーションによって完全なVSYS（仮想システム）として扱われてもよい。

【００９４】

図５Ｃは、クラスVSYS　１１０の項目についての枠組みを与えている。VSYS　１００（仮想システム）は、五つの親（progenitor）、ネットワーク（Networks）１２０、スナップショット（Snapshots）１３０、EFMs１４０、Vサーバー（VServers）１５０および状態（States）１６０を含む包含者項目である。VSYS　１１０はまた、他の項目へのリンクのリスト１８０を含んでいてもよい。該リンクは図５CのVSYS　１１０の場合には、VSYSDescriptor項目へのリンクであり、「basedOn」クローズ（clause）によって特徴づけられている。さらに、VSYS　１１０は、単純なデータ・オブジェクトを含むフィールドのリスト１９０を含む。これらのフィールドは「スロット」とも称されることができる。フィールドは編集可能であっても、読み出し専用であってもよい。この例では、「creationTime」〔生成時刻〕フィールドは読み出し専用であり、その一方、「description」〔説明〕フィールドは編集可能である。「description」フィールドの編集を可能にするために、VSYS項目１１０上で編集リンクが公表される。編集リンクは、「description」と呼ばれる編集可能なフィールドがあるという事実を、クライアント・アプリケーションに利用可能にする。次いで、（HTTP）　PUTオペレーションを使ってフィールド値を編集できる。

【００９５】

図５Ｄは、状態（States）親項目１６０を詳細に示している。「状態」親項目１６０はVSYS包含者１１０に含まれる。「状態」親項目１６０は、それが含まれているVSYSインスタンス（項目）１１０の状態を管理するためのリソースである。「状態」親項目１６０に追加された最も最近の状態項目は「Running」〔実行〕状態１６２であった。「状態」親項目１６０は、クラス「状態（State）」に属する項目を生成する要求のみを受け容れる

ことができることによって、制約される。したがって、VSYS　１１０は、状態項目「Running」に関連付けられた状態にある。クラス「状態」に属する項目の二つの例は、「実行」１６２および「移行」〔Migrating〕１６３である。「状態」親項目１６０内にクラス「状態」の新しい項目を生成し、よって「状態」親項目１６０を含んでいるVSYS項目１１０の状態を変えるために、「状態」親項目１６１に対してPOSTオペレーションが実行されることができる。

【００９６】

クライアント・アプリケーションは、HTTP　POSTオペレーションを使って要求エンティティ１６１を「状態」親項目１６０に提出することによって、VSYS項目１１０の状態を管理できる。要求エンティティ１６１のボディは、生成されることを望む子孫項目（この場合、「移行」状態項目）を記述する。「状態」親項目１６０は、要求エンティティ１６１を受け容れ、「状態」親項目１６０内に新しい「移行」状態項目１６３を生成できる。ひとたび「移行」状態項目１６３が生成されると、VSYS　１１０は移行状態になる。「実行」状態項目１６２は、「状態」親項目１６０内に保持されるので、VSYS　１１０の状態をこのようにして管理することにより、「状態」親項目１６０内で状態履歴が維持される。新しい状態のオペレーション詳細はサーバーによってホストされてもよく、よって要求エンティティ１６１は、新しい状態の有効な名前しか含まないことができる。任意的に、非瞬間的な状態変化をサポートする非同期的な状態管理が想定される。

【００９７】

図６は、クラウド・コンピューティング・ドメインにおけるコンピューティング・システムがどのようにしてデータ構造に基づいて表現されうるかの例である。「システム」リソース２００は「ネットワーク」リソース２１０（これはネットワークへのリンクである）、「名前」フィールド２２０（これは読み出し専用である）および「サーバー（Server）」リソース２３１ａ～ｃを含む「サーバー（Servers）」親項目２３０を含んでいる。「サーバー」親項目２３０は、新しいサーバー項目をPOSTする要求を受け容れ、応答して、新しい「サーバー」リソースを生成する。「サーバー」２３０は、サーバー数に対する所定の限界に達したときに新しいサーバーをPOSTする要求を受け容れるのを止めてもよい。

【００９８】

図７Ａないし７Ｃは、本発明の諸実施形態に方法を実行するためのコンピュータ・プログラムによって出力される表示の表現を呈示している。表現されているドメインは、クライアントが、要求を介してサーバー上の集合にノート（Postie）を追加できるドメインである。サーバーは集合（Posties）中にノートを記憶する。

