(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6210978号
(P6210978)

(45)発行日　平成29年10月11日(2017.10.11)　(24)登録日　平成29年9月22日(2017.9.22)

(51)Int.Cl.　　F I
G06F 15/00 (2006.01)　　G06F 15/00 410A
G06F 9/44 (2006.01)　　G06F 9/06 620C

請求項の数 7　(全 32 頁)

(21)出願番号　特願2014-515926(P2014-515926)
(86)(22)出願日　平成24年6月12日(2012.6.12)
(65)公表番号　特表2014-522542(P2014-522542A)
(43)公表日　平成26年9月4日(2014.9.4)
(86)国際出願番号　PCT/US2012/042102
(87)国際公開番号　WO2012/174021
(87)国際公開日　平成24年12月20日(2012.12.20)
審査請求日　平成27年6月12日(2015.6.12)
(31)優先権主張番号　13/159,174
(32)優先日　平成23年6月13日(2011.6.13)
(33)優先権主張国　米国(US)

(73)特許権者　314015767
マイクロソフト　テクノロジー　ライセンシング，エルエルシー
アメリカ合衆国　ワシントン州　９８０５２　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100101373
弁理士　竹内　茂雄
(74)代理人　100118902
弁理士　山本　修

最終頁に続く

(54)【発明の名称】ユーザー・インターフェース・オブジェクトの自動変換およびコード生成

(57)【特許請求の範囲】
【請求項１】
クライアント・コンピューター上で実行しているクライアント・アプリケーションから、変更されたユーザー・イベント・プロパティを受け取るステップであって、前記変更されたユーザー・イベント・プロパティは、前記クライアント・コンピューターにおいてレンダリングされたユーザー・インターフェース・エレメントに作用するユーザー・イベントによって引き起こされる、前記ユーザー・インターフェース・エレメントに割り当てられた属性に対する変更を含み、前記ユーザー・イベントは、１組のユーザー・イベント・プロパティによって定義され、前記ユーザー・インターフェース・エレメントは、特定のＧＵＩスクリーン内において構成されている、ステップと、
前記受け取られた変更されたユーザー・イベント・プロパティに基づいてグラフィカル・ユーザー・インターフェース（ＧＵＩ）独立オブジェクトを生成するステップと、
前記ＧＵＩ独立オブジェクトにベース・テンプレートとスクリーン・テンプレートを適用して新たなＧＵＩ独立オブジェクトを作成するステップであって、前記ベース・テンプレートは、ＧＵＩスクリーン・レイアウトの１つの表現についてのメタデーターおよびコンテンツを含み、前記スクリーン・テンプレートは、前記ベース・テンプレートに関係するＧＵＩスクリーン・レイアウトのカスタム化表現についてのメタデーターおよびコンテンツを含み、前記スクリーン・テンプレートは、前記特定のＧＵＩスクリーンに固有であり、前記スクリーン・テンプレートのレイアウトは、前記ベース・テンプレートのレイアウトをオーバーライドする、ステップと、

前記新たなＧＵＩ独立オブジェクトを前記クライアント・コンピューター上で実行している前記クライアント・アプリケーションへ送るステップと、
を含むコンピューター実装方法。
【請求項２】
前記スクリーン・テンプレートによってカスタム化された新たなＧＵＩレイアウト・スクリーンは、複数のスクリーンを１つのスクリーンに組み合わせる、請求項１に記載のコンピューター実装方法。
【請求項３】
ユーザー・イベント・プロパティを受け取るステップを含み、前記ユーザー・イベント・プロパティは、１つまたは複数のユーザー・インターフェース・エレメントを有するオブジェクト・メタデーターを含む、請求項１に記載のコンピューター実装方法。
【請求項４】
ユーザー・イベント・プロパティを受け取るステップを含み、前記ユーザー・イベント・プロパティは、１つまたは複数のタプルを有するプロパティ／値集合体を含み、各タプルは、少なくとも３つのフィールドを有し、各タプルは、ユーザー・インターフェース・エレメントの識別子、前記ユーザー・インターフェース・エレメントのプロパティ、および前記プロパティの値を含む、請求項１に記載のコンピューター実装方法。
【請求項５】
論理デバイスと、
前記論理デバイス上で動作可能なサーバー・アプリケーションと、
を備える装置であって、
前記サーバー・アプリケーションは、
クライアント・コンピューター上で実行しているクライアント・アプリケーションから、変更されたユーザー・イベント・プロパティを受け取り、前記受け取られた変更されたユーザー・イベント・プロパティに基づいてグラフィカル・ユーザー・インターフェース（ＧＵＩ）独立オブジェクトを生成するように動作可能であるインタプリター型ランタイム・エンジンであって、前記変更されたユーザー・イベント・プロパティは、前記クライアント・コンピューターにおいてレンダリングされたユーザー・インターフェース・エレメントに作用するユーザー・イベントによって引き起こされる、前記ユーザー・インターフェース・エレメントに割り当てられた属性に対する変更を含み、前記ユーザー・イベントは、１組のユーザー・イベント・プロパティによって定義され、前記ユーザー・インターフェース・エレメントは、特定のＧＵＩスクリーン内において構成されている、インタプリター型ランタイム・エンジンと、
前記ＧＵＩ独立オブジェクトにベース・テンプレートとスクリーン・テンプレートを適用して新たなＧＵＩ独立オブジェクトを作成し、前記新たなＧＵＩ独立オブジェクトを前記クライアント・コンピューター上で実行している前記クライアント・アプリケーションへ送るように動作可能であるテンプレート・プロセッサーであって、前記ベース・テンプレートは、ＧＵＩスクリーン・レイアウトの１つの表現についてのメタデーターおよびコンテンツを含み、前記スクリーン・テンプレートは、前記ベース・テンプレートに関係するＧＵＩスクリーン・レイアウトのカスタム化表現についてのメタデーターおよびコンテンツを含み、前記スクリーン・テンプレートは、前記特定のＧＵＩスクリーンに固有であり、前記スクリーン・テンプレートのレイアウトは、前記ベース・テンプレートのレイアウトをオーバーライドする、テンプレート・プロセッサーと、
を備える、装置。
【請求項６】
前記インタプリター型ランタイム・エンジンは、更に、前記受け取られたユーザー・イベント・プロパティに応答してスクリプト・コードを実行するように動作可能であるスクリプト・インタプリターを備える、請求項５に記載の装置。
【請求項７】
前記インタプリター型ランタイム・エンジンは、更に、データーベースに格納されてい

るファイルに対してファイル管理動作を実行するように動作可能であるファイル・マネージャーを備える、請求項５に記載の装置。

【発明の詳細な説明】

【背景技術】

【０００１】

[0001] クライアント－サーバー・アーキテクチャーは、分散型アプリケーション構造であり、サーバーおよびクライアントと呼ばれる２つの基本的エンティティ間で、アプリケーション・プログラムの計算タスクまたはワークロードを分割する。サーバーとは、リソースまたはサービスの提供側である。クライアントは、リソースまたはサービスの要求側である。サーバーは、物理的または論理的デバイスであり、そのリソースをクライアントと共有する１つ以上のサーバー・プログラムを実行する。クライアントは、物理的または論理的デバイスであり、通例、そのリソースのいずれも共有しないが、コンテンツまたはサービス機能をサーバーに要求する。クライアントおよびサーバーは、多くの場合、別個のハードウェア上においてコンピューター・ネットワークを通じて通信する。しかしながら、場合によっては、クライアントおよびサーバーの双方が同じシステムに存在することもある。したがって、クライアントは、到来する要求を待つサーバーとの通信セッションを開始する。

【０００２】

[0002] クライアント－サーバー・アーキテクチャーの一形態に、多層アーキテクチャー(multi-tier architecture)があり、ｎ－アーキテクチャーと呼ばれることが多い。ｎ－層アーキテクチャーは、アプリケーション・プログラムのある種の態様が多数の層に分離されているクライアント－サーバー・アーキテクチャーである。例えば、ユーザーとデーターベースとの間でデーター要求にサービスするためにミドルウェアを用いるアプリケーションは、多層アーキテクチャーを採用する。ｎ－層アプリケーション・アーキテクチャーは、開発者が柔軟で再利用可能なアプリケーションを作成するためのモデルを提供する。アプリケーションを多数の層に分解することによって、開発者は、特定の層（またはレイヤー）を変更または追加するだけで済み、これによってアプリケーション全体を書き換える必要性を回避する。

【発明の概要】

【０００３】

[0003] ｎ－層アーキテクチャーは、アプリケーション・プログラムを開発および変更するときに、多くの利点をもたらす。しかしながら、多数のクライアントがあるウェブ・ベース環境に合わせてｎ－層アーキテクチャーを開発するときには、多くの困難がある。各クライアントは、異なるウェブ・ブラウザー、ウェブ・サービス、およびウェブ・アプリケーションを含む、異なるウェブ技術を利用する可能性がある。更に、ウェブ技術は、多くの異なるタイプの基礎ハードウェアおよびソフトウェア・アーキテクチャーと共に作業するように設計されており、異なる入力／出力（Ｉ／Ｏ）コンポーネント、フォーム・ファクタ、電力要件、処理能力、通信能力、メモリー・リソース等を有する種々のデバイスを含む。したがって、これらの多くの異質なデバイスおよびアーキテクチャーに跨がって１つ以上の層を実現することは、困難な場合もある。更に、アプリケーション・プログラムのウェブ・バージョンは、アプリケーション・プログラムのウェブ・バージョン以外とは互換性がない場合もあり、各々に別個のソフトウェア・アーキテクチャーの必要性が生ずる。本改良が必要なのは、これらおよびその他の欠点に関してである。

【０００４】

[0004] この摘要は、詳細な説明の章において以下で更に説明する概念から選択したものを簡略化された形式で紹介するために、設けられている。この摘要は、特許請求する主題の主要な特徴や必須の特徴を特定することを意図するのではなく、特許請求する主題の範囲を判断するときに補助として意図するのでもない。

【０００５】

[0005] 総じて、種々の実施形態は、例えば、商用基幹業務アプリケーション・プログ

ラムのような、異なるタイプのアプリケーション・プログラムを実行するのに適したクライアント－サーバー・アーキテクチャーを対象とする。一部の実施形態は、特に、少なくとも１つのプレゼンテーション層を含む、アプリケーション・プログラムの多数の層（またはレイヤー）を有するｎ－層クライアント－サーバー・アーキテクチャーを対象とする。 一実施形態では、例えば、3－層クライアント－サーバー・アーキテクチャーが、プレゼンテーション層を含むことができる。このプレゼンテーション層は、多くの異なるタイプのクライアントと共に動作するようにインタプリター型ランタイム・エンジン・アプリケーション(interpretive runtime engine application)を適応させるときに、ユーザー・イベントのグラフィカル・ユーザー・インターフェース（ＧＵＩ）のレンダリングを分離し改良するように設計された技法を用いて実現される。

