(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6081491号
(P6081491)
(45)発行日　平成29年2月15日(2017.2.15)　(24)登録日　平成29年1月27日(2017.1.27)

(51)Int.Cl.　F I
G06Q 50/32 (2012.01)　G06Q 50/32
G06Q 30/04 (2012.01)　G06Q 30/04

請求項の数 11　(全 18 頁)

(21)出願番号　特願2014-552184 (P2014-552184)
(86)(22)出願日　平成24年9月26日 (2012.9.26)
(65)公表番号　特表2015-507278 (P2015-507278A)
(43)公表日　平成27年3月5日 (2015.3.5)
(86)国際出願番号　PCT/US2012/057385
(87)国際公開番号　WO2013/106099
(87)国際公開日　平成25年7月18日 (2013.7.18)
審査請求日　平成27年5月11日 (2015.5.11)
(31)優先権主張番号　13/346,396
(32)優先日　平成24年1月9日 (2012.1.9)
(33)優先権主張国　米国(US)

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(74)代理人　110001195
特許業務法人深見特許事務所
(72)発明者　テキシアー，エマニュエル
アメリカ合衆国、９４０８６　カリフォルニア州、サニーベール、ウェスト・アイオワ・アベニュ、１０６１

最終頁に続く

(54)【発明の名称】イベントを査定するための関数モデル

(57)【特許請求の範囲】
【請求項１】
コンピュータで実行される方法であって、前記方法は、
イベント処理システムにおいて料金プラン指標に関連付けられたイベントを受け取るステップを備え、前記イベントの受け取りに応答して、
前記料金プラン指標に基づき、複数の料金プラン関数オブジェクトから料金プラン関数オブジェクトを識別するステップを備え、
前記複数の料金プラン関数オブジェクトの各料金プラン関数オブジェクトは、複数の料金プランの異なる料金プランに対応し、
前記料金プラン関数オブジェクトは、複数の関数オブジェクトを含み、
前記複数の関数オブジェクトの第１のサブセットは、１つ以上の条件の第１のセットと、前記条件の第１のセットが満たされた場合に実行される第１の動作とを反映し、
前記複数の関数オブジェクトの第２のサブセットは、１つ以上の条件の第２のセットと、前記条件の第２のセットが満たされた場合に実行される第２の動作と反映し、
前記料金プラン関数オブジェクトを実行するステップを備え、前記料金プラン関数オブジェクトを実行するステップは、
前記複数の関数オブジェクトの第１の関数オブジェクトを実行して第１の結果を生成するステップと、
前記第１の結果に基づき、前記複数の関数オブジェクトの第２の関数オブジェクトを識別するステップと、

前記第２の関数オブジェクトを実行して第２の結果を生成するステップと、
前記第２の結果に基づき、複数のアカウントのうち前記イベントに関連付られたアカウントを更新するステップとを含み、
前記方法は１つ以上の演算装置によって行われ、
前記第２の結果を生成するために前記複数の関数オブジェクトのうちの一部の関数オブジェクトが実行される、方法。
【請求項２】
前記実行するステップ、および前記識別するステップは、単一の処理によって行われる、請求項１に記載の方法。
【請求項３】
前記複数の料金プラン関数オブジェクトは、第１の料金プラン関数オブジェクトと第２の料金プラン関数オブジェクトとを含み、
前記第１の料金プラン関数オブジェクトは、特定の関数オブジェクトを含む第１の複数の関数オブジェクトを含み、
前記第２の料金プラン関数オブジェクトは、前記特定の関数オブジェクトを含む第２の複数の関数オブジェクトを含み、
前記特定の関数オブジェクトのインスタンスは、前記第１の料金プラン関数オブジェクトおよび前記第２の料金プラン関数オブジェクトによって共有される、請求項１または２に記載の方法。
【請求項４】
ドメイン固有言語（ＤＳＬ）に準拠する宣言文を受け取るステップと、
前記料金プラン関数オブジェクトを生成するために前記宣言文を処理するステップとをさらに備える、請求項１から３のいずれか１項に記載の方法。
【請求項５】
ユーザによって確立された複数の規則を反映する規則データを受け取るステップと、
宣言文を生成するために規則データを処理するステップとをさらに備える、請求項４に記載の方法。
【請求項６】
前記宣言文はユーザからの入力において反映される、請求項４または５に記載の方法。
【請求項７】
前記料金プラン関数オブジェクトにおける前記関数オブジェクトはいずれも前記アカウントが更新された後に状態情報を格納しない、請求項１から６のいずれか１項に記載の方法。
【請求項８】
前記料金プラン関数オブジェクトにおける特定の関数オブジェクトに関連付けて、前記特定の関数オブジェクトに対する特定の入力に基づいた前記特定の関数オブジェクトの実行結果を格納するステップと、
第２の顧客と関連付けられた第２のイベントを受け取るステップと、
前記第２のイベントの受け取りに応答して、前記実行結果を生成するために前記特定の関数オブジェクトを実行する代わりに前記実行結果を読み込むステップとをさらに備える、請求項１から７のいずれか１項に記載の方法。
【請求項９】
前記イベントは、システムイベント、または顧客による利用を示す利用データを含む利用イベントである、請求項１から８のいずれか１項に記載の方法。
【請求項１０】
請求項１から９のいずれか１項に記載の方法をコンピュータに実行させるためのプログラム。
【請求項１１】
装置であって、
１つ以上のプロセッサと、

１つ以上のプロセッサによって実行された場合に請求項１から９のいずれか１項に記載の方法が行われる命令を格納する１つ以上の記録媒体とを備える、装置。

【発明の詳細な説明】

【技術分野】

【０００１】

発明の分野

本発明は、顧客によるサービスまたは製品の利用の量を反映させるイベントの査定に関する。

【背景技術】

【０００２】

背景

「査定(Rating)」は、概して製品またはサービスの利用に係る費用または価格を判定する処理をいう。また、査定は、課金および変更（もしくは値引き）を分配または共有することをいう場合もある。たとえば、多くの電気通信キャリアは、月／年ベースのプランなど、前払いプランまたは後払いプランであり得る特定のプランを顧客（もしくは加入者）に選択させる。電気通信キャリアによって提供されるサービスの顧客による利用は、査定エンジンに報告される。査定エンジンは、キャリアまたは第三者によって保全され得る。査定エンジンは、顧客のプランおよび顧客の利用に基づいて、顧客に課金する額またはいくらかかるのかを計算し、顧客のアカウントに対して報告する。量は、残りの利用量の合計（たとえば、分またはギガバイトでの利用）から差し引かれ得る。

【０００３】

電気通信の関係において、査定エンジンは、利用の時間（たとえば、太平洋標準時の午後７時）、利用の期間（たとえば、５分の通話）、通話先（たとえば、固定電話、海外、友達、または家族）、および通話の発信元もしくは通話元の位置（たとえば、モバイル通信ネットワークのローミング）などの複数の要因を考慮に入れ得る。インターネットの関係において、要因は、ダウンロードされるコンテンツの量（たとえば、５ギガバイト）、ストリーミングされるコンテンツのタイプ、コンテンツの質、およびダウンロードの時間を含み得る。