【００９９】

図７ＡないしＣにおける、オントロジー文書およびインターフェース明細が生成されたドメインは、二つのクラスPostieおよびPostiesが存在する単純なドメインである。この例におけるドメイン知識は、ドメインを記述する単純事実の呈示であってもよい。たとえば：postie項目はposties包含者に記憶されることができる；postie項目は削除されることができる；postie項目はposties包含者内に生成される；postiesおよびpostie両方の表現がクライアントに返されることができる、といったことである。

【０１００】

図７Ａは、ドメイン・ビューを表している。これは、グラフィカル・ドメイン・オントロジー表現３０１を探索するためのルート表示である。これは、上に詳述した単純な事実の一部またはすべてを取り込む、OWL文書または他の何らかのオントロジー文書から導出されたものであってもよい。ドメイン・ビュー表現３０１は、本発明のデータ構造およびオペレーション構造が適用されたドメイン知識を表現する。

【０１０１】

図７Ａないし７Ｃに描かれるビューは、インターフェース明細の最終的なビューを表していることを注意しておく。これは、コード生成（コード生成器）４０または変換エンジン４のいずれかからの出力の可能な形を例示する。

【０１０２】
メニュー・バー３４０は、ドメイン・オントロジー表現３０１の、特定のクラスまたは他の要素に焦点を当てる他のビューへのリンクを含む。ビューの右側では、「説明文」領域３５１がそのドメインを要約する何らかのテキストを表示する。
【０１０３】
メニュー・バー３４０において、「すべて表示」ボタン３４１は、図７Ａに表示されるドメイン・ビューにリンクし、「Posties」ボタンは図７ＣのPostiesビューにリンクし、「Postie」ボタンは図７ＢのPostieビューにリンクする。メニュー・バー３４０はすべてのビューにおいてアクセス可能である。
【０１０４】
ドメイン・ビューにおいて、ユーザーがそのドメインの全体的な記述を与えることのできる「説明文」領域３５１が設けられる。ドメイン・オントロジー表現が生成されたプロセスに依存して、以前の何らかの段階でユーザーによって入力されたドメイン記述に基づいて、コンピュータ・プログラムが自分で説明文領域のためのテキストを導出できるのであってもよい。
【０１０５】
この例では、説明文３５１はたとえば「これは単なる例。個々のpostieを含むpostieの集合がある。Postiesは親であり、postie項目を生成する命令を受け容れることができる。」といったものであってもよい。
【０１０６】
ドメイン・オントロジー表現３０１には項目の二つのクラスがある。Posties　３００は（HTTP）GETおよびPOSTオペレーションの両方をサポートする親である。ハイライター３３０は単に、Postiesが現在選択されており、リターン・キーを押すとPostiesビューに移行することを表している。矢印３１０はPosties　３００をPostie　３２０にリンクする。これは、これらのクラスの間に関係があることを示している。クラスとしてのPostie　３２０は（HTTP）オペレーションGETおよびDELETEをサポートする。Posties　３００およびPostie　３２０についてのボックスは、ビュー間で移動するためにメニュー・バーと同様の仕方で使われてもよい。
【０１０７】
図７Ｂは、クラスPostieのクラス・ビューであるPostieビューを表している。ドメイン・オントロジー表現３０１はスケール・ダウンされているが、ドメイン・ビューと同じ情報を含んでいる。同様に、メニュー・バー３４０はドメイン・ビューと同じである。
【０１０８】
クラス・ヘッダ３５２およびクラス・サブヘッダ３５３は、クラスの名前およびその当該ドメイン内での役割の非常に短い要約を示す。サブヘディングの内容はユーザーによって入力されてもよいし、あるいはドメイン記述またはオントロジー文書からコンピュータ・プログラムによって導出されてもよい。この場合、クラス・ヘッダ３５２は「Postie」となっており、サブヘッダ３５３は「Postiesの子孫」となっている。明細セクション３５０ａは、処理エンジンの出力を表示する。処理エンジンは、この実施形態では、前記コンピュータ・プログラムの一部である。これらの出力は、ユーザーにより、APIまたは他のソフトウェア製品として容易に実装できる。
【０１０９】
URIテンプレート領域３５４は、ユーザーのために、クラス・ヘッダ３５２によって示されるクラスの項目についてのURIテンプレートを提供する。この例では、URIテンプレート領域３５４はテキスト：
「URIテンプレート（これは単に提案）
　/Posties
　　/{Postie}」
を含んでいてもよい。テキスト「Posties」は、Postiesビューへのクリック可能なリンクであってもよい。