【０００６】

[0006] 一実施形態では、例えば、装置は、サーバー・プログラムを実行するように構成されている論理デバイスを備えることができる。サーバー・アプリケーションは、エレメントの中でもとりわけ、１組の受け取ったユーザー・イベント・プロパティからグラフィカル・ユーザー・インターフェース（ＧＵＩ）独立オブジェクトを生成するために、インタプリター型ランタイム・エンジンを備えることができる。このＧＵＩ独立オブジェクトは、テンプレート・プロセッサーで処理されて(subject)、新たなＧＵＩ依存オブジェクトを作成し、この新たなＧＵＩ依存オブジェクトが、レンダリングのために、クライアント・アプリケーションに返される。他の実施形態についても説明し、特許請求する。

【０００７】

[0007] 以上のおよびその他の特徴ならびに利点は、以下の詳細な説明を読み、添付図面を検討することから明らかになるであろう。尚、以上の概略的な説明および以下の詳細な説明は例示に過ぎず、特許請求する態様を限定するのではないことは言うまでもない。

【図面の簡単な説明】

【０００８】

【図１Ａ】図１Ａは、従来のデスクトップ・アプリケーション・アーキテクチャーを示す。

【図１Ｂ】図１Ｂは、従来の２層アプリケーション・アーキテクチャーを示す。

【図１Ｃ】図１Ｃは、従来の３層アプリケーション・アーキテクチャーを示す。

【図２】図２は、一実施形態による、多数のクライアントおよびクライアント・アダプターを有する改良ｎ－層クライアント－サーバー・アーキテクチャーのブロック図を示す。

【図３】図３は、一実施形態による、１つのクライアントおよびクライアント・アダプターを有する改良ｎ－層クライアント－サーバー・アーキテクチャーのブロック図を示す。

【図４】図４は、一実施形態による、クライアントおよびクライアント・アダプターのためにグラフィカル・ユーザー・インターフェース（ＧＵＩ）独立オブジェクトを有する改良ｎ－層クライアント－サーバー・アーキテクチャーのブロック図を示す。

【図５】図５は、一実施形態による改良ｎ－層クライアント－サーバー・アーキテクチャーの論理フローを示す。

【図６Ａ】図６Ａは、一実施形態によるＧＵＩ独立オブジェクトの論理図を示す。

【図６Ｂ】図６Ｂは、一実施形態による特定のＧＵＩ独立オブジェクトの論理図を示す。

【図７】図７は、一実施形態による改良ｎ－層クライアント－サーバー・アーキテクチャーの第２の論理フローを示す。

【図８Ａ】図８Ａは、一実施形態にしたがって、ＧＵＩオブジェクト・レイアウトを表すテンプレートを処理する改良ｎ－層クライアント－サーバー・アーキテクチャーのブロック図を示す。

【図８Ｂ】図８Ｂは、一実施形態による、代表的ＧＵＩオブジェクト・レイアウトから生成された第１ユーザー・インターフェース・ビューを示す。

【図８Ｃ】図８Ｃは、一実施形態による、代表的ＧＵＩオブジェクト・レイアウトから生成された第１ユーザー・インターフェース・ビューを示す。

【図９】図９は、一実施形態によるテンプレート処理システムの第３論理フローを示す。

【図１０】図１０は、一実施形態による、改良ｎ－層クライアント－サーバー・アーキテクチャーに適した計算アーキテクチャーの一実施形態を示す。
【図１１】図１１は、一実施形態による改良ｎ－層クライアント－サーバー・アーキテクチャーに適した通信アーキテクチャーの一実施形態を示す。
【発明を実施するための形態】
【０００９】
[0024] 種々の実施形態は、総じて、異なるタイプの商用基幹業務アプリケーション・プログラムを実行するのに適したクライアント－サーバー・アーキテクチャーを対象とする。一部の実施形態は、特に、改良ｎ－層クライアント－サーバー・アーキテクチャーを対象とし、ここでｎは、いずれかの正の整数を表す値である。改良ｎ－層アーキテクチャーは、多数の層（またはレイヤー）のアプリケーション・プログラムを備えており、少なくとも１つのプレゼンテーション層を含むことができる。一実施形態では、例えば、改良ｎ－層アーキテクチャーは、少なくとも１つのプレゼンテーション層と、アプリケーション処理層と、データー管理層とを備えている３層アーキテクチャーとして実現することができる。プレゼンテーション層は、通常、入力／出力動作を扱うというような、ユーザー・インターフェース・ロジックを実装する。アプリケーション処理層は、通常、１組のアプリケーション規則にしたがって、データー処理のような、アプリケーションまたはビジネス・ロジックを実装する。データー管理層は、通常、データー方式の定義、データー格納、データー・クエリーの処理等というような、データー格納およびアクセスを実現する。
【００１０】
[0025] 改良ｎ－層クライアント－サーバー・アーキテクチャーは、インタプリター型ランタイム・エンジンを用いてアプリケーションにおけるＧＵＩレンダリングおよびユーザー・イベントの分離および最適化を容易にするように設計された技法を用いて実現される少なくとも１つのプレゼンテーション層を含むことができる。これによって、インタプリター型ランタイム・エンジン・アプリケーションを２層クライアント－サーバーに基づくアーキテクチャーから、ホストされた３層環境(hosted 3-tier environment)に改造しつつ、インタプリター型ランタイム・エンジン・アプリケーションに対する変更を減少させることが可能になる。
【００１１】
[0026] 図１Ａ、図１Ｂ、および図１Ｃは、改良ｎ－層クライアント－サーバー・アーキテクチャーの種々の実施形態について利点を強調するために、背景として、アプリケーション開発のための従来のアーキテクチャーを示す。図１Ａは、従来のデスクトップ・アーキテクチャーを示す。図１Ｂは、従来の２層アーキテクチャーを示す。図１Ｃは、従来の３層（またはｎ－層）アーキテクチャーを示す。
【００１２】
[0027] 図１Ａは、アプリケーション・プログラム１１２の全ての部分（またはアプリケーション・レイヤー）がクライアント・コンピューター１１０（例えば、デスクトップ・コンピューター）上に実装されているデスクトップ・アーキテクチャー１００の一例である。アプリケーション・プログラム１１２は、例えば、ユーザー・インターフェース（ＵＩ）ロジック、ビジネス・ロジック、およびデーターベース・アクセス・ロジックを実装する種々のアプリケーション・レイヤーを備えることができる。アプリケーション・プログラム１１２は、データーベース１１４からのアプリケーション・データーを格納すること、およびこのデーターにアクセスすることができる。データーベース１１４もクライアント・コンピューター１１０上に実装されている。
【００１３】
[0028] 図１Ｂは、データーベース１１４がここではクライアント・コンピューター１１０から離れている、２層アーキテクチャー１２０の一例である。２層アーキテクチャー１２０では、アプリケーション・プログラム１１２およびこの構成アプリケーション・レイヤーが未だクライアント・コンピューター１１０上に存在する。しかしながら、データ

ーベース１１４はクライアント・コンピューター１１０からデーターベース・サーバー１１６に移動している。クライアント・コンピューター１１０において実行するアプリケーション・プログラム１１２は、データーベース・アプリケーション・プログラム・インターフェース（ＡＰＩ）を介して、データーベース・サーバー１１６に、データー要求を送る。データーベース・サーバー１１６は、データーベース１１４に通信可能に結合されている。次いで、クライアント・コンピューター１１０上で実行するアプリケーション・プログラム１１２に、要求したデーターが戻される。

【００１４】

[0029] 図１Ｃは、３層アーキテクチャー１３０の一例である。３層アーキテクチャー１３０では、アプリケーション・プログラム１１２を、それぞれクライアント・コンピューター１１０およびサーバー１２２上で実行する分散アプリケーション・プログラム１１２、１２４に分離することができる。アプリケーション・プログラム１１２は、ＵＩロジックを有するアプリケーション・レイヤーを実現することができる。アプリケーション・プログラム１２４は、ビジネスおよびデーターベース・アクセス・ロジックを有するアプリケーション・レイヤーを実現することができる。クライアント・コンピューター１１０において実行するアプリケーション・プログラム１１２は、アプリケーション・プログラム１２４を実行しているサーバー１２２にデーターを送ることができる。アプリケーション・プログラム１２４は、次いで、ビジネス・ロジックを実行し、データー要求をデーターベース・サーバー１１６に送ることができる。データー・サーバー１１６は、データーベース１１４に通信可能に結合されている。要求されたデーターおよびビジネス・ロジックを実行した結果は、次に、アプリケーション・プログラム１１２に戻され、クライアント・コンピューター１１０においてレンダリングされる。尚、データーベース・サーバー１１６は、サーバー１２２と一緒に配置してもよく、またはサーバー１２２の一部であってもよいことは、注記してしかるべきである。言い換えると、このハードウェア・アーキテクチャーは、１つのサーバー１２２がアプリケーションおよびデーターベース・サーバー双方として機能するようにしてもよい。２層および３層（またはｎ層）アーキテクチャー間で区別する要素は、アプリケーション・レイヤーの一部または多くをクライアント・コンピューター１１０から外部に移動させ、１つ以上の他のサーバー１１６、１２２間で分散させることである。

【００１５】

[0030] ３層アーキテクチャー１３０のような、ｎ－層アーキテクチャーは、アプリケーション・プログラムを開発および変更するときに、２層アーキテクチャー１２０に対して多くの利点をもたらすことができる。例えば、アプリケーション・プログラム全体の完全な書き換えを行うことなく、１つの層を変更または追加することができる。しかしながら、多数のクライアントがあるウェブ・ベース環境に合わせてｎ－層アーキテクチャーを実現するには、困難がある。各クライアントは、異なるウェブ・ブラウザー、ウェブ・サービス、およびウェブ・アプリケーションを含む異なるウェブ技術を利用している可能性がある。更に、ウェブ技術は、異なる入力／出力（Ｉ／Ｏ）コンポーネント、フォーム・ファクタ、電力要件、処理能力、通信能力、メモリー・リソース等を含む、種々の異なるタイプの基礎ハードウェアおよびソフトウェアと共に作業するように設計されている。したがって、各クライアントの一意の構成に一致させるためにプレゼンテーション層の広範なカスタム化を行わずに、これら多くの異質なデバイスおよびアーキテクチャーにまたがって均一に、プレゼンテーション層のような、所与のアプリケーション・レイヤーを実現するのは困難である可能性がある。更に、アプリケーション・プログラムのウェブ・バージョンが、アプリケーションのウェブ・バージョン以外とは互換性がない場合もあり、これによって各々毎に別個のソフトウェア・アプリケーションの必要性が生ずる。

【００１６】

[0031] 種々の実施形態において、改良ｎ－層アーキテクチャーは、２層クライアント－サーバー・アーキテクチャーを、アプリケーション・プログラムのプレゼンテーション層に薄いクライアントを利用する３層アプリケーション・アーキテクチャーへの移動を可

能にするフレームワークを設ける。一実施形態では、例えば、各クライアント・デバイスが、ウェブ・クライアントの形態とした薄いクライアントを実装することができる。通例、ウェブ・クライアントは、例えば、クライアント・コンピューターにおいて動作するウェブ・ブラウザーのような、ウェブ技術を用いて実装される薄いクライアント・アプリケーションを指す。また、これは、サイトまたはサーバーからカスタム・サービスをサポートするようにブラウザーを改良するプラグインおよびヘルパー・アプリケーション(helper application)も指すことができる。本明細書においてウェブ・クライアントに言及する場合はいつでも、ウェブ・ブラウザーの機能にも言及することができる。