【０００４】

料金プランは複雑化しており、頻繁に変更されている。このため、査定エンジンは、このような複雑かつ変化し続ける料金プランに対して利用情報を処理するように構成されなければならない。通常、料金プランの設計者（たとえば、電気通信キャリアの従業員）は、ユーザインターフェイスを表示するユーザ端末（たとえば、デスクトップコンピュータ、ラップトップコンピュータ、またはタブレットコンピュータ）に入力を提供する。入力は、顧客の利用を監視するために使用される量（たとえば、分、通貨、またはデータ利用で）を計算するために使用される規則のセットを反映する。ユーザインターフェイスは、入力を受け入れ、入力に反映される全ての規則を反映するＸＭＬ文書（または表形式の他の文書）を生成し得る。このようなデータ中心表現に係る問題の１つは、これらが料金プランを表現するのに適していないことである。このような表現は、単独でif-then-else条件などの料金プランの関数的な局面を容易に表現することができない。結果として、従来の査定エンジンは、後述するように、追加の関数モジュールを実装してデータ（モジュール、ハードコードされた論理、または規則エンジンのセットなど）を処理する必要がある。

【０００５】

「イベント」は、製品またはサービスを顧客が利用することに応答して生成されるデータである。たとえば、イベントは、顧客が電話による通話を完了した時に作成される。加えて、イベントは、顧客が通話を開始する時、および任意で通話中の１つ以上の時点において作成され得る。

【０００６】

査定エンジンは、リアルタイムまたはオフラインでイベントを処理し得る。リアルタイ

ム処理は、通常、顧客がたとえば通話を行った時またはインターネットにアクセスした時に、査定エンジンがアクセスをリアルタイムで認める必要がある場合に要求され、これによって顧客はサービスの利用を待たなくても良い。製品またはサービスの提供者（電気通信キャリアなど）は、少なくとも収入漏れおよび機会損失という２つの理由により、特定のイベントをリアルタイムで（またはできる限り早く）処理することを望む。収入漏れは、顧客に権限を与えられた利用の範囲よりも多く顧客が利用できることをいう。機会損失は、顧客にある利用量が与えられた時に顧客に利用量を付与しないことをいう。たとえば、前払い電話の関係において、顧客は電話を使用して通信するための分数が限られている。顧客が契約または加入した電気通信キャリアは、顧客に権限が与えられた分（すなわち、顧客の契約に基づいて）を超える分を顧客に利用させたくない。このため、顧客が分を「使い切った」場合、電気通信キャリアは、まず追加の分に対して支払いを行わずに顧客がさらなる通話を行うことを許容したくない。所有する分を超えて顧客が利用するのを防ぐために、電気通信キャリアは、利用要求を却下し、未だ査定されていない全てのイベントが査定されるのを待ち得る。しかしながら、全ての未済のイベントを査定するには時間がかかり得て、サービスを受ける権限を有する顧客に対してサービスの否認を行うことは、結果として不十分な顧客満足を招き得る。この機会損失は、キャリアの金の損失に換算される。

【０００７】

他の例として、後払い電話の関係において、顧客が自身の分を「使い切った」だけでなく、自身のリミットを約１時間超過し、この超過に対して非常に高い料金が課金されることを１日以上後に知ることを顧客は望まない。このため、顧客に与えられた権限を超えて顧客が製品またはサービスを使用しないように、および同様に顧客に権限が与えられた範囲まで製品やサービスを顧客が使用することが防げられないようにできる限り早くイベントを処理することは必須である。

【０００８】

一部のイベントは、オフラインで処理され得る（月次請求など）。これらのイベントは、リアルタイムで処理される必要はない。しかしながら、電気通信の提供者は、適時に請求を作成する必要がある。たとえば、提供者は、毎月の初日に何億もの請求書を生成しなければならなくなり得る。高速の査定エンジンは、より良好なユーザエクスペリエンスを提供し（請求書が適時に作成される）、提供者は費用のかかるサーバ設備に投資しなくてもよくなる（なぜなら、全てのイベントを処理するために必要なサーバを少ないためである）。リアルタイムでのオンライン処理において、適時に（たとえば、数ミリ秒のうちに）イベントを処理することができなければ、タイムアウト、エラー、および故障などが起こり得る。システムの完全な崩壊を回避するために、提供者は、システムが過負荷となった時に低下モードでシステムを稼働させる。このモードにおいて、および一部の構成された方針に基づき、システムは消極的なサービスの否認（機会損失）または楽観的なサービスの許可（収入漏れ）を作り出し得る。

【発明の概要】

【発明が解決しようとする課題】

【０００９】

しかしながら、イベントを処理する現在の手法は不完全であることが分かっている。１つの手法において、査定エンジンは、一連の複数の（たとえば、プラグイン）モジュールを含む。顧客の各利用イベントに関し、モジュールの各々は、イベントおよび顧客の料金プランに基づいて特定の判定を行い、一連のモジュールにおける次のモジュールに結果を提供するように構成される。たとえば、電話による通話を査定する際、あるモジュールは、現在の日付が顧客の誕生日であるかをチェックするように構成され得て、他のモジュールは、通話の宛先が顧客の「友達および家族」グループ内であるかどうかを判定するように構成され得て、その場合に通話に対していくらと査定するかを判定し、他のモジュールは、一日のうちの時刻を判定し、誕生日である場合や宛先が特定の友達または家族である場合などの例外がないかのように通話に対して課金するように構成される。しかしながら

、このような手法を使用することには、いくつかの欠点がある。たとえば、一連のモジュールにおいてモジュールが終了して他のモジュールが実行される毎にコンテキスト切り替えを行わなければならない。他の例として、一連のモジュールにおける各モジュールは、一部のモジュールが不必要であったとしても実行される。したがって、後続のイベントを処理するために使用され得る計算リソース（たとえば、ＣＰＵサイクルおよびメモリ）を採用して現在のイベントを処理しなければならない。短時間の内に何十万ものイベントが作成される場合、このように「無駄となった」計算リソースは非常に貴重なものとなる。

【００１０】

他の手法において、査定エンジンは、イベントを評価する規則の大きなセットを含む。すなわち、査定エンジンで受け取られる各イベントについて、査定エンジンは各規則に対してイベントを評価する。しかしながら、多くのイベントについては、各規則に対する各イベントの評価が不要となり得る。たとえば、単一の規則に対するイベントの評価により、顧客に対して課金しない決定がなされた場合、セットにおける他の規則の各々に対するイベントの評価は不要である。再度、この手法において、計算リソースが無駄となる。

【００１１】

他の手法において、一連のモジュールまたは規則の大きなセットを用いる代わりに、単一のソフトウェアモジュールに料金プランを反映させる点において料金プランが「ハードコード」される。しかしながら、サービス提供者は、変化する市場の状況に対応するために、異なる料金プランを提供する柔軟性を望んでいる。したがって、ハードコードによる解決法は短期間においてのみ適切なものとなり得て、その後は陳腐化し得る。新しい料金プランの各々にハードコードによる解決法を付与することは、高価かつ冗長なものとなる。さらに、製品およびサービスの提供者の大半は、料金プランの作成および試作を抑制することを望んでいる。しかしながら、各料金プランについてソースコードを書くことをこのような提供者に要求することは望ましくない。

【００１２】

このセクションにおいて記載された手法は、追求され得た手法ではあるが、必ずしも以前に着想または追求された手法であるとは限らない。このため、別途示されない限り、このセクションに記載された手法はいずれも、単にこのセクションに含まれていることで先行技術としての地位を有するものであると仮定すべきでない。

【図面の簡単な説明】

【００１３】

【図１】発明の実施形態に係る、例示的な料金プランファンクタを図表を用いて表す査定図表を示すブロック図である。

【図２】発明の実施形態に係る、イベントを処理するための工程を示すフロー図である。

【図３】発明の実施形態に係る、複数の関数オブジェクトを含む料金プラン関数オブジェクトの例示的なメモリ表現を示すブロック図である。

【図４】発明の実施形態が実装され得るコンピュータシステムを示すブロック図である。

【発明を実施するための形態】

【００１４】

詳細な説明

以下の記載においては、説明を目的として、本発明の完全な理解を提供するために多くの具体的な詳細が述べられる。しかしながら、本発明はこれらの具体的な詳細がなくとも実施され得ることは明白である。他の場合においては、本発明が不用意に不明瞭とならないように、公知の構造および装置がブロック図の形式で示される。