【０１１０】
オペレーション・プロトコル領域３５５は、ユーザーに、クラス・ヘッダ３５２によって示されるクラスの項目についてのHTTPオペレーションの要求の処理をどのようにコーディングするかを案内するテキストを与える。今の例では、クラスとしてのPostieはHTTPオペレーションGETおよびDELETEをサポートするので、オペレーション・プロトコルは：
「Operations
GET <item url='..' class='Postie'>
<date>dateTime</date>
<text>string</text>
</item>
DELETE」
と書かれていてもよい。
【０１１１】
すなわち、オペレーション「GET」について、要求は、引数を必要とせず（ブランク要求）、上に詳述したタグ付けされたテキスト（<itemから/item>まで）はこの要求に応答してサーバーによって生成される応答のフォーマットを表す。応答は、GETオペレーションが実行される対象のpostieのストリングだけでなく、postieの生成時刻を表すdateTimeというデータ・オブジェクトをも含む。同様に、DELETEは引数を必要とせず、サーバーも応答を生成しない。よって、DELETEオペレーションに関しては何のテキストもない。
【０１１２】
属性（Properties）領域３５６は、各クライアントがどの属性をもつかおよびこれらの属性がどの単純なデータ・オブジェクトであるかについてユーザーに案内するためのテキストを与える。今の例では、テキストは
「Properties（ドットは読み出し専用を示す）
date・ dateTime
text string」
と書かれていてもよい。
【０１１３】
図７Ｃは、クラスPostiesのクラス・ビューであるPostiesビューを表している。ドメイン・オントロジー表現３０１はスケール・ダウンされているが、ドメイン・ビューと同じ情報を含んでいる。同様に、メニュー・バー３４０はドメイン・ビューと同じである。
【０１１４】
クラス・ヘッダ３５２およびクラス・サブヘッダ３５３は、クラスの名前およびその当該ドメイン内での役割の非常に短い要約を示す。サブヘディングの内容はユーザーによって入力されてもよいし、あるいはドメイン記述またはオントロジー文書からコンピュータ・プログラムによって導出されてもよい。この場合、クラス・ヘッダ３５２は「Posties」となっており、サブヘッダ３５３は「Postieの親」となっている。明細セクション３５０ｂは、処理エンジンの出力を表示する。処理エンジンは、この実施形態では、前記コンピュータ・プログラムの一部である。これらの出力は、ユーザーにより、APIまたは他のソフトウェア製品として容易に実装できる。
【０１１５】
URIテンプレート領域３５４は、ユーザーのために、クラス・ヘッダ３５２によって示されるクラスの項目についてのURIテンプレートを提供する。この例では、URIテンプレート領域３５４はテキスト：
「URIテンプレート（これは単に提案）
/Posties」
を含んでいてもよい。
【０１１６】
オペレーション・プロトコル領域３５５は、ユーザーに、クラス・ヘッダ３５２によって示されるクラスの項目についてのHTTPオペレーションの要求の処理をどのようにコーデ

ィングするかを案内するテキストを与える。今の例では、クラスとしてのPostiesはHTTPオペレーションGETおよびPOSTをサポートするので、オペレーション・プロトコルは：
「Operations
GET <item url='..' class='Posties'>
<postie url='..' class='Postie'/>
</item>
POST この集合内のPostie子リソースを生成
<item class='Postie'>
<text>string</text>
</item>」

ここでもまた、GETオペレーションは引数を必要とせず、「GET」に続くテキスト（<itemから/item>までの間）は、Posties親に対するGET要求を受領するサーバーによって生成される応答のフォーマットを表す。オペレーションPOSTのほうは引数（string）をもち、「POST」に続く<itemから/item>までの間のテキストは実は、POSTオペレーションを介して新しいpostieを生成する要求が取るべきフォーマットである（stringはpostieについてのストリング）。任意的に、そのような要求に対する応答についてのフォーマットを表すテキストが表示されることができる。しかしながら、そのような応答のフォーマットが上に詳述した「Postie」クラスについてのものと同じであり、

```
<item url='..' class='Postie'>
<date>dateTime</date>
<text>string</text>
</item>
```

の形をもつ応答がPosties親における各postie項目について生成されることはユーザーによって知られているであろう。

【０１１７】

属性（Properties）領域３５６は、各クライアントがどの属性をもつかについてユーザーに案内するためのテキストを与える。ただし、この場合、親項目はPostie項目しか含まず、追加的なフィールドはもたない。よって、属性領域３５６はブランクである。

【０１１８】

クラスPostiesは親クラスなので、追加的なテキスト領域、子領域３５７が設けられる。子領域３５７は、クラス・ヘッダ３５２によって示されるクラスが親となるクラスの詳細を与える。この例では、テキストは、
「Progenitor of
postie Postie」
と書かれていてもよい。ここで、テキストの第二の部分「Postie」はPostieビューへのリンクである。