【００１７】

[0032] 図２は、クライアント－サーバー・システム２００を示す。一実施形態では、クライアント－サーバー・システム２００は、改良ｎ－層クライアント－サーバー・システムを構成することができる。改良ｎ－層クライアント－サーバー・システムは、アプリケーション・プログラムを、少なくとも１つのプレゼンテーション層を含む多数の層に分離することができる。プレゼンテーション層は、インタプリター型ランタイム・エンジンを用いてアプリケーション・プログラムにおけるＧＵＩレンダリングおよびユーザー・イベントの分離および最適化を容易にするように設計された技法を用いて実現することができる。これによって、インタプリター型ランタイム・エンジン・アプリケーションを２層クライアント－サーバーに基づくアーキテクチャーから、ホストされた３層環境に改造しつつ、インタプリター型ランタイム・エンジン・アプリケーションに必要とされる変更を減少させることが可能になる。

【００１８】

[0033] 図１Ａを参照して既に説明したように、多くのアプリケーションは２層アプリケーション・アーキテクチャーに従う。２層アプリケーション・アーキテクチャーでは、アプリケーションは、２つの相互に関係するコンポーネント、即ち、データーベース・サーバーおよびクライアント・アプリケーションに編成される。データーベース・サーバーは、システムおよび会社データーを、拡張ビジネス・ロジックと共にホストすることができる。拡張ビジネス・ロジックは、クライアントにおいて実行するには時間がかかりすぎる、重い方の動作の一部を処理することを可能にする。一方、クライアント・アプリケーションは、機能の中でもとりわけ、ＵＩを配信し、データー入力の有効性判断を行い、報告をレンダリングする機能を実行することができる。

【００１９】

[0034] 図２に示す例示実施形態では、クライアント－サーバー・システム２００は、サーバー２０２と多数のクライアント２０４、２０６を備えることができる。異なるハードウェア・プラットフォーム上に実装するとき、サーバー２０２およびクライアント２０４、２０６は、ネットワーク２５０を通じて、互いに通信することができる。同じハードウェア・プラットフォーム上に実装される場合、サーバー２０２およびクライアント２０４、２０６は、適したバス技術およびアーキテクチャーによって、互いに通信することができる。明確にするために、図２は１つのサーバー２０２および２つのクライアント２０４、２０６のみを示すが、クライアント－サーバー・システム２００は、所与の実施形態に合わせて所望通りに、あらゆる数のサーバーおよびクライアントでも実装できることは認めることができよう。実施形態は、このコンテキストにおいて限定されることはない。

【００２０】

[0035] 一実施形態では、サーバー２０２は、サーバー・アプリケーション２１０を実装する電子デバイスを備えることができる。サーバー・アプリケーション２１０は、商用基幹業務アプリケーションのような、あらゆるタイプのサーバー・アプリケーションでも構成することができる。商用基幹業務アプリケーションの例には、限定ではなく、会計プログラム、企業リソース計画（ＥＲＰ）アプリケーション、顧客関係管理（ＣＲＭ）アプリケーション、サプライ・チェーン管理（ＳＣＭ）アプリケーション等を含むことができる。商用基幹業務アプリケーションは、場合によっては、「中間層」アプリケーションと呼ばれることもある。何故なら、これらは、通例、デスクトップ・コンピューターのよう

なクライアント・デバイスではなく、商用の企業ネットワークにおいてサーバーまたはサーバー・アレイによって実行されるからである。具体的な例には、ワシントン州、RedmondのMicrosoft Corporationが製造するMicrosoft（登録商標）Dynamics GPを含むことができる。Microsoft Dynamics GPは、商用会計ソフトウェア・アプリケーションである。商用基幹業務アプリケーションの他の具体的な例には、ワシントン州、RedmondのMicrosoft Corporationが製造するMicrosoft Dynamics（登録商標）AXを含むことができる。Microsoft Dynamics AXは、商用ERPソフトウェア・アプリケーションである。しかしながら、実施形態はこれらの例に限定されるのではない。

【００２１】

[0036] サーバー２０２がサーバー・アプリケーション２１０のコードを実行しているとき、サーバー２０２はインタプリター型ランタイム・エンジン２１２を形成する。インタプリター型ランタイム・エンジン２１２は、サーバー・アプリケーション２１０に対して多数のアプリケーション・レイヤーを実装する。クライアント－サーバー・システム２００では、これらのアプリケーション・レイヤーを、アプリケーション・ロジック２１４、データーベース・ロジック２１６、およびサーバー・プレゼンテーション・ロジック２１８と呼ぶ。サーバー・アプリケーション２１０は、ネットワーク２５０を通じて信号またはメッセージの形態でクライアント２０４、２０６から受信する制御指令(control directive) によって制御し動作させることができる。

【００２２】

[0037] 一実施形態では、クライアント２０４、２０６は、各々、それぞれのウェブ・クライアント２３０、２４０を実装する電子デバイスを備えることができる。ウェブ・クライアント２３０、２４０は、各々、例えば、それぞれのクライアント２０４、２０６上で実行するウェブ・ブラウザーのインスタンスを備えることができる。また、ウェブ・ブラウザーは、プラグイン、ウェブ・アプリケーション、およびサーバー２０２からのカスタム・サービスをサポートするためにウェブ・ブラウザーを改良するように構成されたヘルパー・アプリケーションも含むことができる。本明細書において、ウェブ・クライアント２３０、２４０に言及するときはいつでも、ウェブ・ブラウザーの機能にも言及することができる。

【００２３】

[0038] クライアント２０４、２０６は、それぞれのクライアント・アダプター２３２、２４２を備えることができる。クライアント・アダプター２３２、２４２の各々は、所与のクライアント２０４、２０６と共に用いるように構成されている。このため、サーバー・アプリケーション２１０およびインタプリター型ランタイム・エンジン２１２は、異なるウェブ技術を用いて異なるクライアントによってアクセスされるときでも、変更する必要がない。

【００２４】

[0039] クライアント・アダプター２３２、２４２は、それぞれのクライアント・プレゼンテーション・ロジック２３８、２４８を備えることができる。クライアント・プレゼンテーション・ロジック２３８、２４８は、例えば、ディジタル・ディスプレイのような、クライアント２０４、２０６の出力デバイス上に、ユーザー・インターフェースのエレメントまたはビュー(view)を提示するように設計することができる。クライアント・プレゼンテーション・ロジック２３８、２４８は、サーバー・アプリケーション２１０に合わせて実現された分散ｎ－層アーキテクチャーにしたがって、サーバー２０２上で実行するサーバー・アプリケーション２１０のアプリケーション・ロジック２１４、データベース・ロジック２１６、およびサーバー・プレゼンテーション・ロジック２１８と相互動作するように設計することができる。

【００２５】

[0040] クライアント・アダプター２３２、２４２、およびそれぞれのクライアント・プレゼンテーション・ロジック２３８、２４８は、異なるクライアント２０４、２０６によってサーバー・アプリケーション２１０にアクセスすることを可能にするために、サー

バー・プレゼンテーション・ロジック２１８と相互動作することができる。各クライアント２０４、２０６は、クライアント２０４、２０６の特定の構成に適応させるために、それぞれのクライアント・プレゼンテーション・ロジック２３８、２４８として、サーバー・プレゼンテーション・ロジック２１８の異なるバージョンを実装することができる。これは、サーバー・プレゼンテーション・ロジック２１８を書き換える必要なく遂行することができ、更に重要なことは、ビジネス・ロジック２１４およびデータベース・ロジック２１６も書き換える必要なく、遂行できることである。更に、サーバー・プレゼンテーション・ロジック２１８およびクライアント・プレゼンテーション・ロジック２３８、２４８は、ネットワーク２５０の通信トラフィックおよびオーバーヘッドを低減するように相互作用することができ、これによって、通信遅延に伴うレイテンシーを低減しつつ、速度および性能を高めることができる。

【００２６】

[0041] 種々の実施形態において、サーバー・プレゼンテーション・ロジック２１８およびクライアント・プレゼンテーション・ロジック２３８、２４８は、グラフィカル・ユーザー・インターフェース（ＧＵＩ）独立オブジェクト２６０を利用した効率的なやり方で相互作用することができる。ＧＵＩ独立オブジェクト２６０は、ＧＵＩ画面（例えば、Microsoft Windows（登録商標）のForms）がデスクトップ環境とウェブ環境との間で自由に移動することを可能にする。ＧＵＩ独立オブジェクト２６０は、サーバー・アプリケーション２１０が背景においてサービスとして実行することを可能にし、従前からのＯＳフォームまたはウェブ・クライアント・フォームのいずれかによって受信することができるユーザー・イベントを待ちつつ、提出されたフォームのタイプには関係なく、スクリプト・イベントを実行することができる。

【００２７】

[0042] ＧＵＩ独立オブジェクト２６０は、情報のタイプの中でもとりわけ、ユーザー・イベント、およびアプリケーション・ロジック・イベントに影響を及ぼし得るユーザー・イベントのプロパティに加えて、クライアント・アダプター２３２、２４２によるＧＵＩ依存レンダリングに影響を及ぼし得るあらゆるユーザー・イベント・プロパティを収容することができる。ＧＵＩ独立オブジェクト２６０は、生成され、インタプリター型ランタイム・エンジン２１２からクライアント・アダプター２３２、２４２に送られ、その後、それぞれのクライアント・プレゼンテーション・ロジック２３８、２４８によって、クライアント・ユーザー・インターフェースにおいてレンダリングされる。

【００２８】

[0043] 図３は、ｎ－層クライアント－サーバー・システム３００の具体的な実施態様を示す。クライアント－サーバー・システム３００は、サーバー３０２およびクライアント３０４を備えることができる。サーバー３０２は、例えば、図２を参照して説明したサーバー２０２を表すことができる。クライアント３０４は、例えば、図２を参照して説明したクライアント２０４、２０６の一方または双方を表すことができる。

【００２９】

[0044] クライアント－サーバー・システム３００において示す例示の実施形態では、サーバー３０２はサーバー・アプリケーション３１０を実装することができる。一実施形態では、例えば、サーバー・アプリケーション３１０は、適したタイプのプログラミング言語の中でもとりわけ、Microsoft Dexterity（登録商標）プログラミング言語を用いてコード化することができる。Microsoft Dexterityアプリケーションとして実装されたときは、サーバー・アプリケーション３１０を全体的に２つの別個のエレメントに分割することができる。第１エレメントはインタプリター型ランタイム・エンジン３１２であり、オペレーティング・システム（ＯＳ）と通信する、およびファイル・マネージャー３１６を通じてデーターベース３２０への接続を管理するというような、アプリケーション環境の技術的側面を担当する(address)。第２エレメントは、アプリケーション規則、ビジネス規則、フォーム、報告、リソース、メタデーター、ならびにユーザーのコマンドおよび入力への応答をイネーブルするアプリケーション・コードというような、アプリケーショ