【００１５】

概要

顧客によるサービスまたは製品の利用の量を示すイベントを査定するための技術が提供される。料金プランは、複数の関数オブジェクトが含まれる料金図表「ファンクタ(functor)」（または関数オブジェクト）によって表される。料金プランファンクタにおいて表されるロジックは、判断（たとえば、バイナリ）ツリーとして表わされ得る。イベントに

ついての情報は、関数オブジェクト（たとえば、「ルート(root)」ノード）の１つへの入力であり、これによって結果が生成される。結果は、複数のオブジェクトのうちのいずれが次に実行されるかを指示し得る。最終的な結果を生成するために実行される関数オブジェクトのセットは、料金プランファンクタにおける全ての関数オブジェクトよりも少なくなり得る。実施形態は、料金プランを記述するための関数的な手法を提示し、ＤＳＬ表現によって、料金プランを構成するデータおよび関数（条件および結果など）の両方を表わすことが容易となる。

【００１６】

ここで提供される例は、ユーザによるサービスまたは製品の利用の量に応じて生成される利用イベントを受け取って処理するものをいうが、発明の実施形態はこれに限定されない。たとえば、イベントは、月次請求サイクルイベントなどのシステムイベントであってもよい。このようなシステムイベントの受け取りに応答し、ここに開示されるイベント処理システムは、システムイベントを処理し、たとえば、月次料金に税金および諸費用を加えた課金を判定して返答し得る。

【００１７】

ドメイン固有言語

ＤＳＬは、特定の問題領域に対する専用のプログラミング言語または仕様言語である。ＤＳＬの作成（これを支援するソフトウェアを用いて）は、その言語が先に存在する言語よりも明瞭に特定のタイプの問題または解決法を表現できる場合に有益である。ＤＳＬを作成する目的の一部としては、ＤＳＬが表現的であり、簡潔であり、拡張可能であり、テストが容易であり、査定イベントを評価するのに特に最適化されていることが含まれ得る。

【００１８】

対照的に、料金プランを表現するための現在の技術には、ユーザが異なるフィールドに値を入力し得るユーザインターフェイスを含むシステムが伴う。フィールドおよびフィールドの値は、料金プランの規則を反映する。ユーザインターフェイスは、入力を受け付け、料金プランを反映する（比較的）大きなＸＭＬ文書（または他のテーブル表現）を生成する。このようなユーザインターフェイスは、ユーザには必要である。なぜなら、包括的なデータ表現（たとえば、ＸＭＬまたはテーブル表現）は料金プランの記述に適していないためである。代わりに、このようなデータ表現は、冗長となり、表現性を欠く傾向がある。ランタイム時において、査定エンジンの１つ以上のモジュールは、たとえばＸＭＬ文書にアクセスし、そこに含まれるＸＭＬデータを解析し、ＸＭＬデータに反映される規則に照らしてイベントを査定する。

【００１９】

プランを査定するためのＤＳＬを用いると、ＤＳＬはとても表現的であることから、翻訳層としてのＵＩは必要ない。さらに、単一のＤＳＬ表現は、料金プランのデータおよび規則の両方を記述することができる。これは、データ中心表現では不可能ではないが難しい。加えて、ＤＳＬ表現のためのランタイムモデルがデータおよび関数ロジックの両方を含み得る一方、データ中心表現のためのランタイムモデルはデータのみを含む場合が多い。他方、データ中心ランタイムモデルのランタイム処理は、通常は遅く、最適化されていない。なぜなら、このようなモデルは、関数ロジックを稼働させるために付加的なシステムを必要とするためである。付加的なシステムは、データ中心表現において表される全ての料金プランについて通常は同じであり、このため最適化されていない。

【００２０】

実施形態によれば、ドメイン固有言語（domain specific language：ＤＳＬ）が作成され、料金プランを表現するために使用される。ＤＳＬのボキャブラリは、コアとなる査定のコンセプトを取り込んだものに限定され得る。ＤＳＬは、プログラマおよび特定分野の専門家の両方が迅速に学習できるほど十分な表現性を有するべきである。ＤＳＬ宣言文の例を以下に示す。

【００２１】

(dayOfYear == @birthday) => linearRate(0.00) |+
(@calledld <: @friendsAndFamily) => linearRate(0.01) |+
[20:00:00,07:00:00] => linearRate(0.02) |+
linearRate(0.05)

このＤＳＬ宣言文において使用されるボキャブラリは、査定ドメインに固有のものである。たとえば、「dayofYear」は、現在の年の現在の日を返答するシングルトンである。「@birthday」は、現在処理されているイベントに関連付けられた顧客の誕生日を判定する複雑なＤＳＬ表現のエイリエアスであり、「linearRate」は、入力パラメータ値（たとえば、「0.00」）を受け付け、出力として値を生成する関数であり、ここで値は顧客への課金額を表わす。「=>」の記号は、前の表現式が真である場合に直ちに後続の表現式が評価されることを示す。記号「|+」は、前の条件（たとえば、「(dayOfYear = @birthday)」）が偽である場合にイベントがＤＳＬ宣言文における後続の表現式を用いて処理されることを示す。また、記号「|+」は、前の条件が部分的に真であり得ること、およびイベントが後続の表現式によって処理されることを示す。たとえば、顧客が顧客の誕生日の午後１１：５５に友達に電話し、通話が１０分間続く。上記の例示的なＤＳＬ宣言文において反映される料金プランによれば、通話の最初の５分は無料であり、最後の５分は最初の記号「|+」の後の表現式によって査定される。

【００２２】

料金プランを表現するための専用のＤＳＬを作成することによって、いくつかの利点が実現される。１つは、ＤＳＬ宣言文の設計者または作成者は、イベント処理システムの内部から遮蔽されることである。他の利点は、言語が（ａ）料金プランを反映するＸＭＬ文書を較正するよりも表現的であり、（ｂ）料金プランをハードコードで適用するためのソースコード（たとえば、Java（登録商標））を書くよりも一層表現的なことである。他の利点は、ＤＳＬ宣言文のテストがコードのテストよりも一層容易なことである。

【００２３】

ＤＳＬ宣言文は、料金プランの設計者などのユーザによって書かれ得る。加えて、または代替的に、ＤＳＬ宣言文は、料金プランを反映する入力を受け付けるように設計されたユーザインターフェイスを介して受け取ったユーザ入力に基づき、自動的に生成され得る。入力は、直接的にＤＳＬ宣言文へ変換され得る、またはＸＭＬなどの中間フォーマットに変換されて中間フォーマットからＤＳＬ宣言文に変換され得る。

【００２４】

料金プランを表現するために作成されるＤＳＬの例示的なレキシコンを以下に示す。

オペレータ：+, -, <, >, ==, |, &&, 3, <:, |+, ||, !, =>

シングルトン：start, end, dayOfYear, dayOfWeek, dayOfMonth, quantity, TRUE, FALSE

他の関数：linearRate, fixedRate, fail, localDate, date, getObject, getBigDecimal, getBoolean

エイリアス：@birthday, @dateOfBirth, @qos, @friendsAndFamily, @balance, @inputValue, @outputVolume, @cellHomeIds, @calledId, @cellId