【０１１９】

図８のＡおよびＢは、それぞれクラスPostieおよびPostiesに関係する明細セクション３５０ａおよび３５０ｂの詳細なビューを与える。クラス・ヘッダ３５２およびクラス・サブヘッダ３５３領域も各例において表示されている。

【０１２０】

オペレーショナル・オントロジーのオペレーションは、HTTPオペレーションであってもよいが、これは任意的である。

【０１２１】

上記の側面のいずれにおいても、さまざまな特徴がハードウェアにおいて、あるいは一つまたは複数のプロセッサ上で走るソフトウェア・モジュールとして実装されうる。ある側面の特徴は、他の側面のいずれに適用されてもよい。

【０１２２】

本発明はまた、本稿に記載された方法のいずれかを実行するためのコンピュータ・プログラムまたはコンピュータ・プログラム・プロダクトならびに本稿に記載された方法のい

ずれかを実行するためのプログラムを記憶しているコンピュータ可読媒体をも与える。本発明を具現するコンピュータ・プログラムは、コンピュータ可読媒体上に記憶されてもよいし、あるいはたとえば、インターネット・ウェブサイトから提供されるダウンロード可能なデータ信号のような信号の形であることもできるし、他の任意の形であることもできる。

【０１２３】

特に、プロセッサを有するコンピューティング・デバイスによって実行されたときに、該コンピューティング・デバイスに本発明を具現する方法を実行させるコンピュータ・プログラムが提供されてもよい。

【０１２４】

（付記）

以上の実施例を含む実施形態に関し、さらに以下の付記を開示する。

（付記１）

ドメイン内の項目の諸クラスの構造、どの項目がどのクラスに属するかおよびどのオペレーションを各クラスがサポートするかを指示するオントロジー文書を生成する、コンピュータ実装される方法であって：

ドメイン内の項目および項目間の関係についての情報を含むドメイン記述を入力する段階と；

データ構造およびオペレーション構造にアクセスする段階であって、データ構造は前記ドメイン記述において記述される項目を諸クラスに分けて特徴づけるための項目および項目間関係のドメイン独立なモデルであり、データ構造は諸項目のクラスとして少なくとも親項目を含み、各親項目は該親項目がそれに対して呼び出されたオペレーション構造からのオペレーションを受け容れる結果として該親項目によって生成される子孫項目だけを含むことができ、オペレーション構造は、ドメイン記述に適用されたときに諸項目の諸クラスについての処理規則を定義するオペレーションの、ドメイン独立なリストを含む、段階と；

前記データ構造および前記オペレーション構造を前記ドメイン記述に適用して、ソフトウェア製品を生成するのに使うオントロジー文書を生成する段階とを含む、

方法。

（付記２）

前記ドメイン記述が処理エンジンに入力され、前記データ構造およびオペレーション構造が前記処理エンジンによりアクセスされ、前記データ構造およびオペレーション構造が前記処理エンジンにおいて前記ドメイン記述に適用され、

前記オントロジー文書が前記処理エンジンによって生成される、

付記１記載のコンピュータ実装される方法。

（付記３）

親項目の各インスタンスは、項目の特定のクラスのインスタンスである子孫項目だけを含むことができる、付記１または２記載のコンピュータ実装される方法。

（付記４）

前記オントロジー文書に基づいてURIテンプレートを処理エンジンにおいて生成し、該処理エンジンから出力する段階をさらに含み、前記URIテンプレートは、ランタイムにおけるドメイン内のリソースの各インスタンスがHTTP　URIを介した参照により取得可能であることを保証するために使うことができ、リソースはフィールドおよび他のリソースへのリンクをもつ項目のクラスを総称する用語である、付記１ないし３のうちいずれか一項記載のコンピュータ実装される方法。

（付記５）

前記オントロジー文書は、管理可能な包含者のクラスを定義し、該管理可能な包含者クラスのインスタンスは状態親項目を含み、状態親項目が含むことのできる子孫項目は、管理可能な包含者のクラスのオペレーティング状態のあらかじめ定義されたリストのうちの一つを指示するために使われる項目の状態クラスのインスタンスである子孫項目のみであ