ン・ロジック３１５をホストするアプリケーション・ディクショナリー(application dictionary)３１３である。このアーキテクチャーは、アプリケーション・ロジック３１５を、例えば、プラットフォームＯＳに対するアップグレードのように、ＵＩスタイルの変更やプラットフォームの進展から隔離する。

【００３０】

[0045] sanScriptコードは、アプリケーションがどのように動作するか制御するために用いられる。sanScriptコードは、通例、小さいセグメント、またはスクリプト単位で書かれ、フィールド、メニュー、スクリーン、およびフォームというような、アプリケーション・ディクショナリー３１３におけるオブジェクトに添付される。スクリプトは、ユーザーがアプリケーションにおけるその特定のオブジェクトと対話処理すると実行する。例えば、プッシュ・ボタンに適用されるスクリプトは、ユーザーがこのボタンをクリックしたときに実行する。

【００３１】

[0046] 図示のように、クライアント３０４は、ウェブ・クライアント３３０を構成することができる。ウェブ・クライアント３３０は、例えば、ウェブ・クライアント２３０、２４０の一方または双方を表すことができる。ウェブ・クライアント３３０は、ユーザー・インターフェースおよびユーザーの対話処理に向けられる１組のコンポーネントおよびサービスを配信することができ、サーバー・アプリケーション３１０と共に使用するためのユーザー入力および軽量ユーザー・インターフェース制御を含む。３層アーキテクチャーへの滑らかな移行を達成するためには、しかしながら、ウェブ・クライアント・アーキテクチャーの導入によって生ずる多数の技術課題を克服して効率的なウェブ・クライアント・インターフェースを可能にする必要がある。

【００３２】

[0047] 本明細書において説明する実施形態の目標は、既存のコードおよびＧＵＩメタデーターに必要な変更を減らすことである。先に述べた課題の一部を解決するために、種々の実施形態は、ユーザー・インターフェース・マネージャー３１８およびＯＳレンダリング・エンジン３２２を、インタプリター型ランタイム・エンジン３１２から切断する技法を対象とする。ユーザー・インターフェース・マネージャー３１８は、ＧＵＩスクリーンのような種々のユーザー・インターフェース・エレメントの所与のＧＵＩシステム内における配置および外観を制御するシステム・ソフトウェアである。ＯＳレンダリング・エンジン３２２は、コンテンツを表示するためのシステム・ソフトウェアである。インタプリター型ランタイム・エンジン３１２は、サーバー・アプリケーション３１０の実行バージョン(executed version)である。

【００３３】

[0048] フォーム（またはスクリーン）の使用は、あらゆるMicrosoft Dexterityアプリケーションの中核コンポーネントである。フォームは、ユーザーがサーバー・アプリケーション３１０と対話処理するときに用いられるメカニズムである。サーバー・アプリケーション３１０が、例えば、Microsoft Dexterityアプリケーションとして実装されるとき、Microsoft Dexterityスクリーンは、通例、そのスクリーンの制御に関連するsanScriptコードを含む。このsanScriptコードは、スクリプト・インタプリター３１４の指示の下でスクリーンおよび制御（例えば、トランザクションを保存する、バッチをポストする）の意図する機能が与えられると、ユーザー・イベントに応答して実行する。

【００３４】

[0049] サービス・アプリーケーション３１０のウェブ・バージョン以外では、ＵＩはユーザー・インターフェース・マネージャー３１８によって管理される(administer)。一方、ユーザー・インターフェース・マネージャー３１８は、ＯＳレンダリング・エンジン３２２と通信して、表示画面上に、実際のMicrosoft Dexterityスクリーンを、開発者によって既に配置が決められている制御エレメントと共に表示する。

【００３５】

[0050] しかしながら、クライアント－サーバー・システム３００のウェブ・クライア

ント３層アーキテクチャーへの移行を容易にするために、ユーザー・インターフェース・マネージャー３１８およびＯＳレンダリング・エンジン３２２を、インタプリター型ランタイム・エンジン３１２の機能から切断することができる。これによって、ウェブ・クライアント３３２は、ユーザー・インターフェース・マネージャー３３６およびレンダリング・エンジン３３８のクライアント・バージョンをクライアント３０４上に実装することが可能になる。更に、これによって、サーバー３０２上で実行しているインタプリター型ランタイム・エンジン３１２が、ウェブ・クライアント３３２による使用のためのＧＵＩ独立オブジェクト３６０を生成することも可能にする。ＧＵＩ独立オブジェクト３６０によって、古いクライアントが典型的なＧＵＩスクリーン（例えば、Microsoft Win32（登録商標）スクリーン）を配給し(serve up)続けることができ、一方、クライアント３０４のウェブ・クライアント３３０も、その同じスクリーンのウェブ・ベース表現を配給することができ、サーバー・アプリケーション３１０の基礎アプリケーション・ロジック３１５のいずれも変更する必要はない。

【００３６】

[0051] ユーザー・インターフェース・マネージャー３１８およびＯＳレンダリング・エンジン３２２をインタプリター型ランタイム・エンジン３１２から切断することによって、スクリーン（フォーム）が自由に、ウェブ外（例えば、デスクトップまたはWin32）環境およびウェブ環境の間で移動することが可能になる。ユーザー・インターフェース・マネージャー３１８およびＯＳレンダリング・エンジン３２２を切断することによって、サーバー・アプリケーション３１０は、背景においてサービスとして実行することができ、従前からのWin32フォームまたはウェブ・クライアント・フォームのいずれかによって受けることができるユーザー・イベントを待ちつつ、なおもそれが提出されたフォームのタイプには関係なくスクリプト・イベントを実行することができる。

【００３７】

[0052] この切断を容易にするために、サーバー・アプリケーション３１０のＧＵＩ依存およびＧＵＩ独立処理レイヤーを最初に分離する。これら２つのレイヤー間における直接通信の代わりに、ＧＵＩ独立オブジェクト３６０を用いて、レンダリングおよびイベント・メタデーターを露出する。ＧＵＩ独立オブジェクト３６０は、アプリケーション・ロジック・イベントに影響を及ぼし得るユーザー・イベント・プロパティに加えて、クライアント・アダプター３３２によるＧＵＩ依存レンダリングに影響を及ぼし得るあらゆるユーザー・イベント・プロパティも収容することができる。次いで、ＧＵＩ独立オブジェクト３６０を（ＧＵＩ依存）クライアント・アダプター３３２に送り、このクライアント・アダプター３３２を、クライアント・ユーザー・インターフェースの画面において、クライアント３０４用ディスプレイ上にレンダリングする。クライアント・アダプター３３２の例には、とりわけ、Microsoft Silverlight （登録商標）、ＨＴＭＬ、Win32 GDI、.NetFormsが含まれるが、必ずしもこれらに限定されるのではない。

【００３８】

[0053] 図４は、ｎ－層クライアント－サーバー・システム４００の具体的な実施態様を示す。クライアント－サーバー・システム４００は、サーバー４０２およびクライアント４０４を備えることができる。サーバー４０２は、例えば、図２、図３を参照して説明したサーバー２０２、３０２を表すことができる。クライアント４０４は、例えば、図２、図３を参照して説明したクライアント２０４、２０６、３０４の内１つまたは全部を表すことができる。

【００３９】

[0054] サーバー４０２上には、サーバー・アプリケーション４１０があり、サーバー・アプリケーション４１０は、インタプリター型ランタイム・エンジン４１２を含むことができる。インタプリター型ランタイム・エンジン４１２は、1つ以上のアプリケーション・レイヤーを実行することを役割とするか、または１つ以上のアプリケーション・レイヤーを実行する他のコンポーネントと結合されていてもよい。インタプリター型ランタイム・エンジン４１２は、更に、スクリプト・インタプリター４１４、ファイル・マネージ

ャー４１６、およびユーザー・インターフェース・マネージャー４１８を備えることができる。スクリプト・インタプリター４１４は、ファイル・マネージャー４１６およびサーバー・ユーザー・インターフェース・マネージャー４１８と通信することができる。ファイル・マネージャー４１６は、データーベース４２０と通信することもできる。

【００４０】

[0055] クライアント４０４上には、クライアント・アダプター４３２を実行するウェブ・クライアント４３０がある。クライアント・アダプター４３２は、ユーザー・インターフェース・マネージャー４３６と、図２に示したクライアント・プレゼンテーション・ロジック２３８、２４８にしたがって、クライアント・ユーザー・インターフェースのような、クライアント・ユーザー・インターフェースにおいてコンテンツを表示するためのレンダリング・エンジン４３８を含むことができる。

【００４１】

[0056] 図４は、３層アプリケーション・アーキテクチャーを表すことができ、一定のアプリケーション・レイヤーをサーバー４０２とクライアント４０４との間で分散させることができる。例えば、クライアント・プレゼンテーション・ロジック２３８および／または２４８はクライアント４０４上に存在することができ、一方アプリケーション・ロジック２１４およびデーターベース・ロジック２１６は、図２に示したように、サーバー４０２上に分散させることができる。図４に示すアーキテクチャーでは、ユーザー・インターフェース・マネージャー４３６およびレンダリング・エンジン４３８の機能が、サーバー４０２上のインタプリター型ランタイム・エンジン４１２から切断され、クライアント４０４上のクライアント・アダプター４３２と共にその機能が配置されている。

【００４２】

[0057] 一実施形態では、インタプリター型ランタイム・エンジン４１２は、スクリプト・インタプリター４１４を含むことができる。スクリプト・インタプリター４１４は、一般に、トランザクションを保存することまたはバッチをポストすることというような、ユーザー・イベントに応答して、スクリプト・コード(scripted code)を実行するように構成されているが、ユーザー・イベントはこれらに限定されるのではない。スクリプト・コードの例には、スクリプトのタイプの中でもとりわけ、プリスクリプト、変更スクリプト、およびポスト・スクリプトを含むことができる。

【００４３】

[0058] 一実施形態では、インタプリター型ランタイム・エンジン４１２は、ファイル・マネージャー４１６を含むことができる。ファイル・マネージャー４１６は、一般に、データーベース４２０に格納されているファイルに対するファイル管理動作を実行するように構成することができる。ファイル管理動作の例には、とりわけ、ファイルを作成する、ファイルを開く、ファイルをコピーする、ファイルを移動する、ファイルを削除することを含むことができる。

【００４４】

[0059] 一実施形態では、インタプリター型ランタイム・エンジン４１２は、ユーザー・インターフェース・マネージャー４３６を含むことができる。ユーザー・インターフェース・マネージャー４３６は、一般に、所与のＧＵＩシステムを実現するユーザー・インターフェース内において、スクリーン・エレメントのような、種々のユーザー・インターフェース・エレメントの配置および外観を制御するように構成することができる。