オペレータは、１つ以上のオペランドを入力として受け付け、オペランド自体は関数であり得る。シングルトンは、イベントからのデータを出力の生成に必要としない関数である。「他の関数」は、出力を生成するためにイベントからのデータまたはイベントについてのデータを入力として必要とする関数である。

【００２５】

実施形態において、料金プランを表現するために作成されるＤＳＬは拡張可能である。すなわち、新しいボキャブラリをサポートするようにＤＳＬのためのＤＳＬパーサーが拡張される限り、追加の関数がサポートされ得る。同様に、ＤＳＬはエイリアスの作成において拡張可能であり得る。上記の例示的なＤＳＬ宣言文は、より複雑なＤＳＬ表現に分解されるエイリアスを含む。たとえば、@birthdayは「getObject(RatingContext#customer/birthday/toString?MMdd")」のエイリアスであり得て、ここでgetObjectは上記の「他の

関数」の例である。関数がより複雑になるにつれ、ＤＳＬにエイリアスが加えられ、ＤＳＬ宣言文の構成が簡潔となり得る。

【００２６】

料金プランファンクタ

実施形態によれば、ＤＳＬパーサーは、ＤＳＬ宣言文を入力として受け付け、宣言文をパースし、料金プランファンクタを生成する。料金プランファンクタは、ＤＳＬ宣言文によって規定された料金プランの関数ロジックを概念的に表わす判断ツリーとして示され得る。したがって、ＤＳＬ宣言文は、その言語において料金プランファンクタを「モデル化」する。実際に、判断ツリーの各論表現（表現式がブール値を返すか、またはイベントを査定するか）は、対応するＤＳＬ宣言文において表現式に対して直接的に対応し得る。ＤＳＬシンタックスは料金プランファクタの表示態様に非常に類似していることから、ＤＳＬの使用は設計者にとって容易なはずである。

【００２７】

判断ツリーとして、料金プランファンクタは順序付けされた関数の集合として示され得る。各関数は、関数オブジェクトなどのオブジェクトとして実装され得る。「ファンクタ」ともいわれる関数オブジェクトは、あたかもオブジェクトが通常の関数であるかのように、通常は同じシンタックスを伴ってオブジェクトを呼び出す、またはコールするコンピュータプログラミング構成である。上記の例示的なレキシコン（すなわち、オペレータ、シングルトン、他の関数、およびエイリアス）において示される各関数は、ファンクタとして実装され得る。エイリアスについては、パース時において、ＤＳＬパーサーはエイリアスに基づいてエイリアスについての１つ以上のファンクタおよび入力を識別し、１つ以上のファンクタを生成する。

【００２８】

実施形態において、料金プランファンクタはバイナリツリーであって、このバイナリツリーにおいて各ノードは少なくとも２つの子（ここでは「右側子」および「左側子」という）を有する。親ノードから右側子への接続は、「=>」のオペレータファンクタによってモデル化され得て、親ノードから左側子への接続は、「|+」のオペレータファンクタによってモデル化され得る。ＤＳＬパーサーは、ＤＳＬ宣言文の一部である全てのファンクタの組み合わせであるファンクタを生成する。

【００２９】

料金プランファンクタの非リーフノードは、真または偽（完全または部分的）を評価する述語もしくは条件ファンクタとして考慮される。述語ファンクタによって表される条件は、任意の条件であり得る。このような条件は、通話先、通話がいつ行われたか、および通話がどのくらいの長さにわたるかに基づき得る。たとえば、条件は査定する量の全てまたは一部についてであり得て、この量は「([20:00:00,07:00:00])」などの期間であり得る。非リーフノードは、２つの子ノードのそれぞれに対する保護として作用し得る。述語ノードの右側子は、述語が真である量を入力として受け付け得る。述語ノードの左側子は、述語が偽である量を入力として受け付け得る。述語ノードの各子ノードは、リーフノードまたは他の非リーフ（もしくはブランチ）ノードであり、各々は料金ファンクタとして考慮される。

【００３０】

リーフノードは、入力としてリーフノードに与えられる量を査定し、通常は量に対する料金（これに加えて、たとえば、総勘定元帳割当、税法など、料金に関する一部の他の情報）を含む結果を返す。たとえば、料金ファンクタは「linearRate(0.01)」について生成され得る。ブランチノードは、 料金ファンクタとして考慮され得るが、料金ファンクタとして共に作用する複数のファンクタの組み合わせである。たとえば、「[20:00:00,07:00:00] => linearRate(0.02) |+ linearRate(0.05)」が料金ファンクタとして考慮され得る。さらに、料金プランファンクタの全体は、料金ファンクタでもあるブランチ（またはツリー）として考慮され得る。料金プランファンクタを実行した結果は、実行されたリーフノードの各々から得られる全ての課金の集合であり（単一の料金ファンクタの結果のリ

スト）、これは料金プランファンクタにおける全てのリーフノードのサブセットであり得る。

【００３１】

実施形態において、ノード間の接続もファンクタであり、これはここではオペレータファンクタという。オペレータファンクタの限定されない例としては、「+」、「-」、「*」、「=>」、および「==」が含まれる。たとえば、「=>」は、左側オペランド（たとえば、述語ファンクタ）および右側右辺オペランド（たとえば、料金ファンクタ）を取るバイナリファンクタである。すなわち、述語ファンクタ（または左側オペランド）が真である場合、料金ファンクタ（または右側オペランド）が実行される。料金ファンクタの評価は、左側オペランド（または述語ファンクタ）が真である場合、量（たとえば、期間）に対する課金を返す。

【００３２】

他の例として、「|+」は、アグリゲータとして作用し、左側（または単に「左」）料金ファンクタおよび右側（または単に「右」）料金ファンクタの２つの料金ファンクタを取るバイナリファンクタである。このバイナリファンクタは、左料金ファンクタによって作られる課金を評価し、量が完全に査定されない場合にはこのバイナリファンクタは右料金ファンクタによって作られる課金を加える。以上に基づき、このような関数的手法により、ファンクタを帰納的に結合させてより複雑なファンクタを作ることができる。表現性のために、ＤＳＬパーサーは、読み取り不能な機納的関数呼出ではなく、より方程式に近い構成を受け付けるように設計される。

【００３３】

実施形態において、ファンクタは不変オブジェクトであり、すなわち作成された後は状態が変更されない。一部の場合において、内部で使用される一部の属性が変化しているが外部からの視点でオブジェクトの状態が変化していないように見える場合であっても、オブジェクトは不変と見なされる。たとえば、高価な計算の結果をキャッシュするためにメモ化を用いるオブジェクトは、依然として不変オブジェクトとして考慮される。したがって、特定のファンクタが所定の入力を伴って呼び出される場合、その入力を用いたファンクタの実行結果はファンクタに関連付けて（またはファンクタ内に）キャッシュまたは格納され得る。したがって、次回にファンクタがその入力を伴って呼び出された場合（たとえば、異なる顧客によって開始され得る他のイベントという関係において）、結果はファンクタのロジックを実行することなく読み出される。この結果も入力として他のファンクタに渡され得る。不変オブジェクトの他の利点は、不変オブジェクトが容易に共有および構成されること、ならびにスレッドセーフであることにある。

【００３４】

したがって、料金プランファンクタは、イベントを査定するための全てのロジックおよびデータ（イベント自体から要求されるデータ以外）を含む関数オブジェクトである。全てのロジックおよびデータを含むというこの特徴は、従来のランタイム手法、すなわち関数ランタイムモデルが料金プランを実行可能オブジェクトとする場合と比した他の利点である。一般的な手法は、料金プランのデータ表現を考慮し、料金プランを評価するよう別個の査定エンジンに要求するのみである。このようなエンジンは、単に料金プランを明細として使用する。ここに記載される実施形態においては、料金ファンクタ自体がエンジンである。