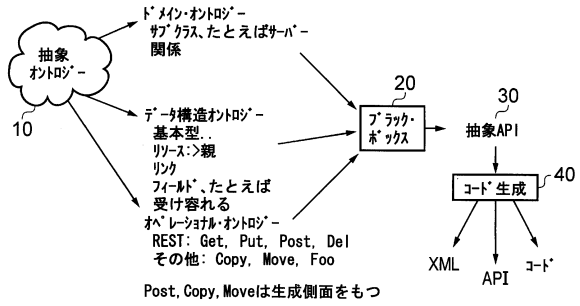
る、付記１ないし４のうちいずれか一項記載のコンピュータ実装される方法。
（付記６）
前記データ構造およびオペレーション構造が制約されたエディタによってアクセスされ、前記制約されたエディタは、ユーザーを、前記データ構造およびオペレーション構造が前記ドメイン記述に適用されるような仕方で前記ドメイン記述を入力することに制約する編集環境である、付記１ないし５のうちいずれか一項記載のコンピュータ実装される方法。
（付記７）
前記オントロジー文書を使って生成されるインターフェース明細は、RESTアーキテクチャ・スタイルにおけるインターフェースの原理と整合する、付記１ないし６のうちいずれか一項記載のコンピュータ実装される方法。
（付記８）
ソフトウェア製品を生成するコンピュータ実装される方法であって、付記１ないし７のうちいずれか一項記載のオントロジー文書を生成する方法を含み、さらに：
処理エンジンにおいて、前記データ構造および前記オペレーション構造に基づく規則のセットを前記オントロジー文書に適用してソフトウェア製品を生成する段階を含む、
コンピュータ実装される方法。
（付記９）
前記データ構造および前記オペレーション構造が、生成されるソフトウェア製品がRESTアーキテクチャ・スタイルにおけるインターフェースの原理と整合することを保証する、付記８記載のコンピュータ実装される方法。
（付記１０）
付記８または９記載のコンピュータ実装される方法であって、前記オントロジー文書は、管理可能な包含者のクラスを定義し、前記管理可能な包含者クラスのインスタンスは状態親項目を含み、前記状態親項目が含むことのできる子孫項目は、管理可能な包含者のクラスのオペレーティング状態のあらかじめ定義されたリストのうちの一つを指示するために使われる項目の状態クラスのインスタンスである子孫項目のみであり、それにより、ランタイムにおける前記管理可能な包含者クラスのインスタンスのオペレーティング状態は、前記管理可能な包含者クラスの前記インスタンスに含まれる前記状態親項目に前記状態クラスのインスタンスを追加することによって管理されるよう動作可能である、方法。
（付記１１）
ランタイムにおける前記管理可能な包含者クラスの前記インスタンスの前記状態は、前記管理可能な包含者クラスの前記インスタンスに含まれる前記状態親項目に対する前記状態クラスの最も最近追加されたインスタンスによって指示され、それにより、前記状態親項目に含まれる子孫項目は、前記管理可能な包含者クラスの前記インスタンスの状態履歴である、付記１０記載のコンピュータ実装される方法。
（付記１２）
前記管理可能な包含者クラスのインスタンスは、仮想コンピューティング・システムである、付記１０または１１記載のコンピュータ実装される方法。
（付記１３）
前記ソフトウェア製品が定義された受け容れ制約条件を含み、前記受け容れ制約条件は、ランタイムにおいて、親項目に対してオペレーションが呼び出された結果として該親項目のインスタンスによって生成されることのできる子孫項目の個別的なインスタンスを制約し、前記受け容れ制約条件はクライアント・ツールに対して利用可能である、付記８ないし１２のうちいずれか一項記載のコンピュータ実装される方法。
（付記１４）
ドメインはクラウド・コンピューティング環境であり、前記ソフトウェア製品はサービスとしてのインフラストラクチャーAPIとして実装されることのできるインターフェース明細である、あるいは前記ソフトウェア製品はサービスとしてのインフラストラクチャーAPIである、付記８ないし１３のうちいずれか一項記載のコンピュータ実装される方法

（付記１５）
ドメイン内の項目の諸クラスの構造、どの項目がどのクラスに属するかおよびどのオペレーションを各クラスがサポートするかを指示するオントロジー文書を生成するよう構成されたコンピューティング装置であって：
ドメイン内の項目および項目間の関係についての情報を含むドメイン記述を入力するための入力手段と；
データ構造およびオペレーション構造にアクセスするアクセス手段であって、データ構造は前記ドメイン記述において記述される項目を諸クラスに分けて特徴づけるための項目および項目間関係のドメイン独立なモデルであり、データ構造は諸項目のクラスとして少なくとも親項目を含み、各親項目は該親項目がそれに対して呼び出されたオペレーション構造からのオペレーションを受け容れる結果として該親項目によって生成される子孫項目だけを含むことができ、オペレーション構造は、ドメイン記述に適用されたときに諸項目の諸クラスについての処理規則を定義するオペレーションの、ドメイン独立なリストを含む、手段と；
前記データ構造および前記オペレーション構造を前記ドメイン記述に適用して、ソフトウェア製品を生成するのに使うオントロジー文書を生成する生成手段とを有する、
装置。
【符号の説明】
【０１２５】
１０　抽象的なオントロジー
２０　ブラックボックス
３０　抽象的なAPI
４０　コード生成器
１００　VSYSs〔仮想システム〕（親項目）
１１０　VSYS（子孫項目）
１１１　要求エンティティ
１２０　ネットワーク（Networks）（親）
１３０　スナップショット（Snapshots）（親）
１４０　EFMs（親）
１５０　Vサーバー（VServers）（親）
１６０　状態（States）（親）
１６１　移行
１６２　実行
１６３　移行
１７０　VSYSDescriptor
１８０　リンク
１９０　フィールド
２００　システム
２１０　「ネットワーク」リソース
２２０　「名前」フィールド
２３０　「サーバー」親項目
２３１　「サーバー」リソース
３００　Posties（親）
３０１　ドメイン・オントロジー表現
３２０　Postie
３３０　ハイライト
３４０　メニュー・バー
３４１　すべて表示
３４２　Posties
３４３　Postie