【００４５】

[0060] 動作において、ユーザーは、ウェブ・クライアント４３０を通じて、クライアント・ユーザー・インターフェースと対話処理することができる。ウェブ・クライアント４３０は、ウェブ・ベース・コンテンツをレンダリングするためのユーザー・インターフェース・コードを有するウェブ・ブラウザーを備えることができる。ウェブ・クライアント４３０は、とりわけ、ＨＴＭＬ、ＸＨＴＭＬ、およびＸＭＬというような、種々のウェブ技術を用いて実現することができる。ウェブ・クライアント４３０の例には、ウェブ・ブラウザー・ソフトウェアのタイプの中でもとりわけ、ワシントン州、RedmondのMicroso

ft Corporationが製造するInternet Explorer（登録商標）を含むことができるが、これに限定されるのではない。

【００４６】

[0061] 一実施形態によれば、動作において、ユーザーはウェブ・クライアント４３０を通じてクライアント・ユーザー・インターフェースと対話処理することができ、ユーザー・イベントを入力することができる。このユーザー・イベントは、クライアント・アダプター４３２によって受け取られ、処理することができる。ユーザー・イベントの例には、限定ではなく、ポインターをあるフィールドに移動する、フィールド上方でホバリングする、フィールドを選択する、ボタン上でのマウス・クリック、テキスト・フィールドに記入する、および同様の動作を含むことができる。ユーザー・イベントは、１組のユーザー・イベント・プロパティを用いて定義することができる。一実施形態では、１組のユーザー・イベント・プロパティ全体ではなく、ユーザー・イベント・プロパティに対する変更のみをウェブ・クライアント４３０からサーバー・アプリケーション４１０に送ればよい。この差分技法によって、通信帯域幅を保存し、レイテンシーを短縮することができる。

【００４７】

[0062] ユーザー・イベント・プロパティは、ユーザー・インターフェース・レイアウト内に表示されるフィールド、スクリーン、またはグラフィカル・オブジェクトというような、ユーザー・インターフェース・エレメントに割りあてることができる属性であればいずれでもよい。ユーザー・イベント・プロパティは、対応するユーザー・インターフェース・エレメントについて提示様式または提示フォーマットの属性を記述する。ユーザー・イベント・プロパティは、情報のタイプの中でもとりわけ、ユーザー・インターフェース・エレメント識別子（ＩＤ）、プロパティ（例えば、境界線、フォント、フォント・サイズ、フォント・カラー、背景、背景カラー、様式、右位置合わせ、中央位置合わせ、右位置合わせ、１スペース、ダブル・スペース等）、およびプロパティ値（例えば、偽、真、０、１等）を含むことができる。例えば、ＧＵＩスクリーンは、Resizeableプロパティが偽に設定された識別子"Window 001"を有する場合があり、これは、ＧＵＩスクリーンのサイズは実行時にユーザーによって変更することができないことを意味する。これらはほんの数例に過ぎず、いずれのユーザー・インターフェース・エレメントおよびユーザー・インターフェース・プロパティでも、所与の実施態様に対して所望通りに実現することができる。実施形態は、このコンテキストにおいて限定されることはない。

【００４８】

[0063] ウェブ・クライアント４３０は、１組の変更ユーザー・イベント・プロパティ４５１を、メッセージ４５０の中で、サーバー・アプリケーション４１０に送ることができる。サーバー４０２上で動作しているユーザー・インターフェース・マネージャー４１８は、このメッセージ４５０における変更ユーザー・イベント・プロパティ４５１を、処理のためにスクリプト・インタプリター４１４に転送する。サーバー・アプリケーション４１０は、サーバー・アプリケーション４１０に対するいずれのアプリケーション・ロジックを実行する前にでも、アプリケーション入力およびアプリケーション状態が適正であることを確認することができる。スクリプト・インタプリター４１４は、次いで、ファイル・マネージャー４１６と通信することができ、ファイル・マネージャー４１６は、クライアント４０４から受信したメッセージ４５０における変更ユーザー・イベント・プロパティ４５１に起因していずれかのアプリケーション規則の実行に必要であれば、データーベース４２０にアクセスする。しかるべきアプリケーション・ロジックの実行時に、インタプリター型ランタイム・エンジン４１２は、ＧＵＩ独立オブジェクト４５２を生成することができる。ＧＵＩ独立オブジェクト４５２は、情報の中でもとりわけ、更新ユーザー・イベント・プロパティ４５４を含むことができる。サーバー４０２によって実装されたユーザー・インターフェース・マネージャー４１８は、ＧＵＩ独立オブジェクト４５２を、あらゆる更新ユーザー・イベント・プロパティ４５４と共に、クライアント４０４に返送することができる。クライアント・アダプター４３２は、クライアント・ユーザー・イ

ンターフェース・マネージャー４３６およびレンダリング・エンジン４３８を通じて、次に、サーバー・アプリケーション４１０によって生成され、受信した更新ユーザー・イベント・プロパティ４５４と共にＧＵＩ独立オブジェクト４５２を用いて、以前にレンダリングした画像を更新することができる。

【００４９】

[0064] 以上で説明した実施形態の動作について、更に１つ以上の論理フローを参照しながら説明することができる。尚、代表的な論理フローは、特に指示がない場合は、必ずしも提示した順序で、またはいずれの特定の順序でも、実行しなくてもよいことは認められよう。更に、論理フローに関して説明する種々の動作(activities)は、シリアル様式またはパラレル様式で実行することができる。論理フローは、所与の１組の設計および性能の制約に対して望まれるように、以上で説明した実施形態の１つ以上のハードウェア・エレメントおよび／またはソフトウェア・エレメント、あるいは代わりのエレメントを用いて実現することができる。例えば、論理フローは、論理デバイス（例えば、汎用コンピューターまたは特殊目的コンピューター）による実行のためのロジック（例えば、コンピューター・プログラム命令）として実現することもできる。

【００５０】

[0065] 図５は、論理フロー５００の一実施形態を示す。論理フロー５００は、１つ以上の実施形態にしたがって実行される動作を示す。例えば、論理フロー５００は、ウェブ・クライアント４３０および／またはサーバー・アプリケーション４１０によって実行する動作を示すことができる。

【００５１】

[0066] 論理フロー５００では、ユーザーが、ブロック５０２において、クライアント側ユーザー・インターフェースにおいて実行しているウェブ・クライアントと対話処理する。例えば、ウェブ・クライアント４３０は、入力デバイスから受け取った１つ以上の制御指令の形態で、ユーザー入力を受けることができる。このユーザー入力は、レンダリング・エンジン４３８によって提示される、ユーザー・インターフェースの１つ以上のユーザー・インターフェース・エレメントに作用する。このユーザー入力は、ユーザー・インターフェース・エレメントと相互作用して、ユーザー・イベントを発生させる。例えば、ユーザーは、ＧＵＩスクリーン上にあるフィールドを選択し、このフィールドに対する値を変更することができる。

【００５２】

[0067] 論理フロー５００では、その中で実行するクライアント・アダプターは、ブロック５０４において、サーバー上で実行するサーバー・アプリケーションと調和するように、ユーザー・イベントを表す制御指令を解釈することができる。例えば、ウェブ・クライアント４３０によって実行されるクライアント・アダプター４３２は、サーバー・アプリケーション４１０と同様に、ユーザー・イベントを解釈することができる。ユーザー・イベントは、限定ではないが、ボタンをクリックする、テキスト・フィールドに記入する等というような、ウェブ・クライアント４３０上で実行しているユーザー・インターフェースとの１つ以上のユーザー対話処理を含むことができる。

【００５３】

[0068] 論理フロー５００では、ブロック５０４における解釈動作において、新たに入力されたユーザー・イベント・プロパティを検査して、このユーザー・イベント・プロパティが、サーバー・アプリケーションに通知することが必要な程に変化しているか否か、菱形５０６において判断する。例えば、クライアント・アダプター４３２は、ユーザー入力、および影響を受けたユーザー・インターフェース・エレメントのプロパティに対する対応する変更をいずれも調べて、ユーザー・イベント・プロパティがある閾値量を超えて変化したか否か判断することができる。例えば、あるフィールドの上方でホバリングしてこれに焦点を移すことは、ユーザー・イベント・プロパティに何らかの変更を誘起するには不十分であると考えられるが、あるフィールドを選択することは、サーバー・アプリケーション４１０に通知するのに十分であろう。

【００５４】
[0069] 論理フロー５００では、通知が必要とされる場合、クライアント・アダプターは、ブロック５０８において、変更保留のユーザー・イベント・プロパティをいずれもサーバー・アプリケーションに送ることができる。例えば、クライアント・アダプター４３２は、変更ユーザー・イベント・プロパティ４５１を、メッセージ４５０において、ネットワーク２５０を通じてサーバー・アプリケーション４１０に送ることができる。実施形態の中には、クライアント・アダプター４３２が多数のユーザー・イベントに対する多数の組の変更ユーザー・イベント・プロパティ４５１を、メッセージ４５０内において、サーバー４０２上で実行しているサーバー・アプリケーション４１０に送ることができる場合もある。この「バッチ」送付は、ユーザー・イベントのときにサーバー・アプリケーション４１０を補助することを含む、多くの場合に有用であることができる。例えば、スクリプト・インタプリター４１４は、サーバー・アプリケーション４１０に対する更新の正確なシーケンスを確保するために、種々のスクリプト（例えば、プリスクリプト、変更スクリプト、ポスト・スクリプト等）の実行時間を合わせることができる。また、バッチ送付は、ネットワーク２５０を通じて送るメッセージが少なくなることによって、通信オーバーヘッドも低減することができる。他の利点も存在し、実施形態はこのコンテキストにおいて限定されることはない。
【００５５】
[0070] 論理フロー５００では、サーバー上で実行しているランタイム・エンジンは、ブロック５１２において、ビジネス論理イベントを実行する前に、ブロック５１０においてサーバー・アプリケーションに対する適正な入力／状態を確保することができる。例えば、サーバー４０２上で実行するインタプリター型ランタイム・エンジン４１２は、アプリケーションまたはビジネス・ロジックを実行する前に常に、サーバー・アプリケーション４１０に対する適正なアプリケーション入力およびアプリケーション状態を確保することができる。
【００５６】
[0071] 論理フロー５００では、ブロック５１４においてビジネス・ロジックの実行によって生じた更新ユーザー・イベント・プロパティを、ＧＵＩ独立オブジェクトと共に、クライアント・アダプターに逆に転送することができる。例えば、アプリケーションまたはビジネス・ロジックの実行によって生じた更新ユーザー・イベント・プロパティ４５４を、クライアント・アダプター４３２に逆に転送するために、ＧＵＩ独立オブジェクト４５２と共に、サーバー・アプリケーション４１０からウェブ・クライアント４３０に送ることができる。
【００５７】
[0072] 論理フロー５００では、次に、クライアント・アダプターは、ブロック５１６において、更新ユーザー・イベント・プロパティおよびＧＵＩ独立オブジェクトを用いて、クライアント・ユーザー・インターフェースにおいて以前にレンダリングした画像を更新することができる。例えば、クライアント・アダプター４３２は、ＧＵＩ独立オブジェクト４５２を受信することができ、レンダリング・エンジン４３８は、更新ユーザー・イベント・プロパティ４５４およびＧＵＩ独立オブジェクト４５２を用いて、クライアント・ユーザー・インターフェースにおいて以前にレンダリングした画像を更新することができる。
【００５８】
[0073] 図６Ａは、サーバー・アプリケーション４１０からのデーターを用いて、クライアント・アダプター４３２のためにどのようにＧＵＩ独立オブジェクト４５２を作成できるかについての一実施形態を示す。既に説明したように、クライアント・アダプター４３２は、更新ユーザー・イベント・プロパティ４５４を有するＧＵＩ独立オブジェクト４５２を受信することができる。更新ユーザー・イベント・プロパティ４５４は、情報の中でもとりわけ、ＧＵＩ独立オブジェクト・メタデーター６０２を含むことができる。一実施形態では、ＧＵＩ独立オブジェクト・メタデーター６０２は、固定、即ち、静止メタデ