【００３５】

図１は、発明の実施形態に係る、料金プランファンクタを図表で表す例示的な査定図表１００を示すブロック図である。リーフノードは、所与のイベントについて計算され得る異なる可能な課金を表わす。査定図表１００における各ノードは、ファンクタに対応する。具体的に、ファンクタ１１０は、「今日は顧客の誕生日ですか？」という質問に回答する。質問への回答が「はい」である場合、ファンクタ１２０が実行され、これによって顧客は確実に利用について課金されない。質問への回答が「いいえ」である場合、ファンクタ１３０が実行される。

【００３６】
ファンクタ１３０は、「顧客は友達または家族に電話をかけていますか？」という質問に回答する。質問への回答が「はい」である場合、ファンクタ１４０が実行され、１分あたり＄０．０１ドルの課金が計算される。したがって、認識された友達または家族に顧客が電話をかけ、その通話が１０分間続いた場合、ファンクタ１４０は、顧客のアカウントに＄０．１０の課金を確実に反映させる。質問に対する回答が「いいえ」である場合、ファンクタ１５０が実行される。
【００３７】
ファンクタ１５０は、「ピーク時間中に通話を行っていますか？」という質問に回答する。質問に対する回答が「はい」である場合（通話期間の少なくとも一部について）、ファンクタ１６０が実行され、１分あたり＄０．０５の課金を計算する（ピーク時間中の通話期間の少なくとも一部について）。質問に対する回答が「いいえ」である場合（通話期間の少なくとも一部について）、ファンクタ１７０が実行され、１分あたり＄０．０２の課金が計算される（ピーク時間中でない通話期間の少なくとも一部について）。いずれの結果においても、それぞれの関数により、適切な課金が確実に顧客のアカウントに反映される。
【００３８】
イベントの処理
図２は、発明の実施形態に係る、イベントを処理するための処理２００を示すフロー図である。処理２００は、イベント処理エンジンによって行われる。処理２００は単一の処理によって行われ得る、または処理２００の異なるステップは異なる処理によって、または同じ処理の異なるスレッドによって行われ得る。
【００３９】
ステップ２１０において、顧客による利用を示すイベントが受け取られる。
ステップ２２０において、料金プランを表わす料金プランファンクタが識別される。複数の料金プランがある場合、複数の料金プランから適切な料金プランファンクタがまず選択される。
【００４０】
ステップ２３０において、料金プランファンクタによって示される第１のファンクタが実行され、第１の結果が生成される。第１のファンクタは、料金プランファンクタにおけるルートノードを表わし得る。
【００４１】
ステップ２４０において、第１の結果は、次に実行するファンクタを判定するために使用される。結果が評価された後、次に呼び出すファンクタは２つ以上あり得る（料金プランに依る）。たとえば、査定図表１００において実行される関数１１０の結果がブールの真である場合、ファンクタ１２０（入力を要求しない）が実行される。実行される関数１１０の結果がブールの偽である場合、ファンクタ１３０（イベントを開始した顧客についての入力が要求される）が実行される。
【００４２】
ステップ２５０において、料金プランファンクタによって示される第２のファンクタが実行され、第２の結果が生成される。第２の結果は、ブール値、またはたとえば整数値であり得る。たとえば、第２のファンクタがファンクタ１３０に対応する場合、第２の結果はブール値である。第２のファンクタがファンクタ１２０に対応する場合、第２の結果は整数値である。
【００４３】
ステップ２６０において、顧客に対する課金額が第２の結果に基づいて判定される。例示的な査定図表１００において、ファンクタ１６０またはファンクタ１７０を実行した結果は、ファンクタ１３０を実行した結果がブール値であっても、ファンクタ１３０を実行した結果に「基づいている」として考慮される。ファンクタ１６０またはファンクタ１７０の実行は、ファンクタ１３０を実行した結果が偽を示すという事実に依存する。

【００４４】
この例はファンクタ評価のみについて言及したが、実施形態は、単一のイベントに応じてより多くのファンクタ評価を含み得る。
【００４５】
イベントを査定する（利用イベントまたはシステムイベントであるか）ために料金プランファンクタ（すなわち、ファンクタの集合からなる）を使用する利点は、構成可能性および低下した複雑性を含む点にある。低下した複雑性（利用イベントを処理するための現在の手法と比して）は、オブジェクトの「フラットな」クラス階層によって実現される。すなわち、料金プランファンクタにおける全てのオブジェクトはファンクタであり得る。異なるファンクタ間における継承はない。
【００４６】
構成可能性に関して、ファンクタは他のファンクタと結合され、特定の料金プランのより複雑な規則を表わすブランチまたはツリーが形成され得る。したがって、複雑な料金プランファンクタは、複数の「単純な」ファンクタから構成され得る（または作られ得る）。オブジェクト指向プログラミングの特徴であって複雑さを増す承継よりも、構成可能性の方が好ましい。
【００４７】
料金プランファンクタは、可変性（すなわち、オペランドの順序が変わっても最終結果は変わらない）、結合性（すなわち、動作が行われる順序を変更しても最終結果は変わらない）、および分配性などの数学的な特性を活用する。たとえば、以下のＤＳＬ宣言文は、ＤＳＬ宣言文をパースすることによって得られる料金図表ファンクタの分配特性を例示している。
【００４８】
```
(dayOfYear==@birthday) =>
(([07:00:00, 20:00:00] => linearRate(0.05))
|+ ([20:00:00, 07:00:00] => linearRate(0.01)))
```
これは、以下の宣言文と等価である（すなわち、同じ結果をもたらす）。
【００４９】
```
(dayOfYear==@birthday) => (([07:00:00, 20:00:00] => linearRate(0.05))
|+ (dayOfYear==@birthday) =>([20:00:00, 07:00:00] => linearRate(0.01))
```
複数の料金プラン
（利用）イベントを処理する製品またはサービスの提供者は、将来の顧客に対し、いずれかが選択される複数のプランを提示し得る。このため、実施形態において、複数の料金プランファンクタが各料金プランについて生成される。上に示されるように、料金プランファンクタの１つのソースは、料金プランの設計者によって構成される複数のＤＳＬ宣言文を含み得る。加えて、または代わりに、料金プランファンクタの他のソースは、料金プランを反映する規則（すなわち、ＤＳＬ宣言文の形式ではない）をユーザが入力するユーザインターフェイスであり得る。入力されたデータは、次に料金プランを反映するＤＳＬ宣言文に変換される。
【００５０】
このため、実施形態において、イベントがイベント処理エンジンに到達すると、イベント処理エンジンは複数の料金プランファンクタから料金プランファンクタを識別し、各料金プランファンクタは異なる料金プランに対応する。イベントは、（イベントを開始した）顧客が前に選択した料金プランを示す料金プランデータを含み得る。代替的に、イベントは、特定の料金プランと関連付けられ得る（たとえば顧客・料金プランテーブル）顧客識別子を単に示し得る。したがって、イベント処理エンジンは、顧客と関連付けられた料金プランを識別し、料金プランと関連付けられた料金プランファンクタを識別し、料金プランと関連付けられた料金プランファンクタにイベントを渡し得る。料金プランファンクタは、不揮発性または持続性メモリに格納され、揮発性メモリにロードされ得る。代替的に、料金プランファンクタは、既に揮発性メモリにロードされ得る。