３５０　明細セクション
３５１　説明文
３５２　クラス・ヘッダ
３５３　サブヘッダ
３５４　URIテンプレート
３５５　オペレーション・プロトコル
３５６　属性
３５７　子

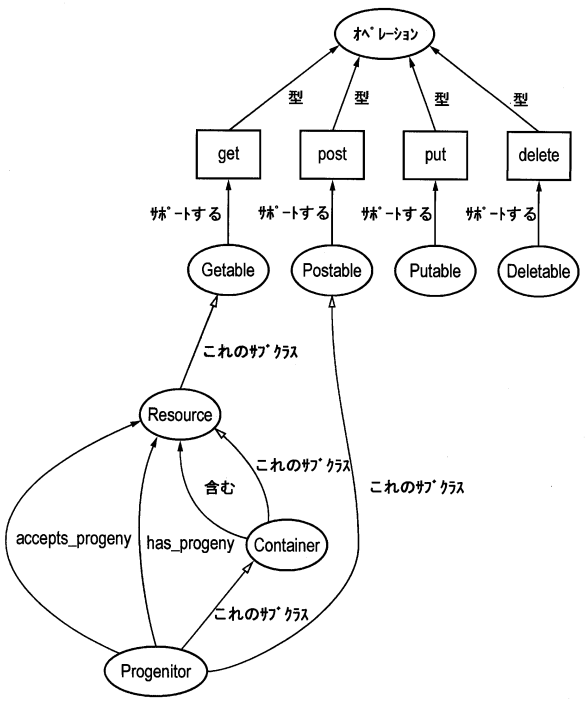
【図１】

本発明を具現するAPIを生成することに向けたアプローチの概念的マップを示す図



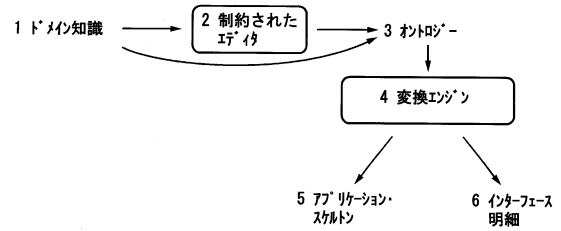
【図２】

オペレーション、クラスの型およびそれらの間のリンクの図的な表現を示す図



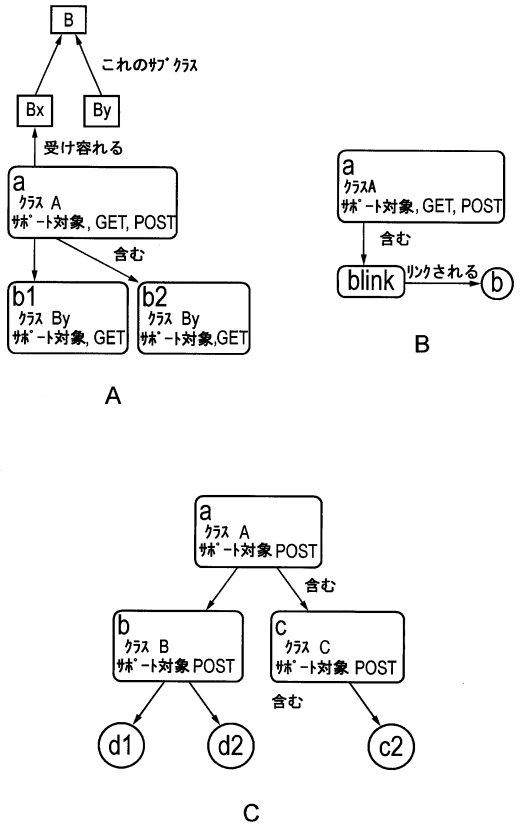
【図３】

本発明を具現するプロセスを示す図



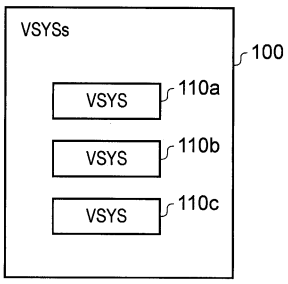
【図４】

Ａ～Ｃは、ドメイン・オントロジーにおいて普通に観察されるパターンを示す図



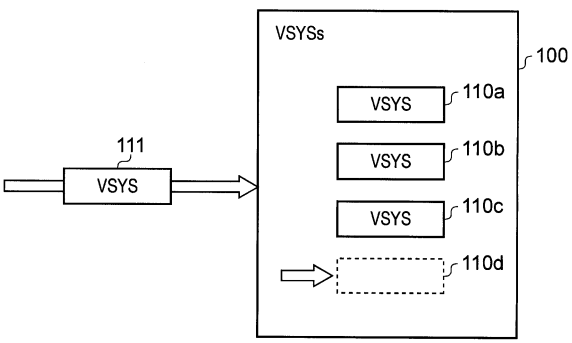