ーターを含むことができる。更に、更新ユーザー・イベント・プロパティ４５４は、プロパティ／値集合体６０４を含むことができる。固定／静止ＧＵＩ独立オブジェクト・メタデーター６０２を、ＧＵＩ独立プロパティ／値集合体６０４と組み合わせて、ＧＵＩ独立オブジェクト６０６を生成することができ、このＧＵＩ独立オブジェクト６０６は、クライアント・アダプター４３２によって、ウェブ・クライアント４３０においてレンダリングすることができる。

【００５９】

[0074] 図６Ｂは、図６Ａにおいて述べた構造(construct)を用いてどのようにして具体的なＧＵＩ独立オブジェクト４５２を作成することができるかについての一実施形態を示す。更新ユーザー・イベント・プロパティ４５４は、情報の中でもとりわけ、オブジェクト・メタデーター６１２、およびプロパティ／値集合体６１４を含むことができる。

【００６０】

[0075] 更新ユーザー・イベント・プロパティ４５４は、１つ以上のユーザー・インターフェース・エレメントを有するオブジェクト・メタデーター６１２を含むことができる。この例では、オブジェクト・メタデーター６１２は、フィールドＡ、フィールドＢ、およびフィールドＣと称する３つのユーザー・インターフェース・エレメントを、フィールドの形態で含む。フィールドＡ、Ｂ、およびＣの各々は、それぞれ、「フィールドＡ」、「フィールドＢ」、および「フィールドＣ」という句からなるデフォルト・フォント・テキストの周囲に境界があるテキスト・ボックスとして包括的に示されている。

【００６１】

[0076] 更に、更新ユーザー・イベント・プロパティ４５４は、プロパティ／値集合体６１４も含むことができる。一実施形態では、プロパティ／値集合体６１４は、１つ以上のタプル（または行）を有するテーブルのような、データー構造として実装することができ、各タプルが、ユーザー・インターフェース・エレメントの識別子、ユーザー・インターフェース・エレメントのプロパティ、およびプロパティの値を含む属性（または列）を構成する。識別子、プロパティ、および値のテーブルは、オブジェクト・メタデーター６１２のフィールドに対応することができる。

【００６２】

[0077] 一緒に組み合わせると、その結果はＧＵＩ独立オブジェクト６１６になることができる。ＧＵＩ独立オブジェクト６１６において示すように、フィールドＡは包括メタデーター・バージョンから変化していない。何故なら、そのプロパティや値はいずれも、プロパティ／値集合体６１４において変化させられなかったからである。フィールドＢは、その境界をなくして示されている。これは、プロパティ／値集合体６１４において、プロパティ「境界」が値「偽」に設定されたからである。フィールドＣにおけるテキストは太字で示されている。何故なら、プロパティ／値集合体６１４において、プロパティ「太字」が値「真」に設定されたからである。ここで、オブジェクト６１６は、クライアント・アダプター４３２のレンダリング・エンジン４３８によって、ウェブ・クライアント４３０においてクライアント４０４上にレンダリングすることができる。

【００６３】

[0078] 図７は、論理フロー７００の一実施形態を示す。論理フロー７００は、１つ以上の実施形態にしたがって実行される動作を示すことができる。例えば、論理フロー７００は、消去されたクライアント・アダプター４３２を復元する目的で、ウェブ・クライアント４３０および／またはサーバー・アプリケーション４１０によって実行される動作を示すことができる。

【００６４】

[0079] 本明細書において説明する実施形態の他の有益性は、クライアント・アダプター４３２が消去された場合、所与のクライアント４０４においてレンダリングされた画像を復元できることである。クライアント・アダプター４３２が消去された場合、種々のＧＵＩ依存オブジェクト４５２で構成されたレンダリング画像も消去される。しかしながら、サーバー・アプリケーション４１０はＧＵＩ独立オブジェクト４５２の形態で状態を保

持し続けることができる。図７に示すように、ブロック７０２において、ユーザーは、クライアント側ユーザー・インターフェースにおいて実行するウェブ・クライアント４３０と対話処理して、クライアント・アダプター４３２の新たなインスタンスを作成することができる。次いで、このクライアント・アダプター４３２の新たなスタンスは、ブロック７０４において、サーバー・アプリケーション４１０に再接続することができる。再接続のときに、サーバー・アプリケーション４１０は、全てのＧＵＩ独立オブジェクト４５２について最後に分かっていた状態をなおも維持することができる。ブロック７０６において、ＧＵＩ独立オブジェクト４５２について最後に分かっていた状態を、クライアント４０４に転送し、クライアント４０４がこれを受け取る。ＧＵＩ独立オブジェクト４５２について最後に分かっていた状態を、次に、ブロック７０８においてクライアント４０４のウェブ・クライアント４３０と同期させることができる。その結果、サーバー・アプリケーション４１０によって格納された情報を用いて、クライアント・アダプター４３２の現在の状態を効果的に復元することができる。

【００６５】

[0080] これまでの開示は、ＧＵＩ独立オブジェクト４５２を作成するために、どのようにレンダリング・エンジン４３８をインタプリター型ランタイム・エンジン４１２から切断すればよいかについて説明した。ＧＵＩ独立オブジェクト４５２は、とりわけ、Microsoft Windows（登録商標）のフォームまたはMicrosoft Silverlight（登録商標）のＵＩインターフェースのような、ユーザー・インターフェース・ビューとしてレンダリングすることができる。以下の説明では、ＧＵＩ独立オブジェクト４５２を、例えば、Microsoft Windows（登録商標）のフォームまたはMicrosoft Silverlight ＵＩというような、ユーザー・インターフェース・ビューのレンダリング画像に、どのように変換することができるかに照準を当てる。

【００６６】

[0081] 一実施形態によれば、システム４００は、インタプリター型ランタイム・エンジン４１２によって生成されたＧＵＩ独立オブジェクト・メタデーター６０２を、ユーザー・インターフェース・テンプレートに変換しつつ、元のメタデーター・コードをマスター・コード・ベースとして保持することができる。ＧＵＩ独立オブジェクト４５２は、レンダリング・メタデーターおよびイベント・メタデーターの双方を含む。既に説明したように、ＧＵＩ独立オブジェクト４５２は、インタプリター型ランタイム・エンジン４１２からまたはＧＵＩ独立オブジェクト４５２について詳細を提示することができるオブジェクト・モデルから直接生成することができる。

【００６７】

[0082] クライアント４０４上のユーザー・インターフェースをエンド・ユーザーに提示するタスクがレンダリング・エンジン４３８に課せられている間も、インタプリター型ランタイム・エンジン４１２の一部であるユーザー・インターフェース・マネージャー４１８には、引き続きＵＩイベントを処理するタスクが課せられていることを思い出されたい。インタプリター型ランタイム・エンジン４１２は、主にこれらのタスクを実行することから既に解放されているので、得られたＧＵＩ独立オブジェクト４５２は、インターフェースのいずれかを生成するために、クライアント・アダプター４３２によって処理される。これは、例えば、テンプレート・プロセッサーの実装によって遂行することができる。

【００６８】

[0083] 種々の実施形態では、テンプレート・プロセッサーは、ＧＵＩスクリーンの包括的表現を取り込み(take)、そのコンテンツ（例えば、フィールド、ボタン、およびイベント）の拡張可能マークアップ言語（ＸＭＬ）バージョンを適用することができる場合がある。このバージョンは、ＧＵＩスクリーン・テンプレートとして知られていることもある。加えて、テンプレート・プロセッサーは、ベース・テンプレートとして知られているバージョンにも適用することができる場合もある。インタプリター型ランタイム・エンジン４１２は、 ベース・テンプレートを古いクライアント(classical client)においてそ

のまま表示することができ、少なくともスクリーン用のＧＵＩテンプレートがない場合に基本的な変換を行う。ＧＵＩスクリーンおよびベース・テンプレートは、ＧＵＩスクリーン・レイアウトを表すメタデーターおよびコンテンツを含む。ＧＵＩスクリーン・テンプレートは、ベース・テンプレートに関係するが、そのカスタム化したバージョンである。

【００６９】

[0084] テンプレートは、既存のＧＵＩ独立オブジェクトのレイアウトを変更するように設計することができる。本明細書において既に紹介した例では、２つのタイプのテンプレート、即ち、ベース・テンプレートおよびＧＵＩスクリーン・テンプレートがある。尚、種々の実施態様では必要に応じて異なるテンプレートを用いてもよことは認めることができよう。

【００７０】

[0085] 第１テンプレートは、ベース・テンプレートと呼ぶことができる。多くのアプリケーションに対して、文字通り数千ものＧＵＩスクリーンがあると考えられ、各々に対してテンプレート（即ち、新たなレイアウト）を開発し適用するには多大な時間と労力がかかるであろう。しかしながら、ベース・テンプレートは、ＧＵＩスクリーン毎に新たなテンプレート・レイアウトを作成する必要なく、基本変換を適用することができる。どちらかと言えば、１つの新しいレイアウトＧＵＩスクリーンを作成するために、従来のロジックを適用することができる。ベース・テンプレートは、１つのＧＵＩスクリーンを出力することができ、通例、多数のＧＵＩスクリーンを１つに組み合わせるように設計されていない。

【００７１】

[0086] 第２テンプレートは、ＧＵＩスクリーン・テンプレートと呼ぶことができる。ＧＵＩスクリーン・テンプレートは、所与のＧＵＩスクリーンのテンプレートを構成することができる。ＧＵＩスクリーン・テンプレート・レイアウトは、ヘッダおよびコンテンツ・セクション双方のベース・テンプレート・レイアウトをオーバーライドすることができる。オーバーライドされるコンテンツ・テンプレートは、格子レイアウトのようなテーブル状であってもよく、および／またはアコーディオン・レイアウトのように纏めることができる。新たなレイアウトはＸＭＬファイルにおいて定義することができる。ＧＵＩスクリーン・テンプレート・レイアウトは、多数のＧＵＩスクリーンを１つに組み合わせることができ、それが変化させているＧＵＩスクリーンに特定的であることができる。