【００５１】
関連する実施形態において、複数の料金プランが単一のイベントに適用され得る。したがって、単一のイベントは、処理のために２つ以上の料金プランファンクタに渡され得る。たとえば、１つの料金プランは課金を計算するためのものであり、他の料金プランは計算された課金に対して適用される値引きを計算するためのものであり得る。
【００５２】
ファンクタの「共有(sharing)」
複数の料金プランは、多くの属性を共通で有し得る。たとえば、複数の料金プランは、たとえば顧客によって開始されたイベントの日が顧客の誕生日である場合に顧客は対応する利用に関して課金されない「誕生日規則」を有し得る。各料金プランファンクタが「誕生日」ファンクタのインスタンスを有する場合、誕生日ファンクタのインスタンスが複数となり得る。しかしながら、誕生日規則を含む各料金プランに対して「誕生日ファンクタ」の異なるインスタンスを生成する代わりに、誕生日ファンクタの単一のインスタンスがメモリに格納され、複数の料金プランファンクタによって「共有」される。このような手法により、システムにおけるファンクタインスタンスの数が減少し、イベント処理エンジンのメモリフットプリントが低下する。シングルトン関数（dayOfYearファンクタなど）に加え、非シングルトンファンクタも共有され得る。たとえば、開始と終了を取るTimeRangeファンクタは、査定システムにわたって共有されるファンクタの複数のインスタンスを有し得る。通常、多くの料金プランは、同じピーク期間およびオフピーク期間を共有することから、TimeRangeファンクタ（[19:00:00,07:00:00]によって表されるオフピークおよび [07:00:00,19:00:00]によって表されるピークなど）の同じインスタンスを共有し得る。
【００５３】
ランタイムの実行
実施形態において、イベント処理エンジンが受け取る各イベントについて、イベントを扱う処理またはスレッドは、イベントに対応する料金プランを判定し、料金プランに対応する料金プランファンクタを識別し、料金プランファンクタを実行させる。複数の料金プランがイベントに対応すると判定された場合、複数の料金プランファンクタが識別され、各料金プランファンクタが実行される。これらのステップを行い得る処理またはスレッドは、ここでは「イベント処理スレッド」といい、これはシングルスレッド処理またはマルチスレッド処理であり得る。ファンクタが不変である実施形態において、マルチスレッドで料金プランを評価することは完全にスレッドセーフである。
【００５４】
料金プランを表わすバイナリツリーモデルはルートファンクタというが、ＤＳＬ表現をパースした結果は、固有の料金プランファンクタオブジェクトである。ひとたびこのファンクタが生成されると、料金プランの評価はＦ（Ｘ）＝Ｙの呼出ほど単純である。ここで、Ｆは料金プランファンクタであり、Ｘはイベントであり、Ｙは結果 （これは適用可能な課金のリストであり得る）である。料金プランファンクタ「Ｆ」を構成するファンクタの呼出の順序は、料金プランファンクタの実装に組み込まれる。ひとたび料金プランファンクタが識別されると、イベント処理プログラムは料金プランファンクタを「コール」する。
【００５５】
実施形態において、ファンクタは状態を維持しないことから、複数のスレッド（たとえば、異なるイベントの処理に伴う）が特定のファンクタの単一のインスタンスを実行する場合には、データ破壊または不正確な結果の危険がない。
【００５６】
ある意味で、イベント処理スレッドは「ランダムに」ファンクタの「方程式」を評価する。次にどのファンクタを実行するかを判定する決定は、料金プランファンクタの一部であるオペレータファンクタに委任される。したがって、外部のシステムは、料金プランをどのように評価するかについて決定する必要はない。関数ロジックは、料金プランファン

クタ自体に組み込まれる。すなわち、料金プランファンクタ自体が実行可能なコードを含むのに対し、以前の査定エンジンの実装は、料金プランを表わすデータを読み取ってそのデータに基づいて動作を行うために外部のシステムに依存する。したがって、関数料金プランモデルは、以前の査定エンジンの実装に対して重要な利点を有する。

【００５７】

料金プランファンクタのメモリ表現

図３は、発明の実施形態に係る、複数のファンクタを含む料金プランファンクタの例示的なメモリ表現３００を示すブロック図である。表現３００は、例示的な料金プランに対応する料金プランファンクタにおける異なるタイプの動作を表わす多数のファンクタを含む。具体的に、表現３００は、４つのファンクタに直接的に結合されるアグリゲータオペレータ３０２（「|+」）を含み、４つのファンクタは、３つのif-thenオペレータ３０４Ａ～Ｃ（「=>」）および入力が0.05である料金ファンクタ３０６Ａ（「linearRate」）である。if-thenオペレータ３０４Ａは、等号オペレータ３０８Ａ（「==」）および入力が0.00である料金ファンクタ３０６Ｂに結合される。等号オペレータ３０８Ａが真である場合、if-thenオペレータ３０４Ａにより、料金ファンクタ３０６Ｂが評価される。等号オペレータ３０８Ａのオペランドは、シングルトンオペレータ３１０（「DayofYear」）およびエイリアスオペレータ３１２Ａ（「@Birthday」）である。等号オペレータ３０８Ａは、この料金プランファンクタによって処理されるイベントに関連付けられた人の誕生日がイベントの発生した日と同じ日である場合に真と評価する。そうでない場合には、等号オペレータ３０８Ａは偽と評価する。

【００５８】

査定される追加の時間または利用量がある場合（この例においては、等号オペレータ３０８Ａが偽と判定する場合）、if-thenオペレータ３０４Ｂが評価される。if-thenオペレータ３０４Ｂは、２つのオペランドを有する「within」オペレータ３０８Ｂ（「<:」）に結合され、２つのオペランドは、エイリアスオペレータ３１２Ｂ（「@callerID」）およびエイリアスオペレータ３１２Ｃ（「@friendsAndFamily」）である。この料金プランファンクタの「ブランチ」は、通話先が呼出元の友達および家族のセット内である場合にif-thenオペレータ３０４Ｂの「then」部分が評価されることを示す。この「then」部分は、入力が0.01である料金ファンクタ３０６Ｃ（「linearRate」）である。

【００５９】

査定される追加の時間または利用量がある場合（この例においては、等号オペレータ３０８Ｂが偽と判定する場合）、if-thenオペレータ３０４Ｃが評価される。if-thenオペレータ３０４Ｃは、入力が20:00:00および07:00:00である時間範囲ファンクタ３１４（「[]」）ならびに入力が0.02である料金ファンクタ３０６Ｄ（「linearRate」）に結合される。この料金プランファンクタの「ブランチ」は、サービス利用（たとえば、電話）の少なくとも一部が午後８時と午前７時の間である場合にその部分が料金ファンクタ３０６Ｄによって査定され、料金ファンクタ３０６Ｄは１分あたり２セントと査定する。

【００６０】

査定される追加の時間または利用量がある場合（全利用期間がif-thenオペレータ３０４Ｃによって評価されなかった場合であり得る）、査定ファンクタ３０６Ａが評価される。料金ファンクタ３０６Ａ（「linearRate」）は入力として0.05を取る。この料金プランファンクタの「ブランチ」は、（１）電話の日付が呼出元の誕生日でない場合、（２）通話先が呼出元の友達および家族のリストに載っていない場合、および（３）通話の少なくとも一部が午前７時より後および午後８時より前に発生した場合に評価される。

【００６１】

ハードウェアの概要

１つの実施形態によれば、ここに記載される技術は、１つ以上の特定用途向け演算装置によって実装される。特定用途向け演算装置は、ハードウェアによって技術を実装し得る、または技術を実装するように永続的にプログラミングされた１つ以上の特定用途向け集積回路（ＡＳＩＣ）もしくはフィールドプログラマブルゲートアレイ（ＦＰＧＡ）などの