【図５Ａ】

本発明が仮想コンピューティング・ドメインに適用される実施形態における仮想システムのグループの概略図



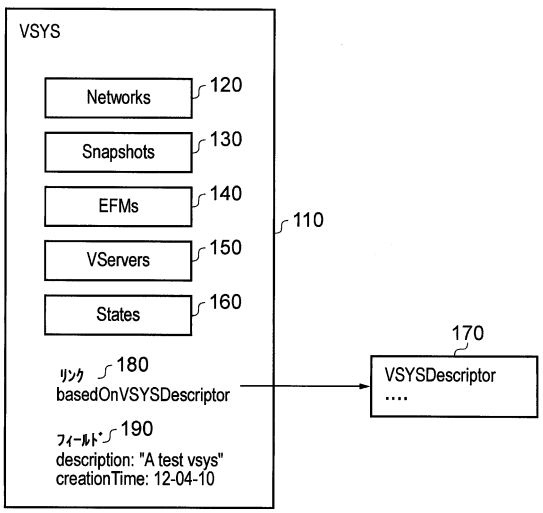
【図５Ｂ】

本発明が仮想コンピューティング・ドメインに適用される実施形態における仮想システムのグループに対して実行されるpost〔ポスト〕オペレーションの概略図



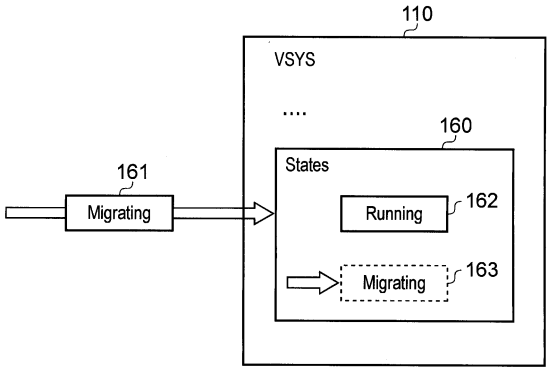
【図５Ｃ】

本発明が仮想コンピューティング・ドメインに適用される実施形態における単一の仮想システム・リソースの概略図



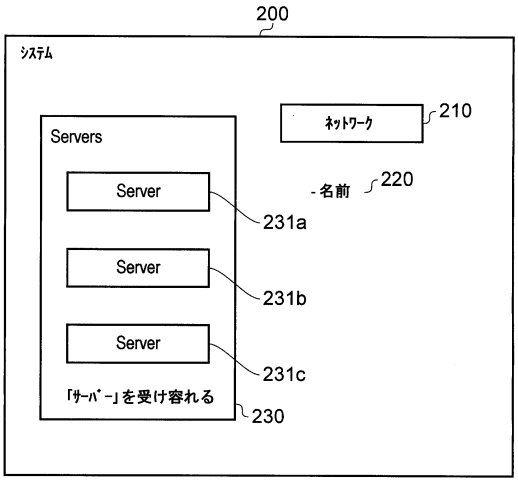
【図５Ｄ】

本発明が仮想コンピューティング・ドメインに適用される実施形態において新しい状態が仮想システム・リソース内の「状態」親項目（States progenitor item）にpostされる概略図