【００７２】

[0087] 本明細書において用いる場合、「ＧＵＩスクリーン」という用語は、ディスプレイの提示フィールドまたは表示エリアの一部または全部を消費するように構成されたユーザー・インターフェース・エレメントを指すことができる。例えば、あるユーザー・インターフェース・エレメントは、ディスプレイ上で境界、ボックス、または他の枠状ユーザー・インタフェース・エレメントによって輪郭が定められるディスプレイの一部またはサブセクションだけを消費するように設計される。場合によっては、ＧＵＩスクリーンが１組のＵＩ制御部を有し、ユーザーがディスプレイの提示フィールド周囲においてＧＵＩスクリーンを拡大、縮小、または移動させること、あるいはディスプレイの提示フィールドからＧＵＩスクリーンを完全に削除することを可能にすることもできる。ＧＵＩスクリーンの例には、アプリケーションおよびオペレーティング・システムの中でもとりわけ、例えば、Microsoft WindowsまたはMicrosoft Windows Formユーザー・インターフェース・アプリケーションによって生成されるＧＵＩ「ウィンドウ」のような、ユーザー・インターフェース・エレメントを含むことができる。

【００７３】

[0088] テンプレート・プロセッサーは、以上で説明したテンプレートを、とりわけ、ＧＵＩスクリーン・レイアウトについての詳細を含むＧＵＩ独立オブジェクト４５２のカスタム化バージョンを生成するために適用することができる。レンダリング・エンジン４３８は、ＵＩ変換ロジックと共に、ＧＵＩ独立オブジェクト４５２の新たなカスタム化バージョンを受け取り、エンド・ユーザーに提示するためにクライアント４０４に対して新

たなカスタム化ＧＵＩビュー（例えば、ＧＵＩウィンドウ）を生成することができる。レンダリング・エンジン４３８は、ＧＵＩオブジェクト属性をＧＵＩ制御部およびプロパティとマッピングし（例えば、ヘッダからリボンに）、クライアント４０４が望む特定のＧＵＩ制御部およびレイアウトを生成することができる。

【００７４】

[0089] 図８Ａは、一実施形態にしたがって、ＧＵＩオブジェクト・レイアウトを表すテンプレートを処理するテンプレート処理システム８００のブロック図を示す。一実施形態では、テンプレート処理システム８００は、サーバー８０２のサーバー・アプリケーション８１０の一部として実装することができる。サーバー・アプリケーション８１０およびサーバー８０２は、例えば、図４を参照して説明したような、それぞれのサーバー・アプリケーション４１０およびサーバー４０２を表すことができる。しかしながら、テンプレート処理システム８００は、例えば、クライアント８０４のクライアント・アプリケーション８３０を含む、ｎ－層クライアント－サーバー・アーキテクチャーの種々の他の部分に実装することもできる。

【００７５】

[0090] 一実施形態では、クライアント・アプリケーション８３０は、例えば、図４を参照して説明した、ウェブ・クライアント４３０および／またはクライアント４０４のクライアント・アダプター４３２を表すことができる。加えてまたは代わりに、クライアント・アプリケーション８３０は、例えば、サーバー・アプリケーション８１０のネーティブ・バージョンまたはデスクトップ・バージョンというような、クライアント・アプリケーション４３０とは異なるクライアント・アプリケーションとして実装することもできる。他のクライアント・アプリケーションも同様に実装することができる。実施形態はこのコンテキストにおいて限定されることはない。

【００７６】

[0091] 図４を参照して既に説明したように、変更ユーザー・イベント・プロパティ４５１を含むメッセージ４５０をサーバー・アプリケーション４３０に送ることができる。すると、サーバー・アプリケーション４１０のインタプリター型ランタイム・エンジン４１２は、変更ユーザー・イベント・プロパティ４５１を処理してＧＵＩ独立オブジェクト４５２を生成する。

【００７７】

[0092] 図８Ａに示す例示実施形態では、ユーザー・イベント８０４によって同様のプロセスを表すことができ、クライアント８０４のクライアント・アプリケーション８３０が、インタプリター型ランタイム・エンジン８５０にユーザー・インターフェース・マネージャー８０６を通じてユーザー・イベント８０４を転送し、ＧＵＩ独立オブジェクト８１２を生成することができる。しかしながら、この時点において、ＧＵＩ独立オブジェクト８１２は、暫定的なＧＵＩ独立オブジェクトであり、クライアント・アプリケーション８３０による提示の準備はできていない。特定のクライアント・アプリケーション８３０による使用に合わせてＧＵＩ独立オブジェクト８１２を精細化するために、ＧＵＩ独立オブジェクト８１２を、更に処理するために、テンプレート・プロセッサー８１４に転送することができる。すると、テンプレート・プロセッサー８１４は、ＧＵＩ独立オブジェクト８１２からのベース・テンプレート８１６およびＧＵＩスクリーン・テンプレート８１８を適用することができる。

【００７８】

[0093] ベース・テンプレート８１６は、ＧＵＩスクリーン毎に新たなテンプレート・レイアウトを作成する必要がない基準変換を適用することができる。ベース・テンプレート８１６は変換ロジックを適用して、ＧＵＩビュー８１１として示される１つの新たなレイアウトを作成することができる。この実施形態では、ＧＵＩビュー８１１は、多数のＧＵＩスクリーンを１つに組み合わせるようには設計されていない。

【００７９】

[0094] ＧＵＩスクリーン・テンプレート・レイアウトは、ヘッダおよびコンテンツ・

セクション双方のベース・テンプレート８１６のベース・テンプレート・レイアウトをオーバーライドすることができる。オーバーライドされるコンテンツ・テンプレートは、格子レイアウトのようなテーブル状であってもよく、および／またはアコーディオン・レイアウトのように纏めることができる。一実施形態では、新たなレイアウトは、適したウェブ関係レイアウト・フォーマットでもとりわけ、ＸＭＬファイルにおいて定義することができる。ＧＵＩスクリーン・テンプレート・レイアウトは、多数のＧＵＩスクリーンを１つに組み合わせることができ、それが変化させているＧＵＩスクリーンに特定的であることができる。

【００８０】

[0095] テンプレート・プロセッサー８１４は、次いで、以上で説明したテンプレート８１６、８１８を適用して、新たなそして高度にカスタム化したＧＵＩ独立オブジェクト８２０を生成することができる。このＧＵＩ独立オブジェクト８２０は、ＧＵＩスクリーン・レイアウトの詳細を含む。新たなＧＵＩ独立オブジェクト８２０は、ＧＵＩ独立オブジェクト８１２について更に具体的な実施態様を含むことができる。一方、ＧＵＩ独立オブジェクト８１２は、図３および図４をそれぞれ参照して説明したＧＵＩ独立オブジェクト３６０、４５２を表す。新たなＧＵＩ独立オブジェクト８２０は、サーバー・アプリケーション８１０から、クライアント８０４上で実行するクライアント・アプリケーション８３０のレンダリング・エンジン８２２に戻される。

【００８１】

[0096] レンダリング・エンジン８２２は、新たなＧＵＩ独立オブジェクト８２０を受け取り、エンド・ユーザーにカスタム化した新たなＧＵＩビュー８２４を生成するように設計されているＵＩ変換ロジックを実行する。レンダリング・エンジン８２２は、ＧＵＩオブジェクト属性をＧＵＩ制御部およびプロパティとマッピングし（例えば、ヘッダからリボンへというような）、クライアント・アプリケーションが望む具体的なＧＵＩ制御部およびレイアウトを生成することができる。

【００８２】

[0097] 図８Ｂおよび図８Ｃは、それぞれ、ＧＵＩビュー８１１、８２４の更に詳しい図を示す。例えば、ベース・テンプレートを表すＧＵＩスクリーンを図８ＢにおけるＧＵＩビュー８１１に示すことができ、ＧＵＩスクリーン・テンプレートから得られるＧＵＩスクリーンを図８Ｃにおけるビュー８２４に示すことができる。

【００８３】

[0098] 図８Ｂおよび図８Ｃに示す例示実施形態では、ベース・テンプレート８１６から組み立てられたＧＵＩビュー８１１（図８Ｂ）は、スクリーン・テンプレート８１８から組み立てられたＧＵＩビュー８２４（図８Ｃ）と同様に見えるが、それでも異なっている。スクリーン・テンプレート８１８はベース・テンプレート８１６に取って代わることを思い出されたい。即ち、ベース・テンプレート８１６を最初に適用することができ、続いてスクリーン・テンプレート８１８を適用して、ベース・テンプレート８１６をカスタム化することができる。図８Ｂ（ベース・テンプレートの表現）および図８Ｃ（スクリーン・テンプレートの表現）に図示されている例では、ボタンおよびフィールド・ボックスの多くは構成し直されている。例えば、ベース・テンプレート８１６から組み立てられたＧＵＩビュー８１１における左下のボタンは、スクリーン・テンプレート８１８から組み立てられたＧＵＩビュー８２４の左上部分にあるメニュー・バーに配置し直されている。８１１（図８Ｂ）および８２４（図８Ｃ）のＧＵＩビューは双方共、リボンＵＩ表現８２６を示す。リボンとは、機能毎に編成されるメニューおよびボタンのグループを収容する大きなツールバーのことである。各リボンは、機能的にタブと関連付けられる。図８ＣのＧＵＩビュー８２４を参照すると、タブは"Vendor"（販売業者）となっている。更に、このvendorタブのレイアウトは、２つのセクションで構成されている。セクション１は、「一般」８２８という名称が付けられており、一方セクション２は「アドレス」８６０という名称が付けられている。これは、図８ＢにおけるＧＵＩビュー８１１とは、ＧＵＩビュー８２４におけるタブ（Address（アドレス）、Accounts（アカウント）、Options（選択

肢）、Email（電子メール）、Withholding（保留）)が、ＧＵＩビュー８１１におけるボタン（Options（選択肢）、Address（アドレス）、Accounts（アカウント）、E-mail（電子メール））と置き換えられていることが異なっている。尚、ＧＵＩビュー８１１、８２４間では他の変更および改変(alteration)が既に行われていること、および行うことができることは認められよう。このように、テンプレート処理システム８００は、異なるクライアント・アダプター（例えば、クライアント・アダプター２３２、２４２）を実現する異なるウェブ・クライアント（例えば、ウェブ・クライアント２３０、２４０）に合わせてカスタム化したＧＵＩビューを提供することができ、カスタム化ＧＵＩビューは、サーバー・アプリケーション（例えば、４１０、８１０）によって供給されるネーティブなＧＵＩビューから得られる。

【００８４】

[0099] 図９は、論理フロー９００を示す。論理フロー９００は、１つ以上の実施形態によって実行される動作を示すことができる。例えば、論理フロー９００は、図８に示すような、サーバー・アプリケーション８１０および／またはクライアント・アプリケーション８３０によって実行される動作を示すことができる。加えてまたは代わりに、論理フロー９００は、図４に示すような、サーバー・アプリケーション４１０および／またはウェブ・クライアント４３０によって実行される動作を示すこともできる。実施形態は、このコンテキストにおいて限定されることはない。

【００８５】

[00100] 論理フロー９００では、ユーザーは、クライアント側ユーザー・インターフェースにおいて実行しているウェブ・クライアントと対話処理して、ユーザー・イベントを入力する。ブロック９０２、９０４、および９０６は、図５において更に詳しく説明した論理プロセスの省略表現を表すことができる。