デジタル電子装置を含み得る、またはファームウェア、メモリ、他の記憶装置、または組み合わせにおけるプログラム命令に従って技術を実装するようにプログラムされた１つ以上の汎用ハードウェアプロセッサを含み得る。このような特定用途向け演算装置は、技術を達成するために、カスタムプログラミングにカスタムハードワイヤードロジック、ＡＳＩＣ、またはＦＰＧＡを結合させ得る。特定用途向け演算装置は、技術を実装するために、デスクトップコンピュータシステム、ポータブルコンピュータシステム、携帯用デバイス、ネットワークデバイス、またはハードワイヤードロジックおよび／もしくはプログラムロジックを組み込む任意の他の装置であり得る。

【００６２】

たとえば、図４は、発明の実施形態が実装され得るコンピュータシステム４００を示すブロック図である。コンピュータシステム４００は、バス４０２もしくは情報を通信するための他の通信機構と、情報を処理するためにバス４０２に結合されたハードウェアプロセッサ４０４とを含む。ハードウェアプロセッサ４０４は、たとえば、汎用マイクロプロセッサであり得る。

【００６３】

コンピュータシステム４００は、情報およびプロセッサ４０４によって実行される命令を格納するためにバス４０２に結合される、ランダムアクセスメモリ（ＲＡＭ）もしくは他の動的記憶装置などのメインメモリ４０６も含む。メインメモリ４０６は、プロセッサ４０４によって実行される命令の実行中に一時変数または他の中間情報を格納するためにも使用され得る。このような命令は、プロセッサ４０４にアクセス可能な非一時的な記録媒体に格納されると、命令において指定される動作を行うようにカスタマイズされた特定用途向けマシンにコンピュータシステム４００を変える。

【００６４】

コンピュータシステム４００は、プロセッサ４０４のための静的情報および命令を格納するためにバス４０２に結合される読み取り専用メモリ（ＲＯＭ）４０８または他の静的記憶装置をさらに含む。磁気ディスクもしくは光ディスクなどの記憶装置４１０が設けられ、情報および命令を格納するためにバス４０２に結合される。

【００６５】

コンピュータシステム４００は、コンピュータのユーザに対して情報を表示するための陰極線管（ＣＲＴ）などのディスプレイ４１２にバス４０２を介して結合され得る。英数字および他のキーを含む入力装置４１４は、プロセッサ４０４に対して情報およびコマンド選択を通信するためのバス４０２に結合される。他のタイプのユーザ入力装置は、方向情報およびコマンド選択をプロセッサ４０４に対して通信するための、およびディスプレイ４１２上でのカーソル移動を制御するための、マウス、トラックボール、またはカーソル方向キーなどのカーソル制御４１６である。この入力装置は、通常、第１の軸（たとえば、ｘ）および第２の軸（たとえば、ｙ）の２つの軸において２つの自由度を有し、これによって装置が面における位置を指定することができる。

【００６６】

コンピュータシステム４００は、カスタマイズされたハードワイヤードロジック、１つ以上のＡＳＩＣもしくはＦＰＧＡ、ファームウェアおよび／またはプログラムロジックをコンピュータシステムと組み合わせて使用してここに記載する技術を実装し、コンピュータシステム４００を専用マシンとする、または専用マシンへとプログラミングする。一実施形態によれば、ここでの技術は、メインメモリ４０６に含まれる１つ以上の命令の１つ以上のシーケンスをプロセッサ４０４が実行することに応答してコンピュータシステム４００によって行われる。このような命令は、記憶装置４１０などの他の記録媒体からメインメモリ４０６に読み込まれ得る。メインメモリ４０６に含まれる命令のシーケンスを実行することにより、ここに記載の処理ステップがプロセッサ４０４によって実行される。代替的な実施形態において、ハードワイヤード回路は、ソフトウェア命令の代わりに、またはソフトウェア命令と組み合わせて使用され得る。

【００６７】

ここで使用される「記録媒体」の用語は、特定の方法でマシンを動作させるデータおよび／または命令を格納する任意の非一時的な媒体をいう。このような記録媒体は、不揮発性の媒体および／または揮発性の媒体を含み得る。不揮発性の媒体は、たとえば、記憶装置４１０などの光ディスクまたは磁気ディスクを含む。揮発性の媒体は、メインメモリ４０６などの動的メモリを含む。記録媒体の一般的な形式は、たとえば、フロッピー(登録商標）ディスク、フレキシブルディスク、ハードディスク、ソリッドステートドライブ、磁気テープもしくは任意の他の磁気データ記録媒体、ＣＤ－ＲＯＭ、任意の他の光データ記録媒体、穴のパターンを伴う任意の物理媒体、ＲＡＭ、ＰＲＯＭ、およびＥＰＲＯＭ、ＦＬＡＳＨ－ＥＰＲＯＭ、ＮＶＲＡＭ、任意の他のメモリチップもしくはカートリッジを含む。

【００６８】

記録媒体は、伝送媒体とは異なるが、伝送媒体と併せて使用され得る。伝送媒体は、記録媒体間の情報の転送に関わる。たとえば、伝送媒体は、バス４０２を包含するワイヤを含む、同軸ケーブル、銅線、および光ファイバを含む。伝送媒体は、電波および赤外線データ通信時に生成されるような音波または光波の形式を取り得る。

【００６９】

様々な形式の媒体は、プロセッサ４０４が実行する１つ以上の命令の１つ以上のシーケンスの搬送に関わり得る。たとえば、命令は、リモートコンピュータの磁気ディスクまたはソリッドステートドライブ上に最初は担持される。リモートコンピュータは、そのダイナミックメモリに命令をロードし、モデムを用いて電話回線を介して命令を送り得る。コンピュータシステム４００内のモデムは、電話回線上のデータを受け取り、赤外線送信器を使用してデータを赤外線信号に変換し得る。赤外線検知器は、赤外線信号で運ばれたデータを受け取り得て、適切な回路がデータをバス４０２上に配置し得る。バス４０２はデータをメインメモリ４０６に運び、ここからプロセッサ４０４は命令を検索して実行する。メインメモリ４０６によって受け取られる命令は、プロセッサ４０４による実行の前または後に記憶装置４１０上に任意に格納され得る。

【００７０】

コンピュータシステム４００は、バス４０２に結合される通信インターフェイス４１８も含む。通信インターフェイス４１８は、ローカルネットワーク４２２に接続されるネットワークリンク４２０に結合される双方向データ通信を提供する。たとえば、通信インターフェイス４１８は、統合サービスデジタル網（ＩＳＤＮ）カード、ケーブルモデル、衛星モデム、またはモデムであり得て、対応するタイプの電話回線へのデータ通信接続を提供する。他の例として、通信インターフェイス４１８は、ローカルエリアネットワーク（ＬＡＮ）カードであり得て、対応のＬＡＮへのデータ通信接続を提供する。無線リンクも実装され得る。このような施行例において、通信インターフェイス４１８は、様々なタイプの情報を表わすデジタルデータストリームを運ぶ電気信号、電磁信号、または光信号を送信および受信する。

【００７１】

ネットワークリンク４２０は、１つ以上のネットワークを介して他のデータ装置へのデータ通信を通常は提供する。たとえば、ネットワークリンク４２０は、ローカルネットワーク４２２を介して、ホストコンピュータ４２４へ、またはインターネットサービスプロバイダ（ＩＳＰ）４２６によって運営されるデータ機器へ接続を提供し得る。ＩＳＰ４２６は、次に、現在では一般的に「インターネット」４２８といわれる全世界パケットデータ通信網を介してデータ通信サービスを提供する。ローカルネットワーク４２２およびインターネット４２８の両方は、デジタルデータストリームを運ぶ電気信号、電磁信号、または光信号を使用する。様々なネットワークを介した信号、およびネットワークリンク４２０上で通信インターフェイス４１８を介した信号は、コンピュータシステム４００に対してデジタルデータを運ぶ、伝送媒体の例示的な形式である。