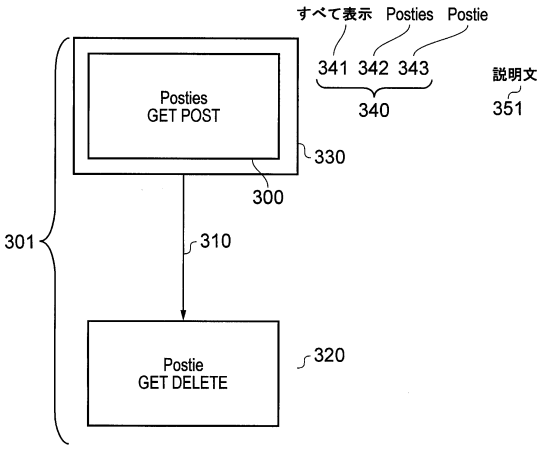
【図６】

本発明の諸実施形態におけるデータ構造を例示する「システム」リソースの概略図



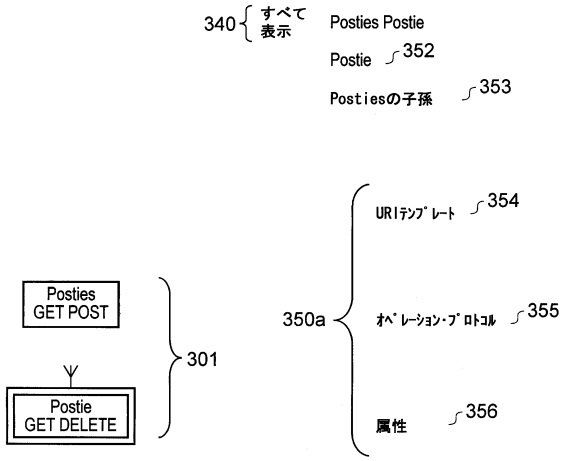
【図７Ａ】

本発明を具現するコンピュータ・プログラムによって出力されるドメイン全体表示の例を示す図



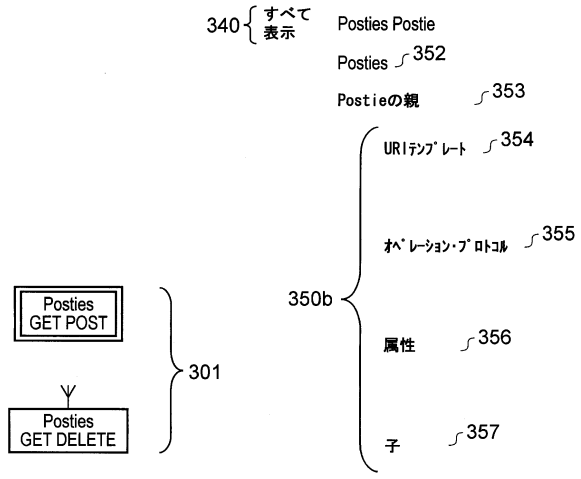
【図７Ｂ】

本発明を具現するコンピュータ・プログラムによって出力される子孫クラス表示の例を示す図



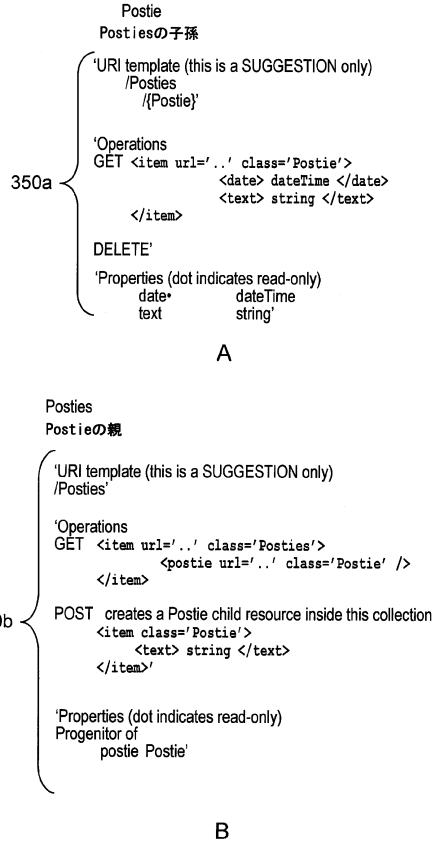
【図７Ｃ】

本発明を具現するコンピュータ・プログラムによって出力される親クラス表示の例を示す図



【図８】

Ａは子孫クラス表示の明細セクションの詳細図である。Ｂは親クラス表示の明細セクションの詳細図



フロントページの続き

(56)参考文献 特開２００８－２６２５５１（ＪＰ，Ａ）
特表２００６－５１０９６８（ＪＰ，Ａ）
特開２００９－０８７３３９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４４
Ｇ０６Ｆ　　３／０４８