【００７２】

コンピュータシステム４００は、ネットワーク、ネットワークリンク４２０、および通

信インターフェイス４１８を介して、メッセージの送信、およびプログラムコードを含むデータの受信を行い得る。インターネットの例において、サーバ４３０は、インターネット４２８、ＩＳＰ４２６、ローカルネットワーク４２２、および通信インターフェイス４１８を介してアプリケーションプログラムのための要求されたコードを送信し得る。

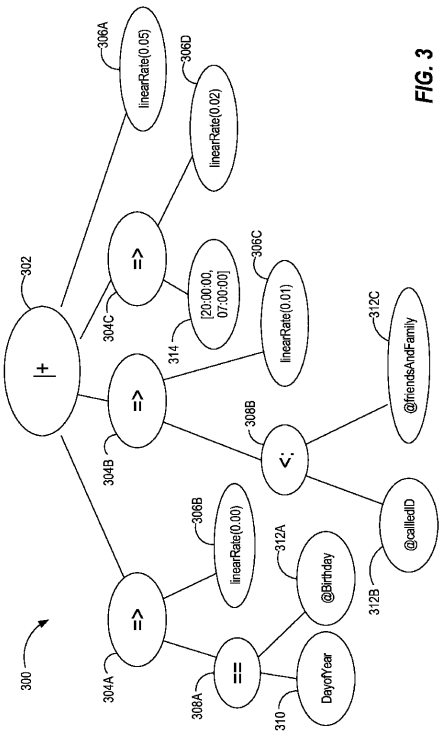
【００７３】

受信されたコードは、受信され次第プロセッサ４０４によって実行され得る、および／または記憶装置４１０に格納され得る、もしくは後で実行するために他の不揮発性記憶装置に格納され得る。

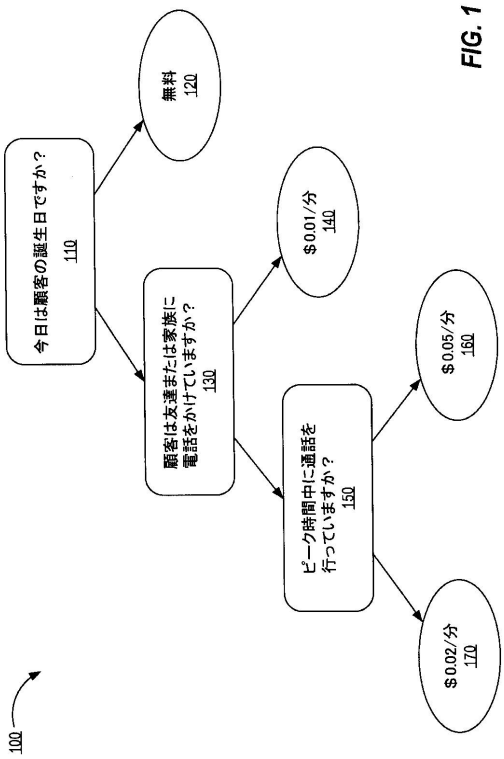
【００７４】

上記の明細書において、実装毎に異なり得る多数の具体的な詳細を参照して本発明の実施形態が記載された。このため、明細書および図面は、限定的ではなく例示的なものとして考慮される。発明の範囲についての唯一および排他的な指標、ならびに出願人が何を発明の範囲として意図しているかは、本件出願に由来する請求項のセットにおける文言上の範囲および均等物の範囲であり、このような請求項が由来する特定の形式で記載され、後の任意の修正が含まれる。

【図３】



【図１】



【図２】

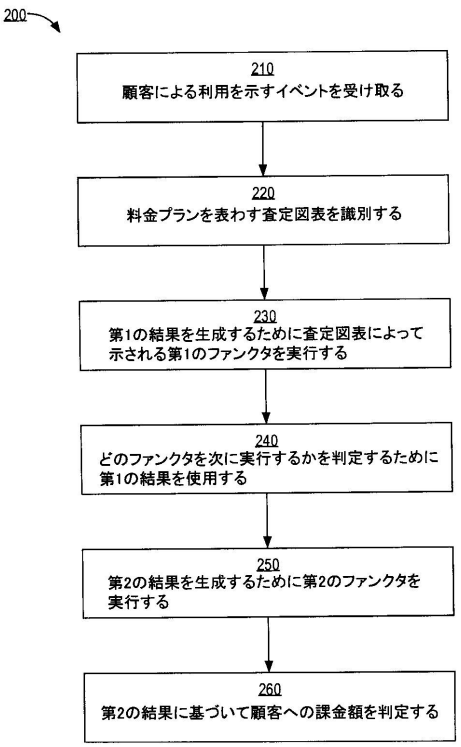


FIG. 2

【図４】

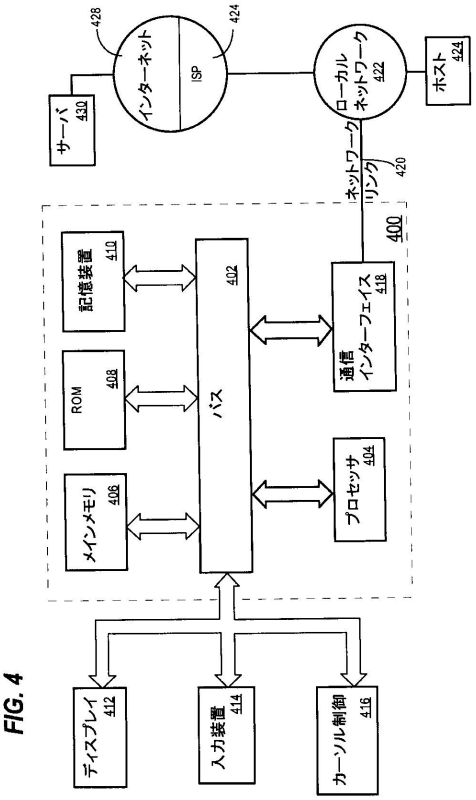


FIG. 4

フロントページの続き

(72)発明者　マラクサムドラ，ギレーシュ
アメリカ合衆国、９４０６７　カリフォルニア州、サニーベール、アルバトロス・ドライブ、１６３７
(72)発明者　ケメラー，イェンス
アメリカ合衆国、９４０４３　カリフォルニア州、マウンテン・ビュー、ティレラ・アベニュ、２５４
(72)発明者　ピロ，ルイス（ジュニア）
アメリカ合衆国、９５１２５　カリフォルニア州、サン・ノゼ、ウェストゲイト・アベニュ、２５８５

審査官　山内　裕史

(56)参考文献　特開２００１－１８８８４１（ＪＰ，Ａ）
特開２０１１－１５４６５２（ＪＰ，Ａ）
特開２００３－１３４１１１（ＪＰ，Ａ）
特開２００９－０９９１３６（ＪＰ，Ａ）
米国特許出願公開第２００８／００１４９０４（ＵＳ，Ａ１）
特表２００９－５３８５８９（ＪＰ，Ａ）
米国特許出願公開第２０１０／０１６９２３４（ＵＳ，Ａ１）
米国特許出願公開第２００６／０１１６１０５（ＵＳ，Ａ１）
米国特許第０６８０４３３７（ＵＳ，Ｂ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｑ　１０／００　－　９９／００