【００８６】

[00101] 論理フロー９００では、ブロック９０６においてランタイム・エンジンによってＧＵＩ独立オブジェクトが生成されると、ブロック９０８においてこのＧＵＩ独立オブジェクトをテンプレート・プロセッサーに転送することができる。例えば、ＧＵＩ独立オブジェクト８１２を、クライアント・アプリケーション８３０によって、サーバー・アプリケーション８１０内において実行するテンプレート・プロセッサー８１４に、更に処理するために転送することができる。

【００８７】

[00102] 論理フロー９００では、テンプレート・プロセッサーは、ブロック９１２において、生成したベース・テンプレートおよびスクリーン・テンプレートを適用して、新たなＧＵＩオブジェクトを生成することができる。例えば、テンプレート・プロセッサー８１４は、生成したベース・テンプレート８１６およびスクリーン・テンプレート８１８を適用して、カスタム化した新たなＧＵＩ独立オブジェクト８２０を生成することができる。

【００８８】

[00103] 論理フロー９００では、ブロック９１４において、新たなＧＵＩ独立オブジェクトをクライアントに返送することができる。例えば、カスタム化した新たなＧＵＩ独立オブジェクト８２０を、クライアント・アプリケーション８３０に（ネットワーク２５０を通じて）返送することができ、ここから、更に処理するために、レンダリング・エンジン８２２に転送することができる。

【００８９】

[00104] 論理フロー９００では、ブロック９１６においてレンダリング・エンジンはＧＵＩオブジェクトを変換することができ、新たなＧＵＩスクリーンを生成することができる。例えば、レンダリング・エンジン８２２はＧＵＩ独立オブジェクト８２０を変換することができ、カスタム化ＧＵＩビュー８２４を生成することができる。カスタム化ＧＵＩビュー８２４は、クライアント・アプリケーション８３０（あるいはクライアント・アダプターまたはクライアントＯＳ）によってレンダリングされる。

【００９０】
[00105] 図１０は、既に説明したような種々の実施形態を実現するのに適した計算アーキテクチャー例１０００の一実施形態を示す。計算アーキテクチャー１０００は、１つ以上のプロセッサー、コプロセッサー、メモリー・ユニット、チップセット、コントローラ、周辺機器、インターフェース、発振器、タイミング・デバイス、ビデオ・カード、オーディオ・カード、マルチメディア入力／出力（Ｉ／Ｏ）コンポーネント等というような、種々の一般的な計算エレメントを含む。しかしながら、実施形態は、計算アーキテクチャー１０００による実施態様に限定されるのではない。
【００９１】
[00106] 図１０に示すように、計算アーキテクチャー１０００は、演算装置１００４、システム・メモリー１００６、およびシステム・バス１００８を備えている。演算装置１００４は、種々の市販されているプロセッサーのいずれでも可能である。デュアル・マイクロプロセッサーおよびその他のマルチプロセッサー・アーキテクチャーも、演算装置１００４として採用することができる。システム・バス１００８は、システム・メモリー１００６から演算装置１００４までを含むがこれらには限定されないシステム・コンポーネントにインターフェースを設ける。システム・バス１０１０は、様々なタイプのバス構造のいずれでも可能であり、そのバス構造は、更に、メモリー・バス（メモリー・コントローラを有するまたは有さない）、周辺バス、および種々の市販されているバス・アーキテクチャーのいずれかを用いるローカル・バスに相互接続することもできる。
【００９２】
[00107] システム・メモリー１００６は、リード・オンリー・メモリー（ＲＯＭ）、ランダム・アクセス・メモリー（ＲＡＭ）、ダイナミックＲＡＭ（ＤＲＡＭ）、倍速データーＤＲＡＭ（ＤＤＲＡＭ）、同期ＤＲＡＭ（ＳＤＲＡＭ）、スタティックＲＡＭ（ＳＲＡＭ）、プログラマブルＲＡＭ（ＰＲＯＭ）、消去可能プログラマブルＲＯＭ（ＥＰＲＯＭ）、電気的消去可能プログラマブルＲＯＭ（ＥＥＰＲＯＭ）、フラッシュ・メモリー、強誘電体ポリマー・メモリーのようなポリマー・メモリー、オーボニック・メモリー、位相変化または強誘電体メモリー、シリコン－酸化物－窒化物－酸化物－シリコン（ＳＯＮＯＳ）メモリー、磁気または光カード、あるいは情報を格納するのに適した他のあらゆるタイプの媒体というような、種々のタイプのメモリー・ユニットを含むことができる。図１０に示す例示実施形態では、システム・メモリー１００６は、不揮発性メモリー１０１０および／または揮発性メモリー１０１２を含むことができる。基本入力／出力システム（ＢＩＯＳ）は、不揮発性メモリー１０１０に格納することができる。
【００９３】
[00108] コンピューター１００２は、種々のタイプのコンピューター読み取り可能記憶媒体を含むことができ、内部ハード・ディスク・ドライブ（ＨＤＤ）１０１４、リムーバブル磁気ディスク１０１１０に対する読み取りおよび書き込みを行う磁気フロッピー（登録商標）・ディスク・ドライブ（ＦＤＤ）１０１６、ならびにリムーバブル光ディスク１０２２（例えば、ＣＤ－ＲＯＭまたはＤＶＤ）に対する読み取りおよび書き込みを行う光ディスク・ドライブ１０２０を含む。ＨＤＤ１０１４、ＦＤＤ１０１６、および光ディスク・ドライブ１０２０は、それぞれ、ＨＤＤインターフェース１０２４、ＦＤＤインターフェース１０２６、および光ドライブ・インターフェース１０２１０によって、システム・バス１００１０８に接続することができる。外部ドライブ実装のためのＨＤＤインターフェース１０２４は、ユニバーサル・シリアル・バス（ＵＳＢ）およびIEEE1394インターフェース技術の内少なくとも１つまたは両方を含むことができる。
【００９４】
[00109] ドライブおよび付随するコンピューター読み取り可能媒体は、データー、データー構造、コンピューター実行可能命令等の揮発性および／または不揮発性格納機能を設ける。例えば、多数のプログラム・モジュールをドライブおよびメモリー・ユニット１０１０、１０１２に格納することができる。プログラム・モジュールには、オペレーティング・システム１０３０、１つ以上のアプリケーション・プログラム１０３２、他のプロ

グラム・モジュール１０３４、およびプログラム・データー１０３６が含まれる。１つ以上のアプリケーション・プログラム１０３２、他のプログラム・モジュール１０３４、およびプログラム・データー１０３６は、例えば、クライアント－サーバー・システム２００、３００、および４００のソフトウェア・コンポーネントを含むことができる。

【００９５】

[00110] ユーザーは、１つ以上の有線／ワイヤレス入力デバイス、例えば、キーボード１０３１０およびマウス１０４０のようなポインティング・デバイスによって、コンピューター１００２にコマンドおよび情報を入力することができる。他の入力デバイスは、マイクロフォン、赤外線（ＩＲ）リモコン、ジョイスティック、ゲーム・パッド、スタイラス・ペン、タッチ・スクリーン等を含むことができる。これらのおよびその他の入力デバイスは、多くの場合、システム・バス１００１０に結合されている入力デバイス・インターフェース１０４２を通じて演算装置１００４に接続されるが、パラレル・ポート、IEEE1394シリアル・ポート、ゲーム・ポート、ＵＳＢポート、ＩＲインターフェース等のような、他のインターフェースによって接続することもできる。

【００９６】

[00111] １つ以上のモニター１０４４または他のタイプのディスプレイ・デバイスも、ビデオ・アダプター１０４６のようなインターフェースを通じて、システム・バス１００８に接続されている。モニター１０４４に加えて、コンピューターは、通例、スピーカー、プリンター等のような、他の周辺出力デバイスも含む。また、１つ以上のモニター１０４５は、入力デバイス・インターフェース１０４２および／またはＵＳＢハブ１０４３のようなハブを通じてシステム・バス１００１０にも接続することができる。モニター１０４５は、ビデオ・カメラ、アレイ・マイクロフォン、タッチ・センサー、動きセンサー、スピーカー等というような、種々のコンポーネントを備えることもできる。これらのコンポーネントは、ＵＳＢハブ１０４３を通じて入力デバイス・インターフェース１０４２に接続することができる。

【００９７】

[00112] コンピューター１００２は、論理接続を用いるネットワーク接続環境において、リモート・コンピューター１０４１０のような１つ以上のリモート・コンピューターへの有線通信および／またはワイヤレス通信によって動作することができる。リモート・コンピューター１０４１０は、ワークステーション、サーバー・コンピューター、ルータ、パーソナル・コンピューター、携帯用コンピューター、マイクロプロセッサー・ベースの娯楽用機器、ピア・デバイス、または他の一般的なネットワーク・ノードとすることができ、通例、コンピューター１００２に関して説明したエレメントの多くまたは全てを含む。しかし、簡潔さという目的のために、メモリー／記憶デバイス１０５０のみが示されている。図示されている論理接続は、ローカル・エリア・ネットワーク（ＬＡＮ）１０５２および／またはそれよりも大きいネットワーク、例えば、ワイド・エリア・ネットワーク（ＷＡＮ）１０５４への有線／ワイヤレス接続(connectivity)を含む。このようなＬＡＮおよびＷＡＮネットワーク接続環境は、事務所や会社では極普通であり、イントラネットのような企業規模のコンピューター・ネットワークを設置し易くする。これらのネットワークの全ては、地球規模の通信ネットワーク、例えば、インターネットに接続することもできる。

【００９８】

[00113] ＬＡＮネットワーク接続環境において用いる場合、コンピューター１００２は、有線および／またはワイヤレス通信ネットワーク・インターフェースまたはアダプター１０５６を介してＬＡＮ１０５２に接続される。アダプター１０５６は、ＬＡＮ１０５２への有線および／またはワイヤレス通信をし易くすることができ、アダプター１０５６のワイヤレス機能と通信するために、そこに配置されるワイヤレス・アクセス・ポイントを含むこともできる。

【００９９】

[00114] ＷＡＮネットワーク接続環境において用いる場合、コンピューター１００２

はモデム１０５１０を含むことができ、あるいはＷＡＮ１０５４上の通信サーバーに接続されるか、または一例としてインターネットのようなＷＡＮ１０５４を通じて通信を確立する他の手段を有する。モデム１０５１０は、内蔵型でも外付けでも可能であり、更には有線および／またはワイヤレス・デバイスも可能であり、入力デバイス・インターフェース１０４２を通じてシステム・バス１００１０に接続する。ネットワーク接続環境では、コンピューター１００２に関して図示したプログラム・モジュール、またはその一部を、リモート・メモリー／ストレージ・デバイス１０５０に格納することができる。尚、図示するネットワーク接続は一例であり、コンピューター間で通信リンクを確立する他の手段も用いることができることは認められよう。

【０１００】

[00115] コンピューター１００２は、例えば、プリンター、スキャナー、デスクトップおよび／または携帯用コンピューター、パーソナル・ディジタル・アシスタント（ＰＤＡ）、通信衛星、ワイヤレスで検出可能なタグ（例えば、キオスク、売店、休憩室）に付随するあらゆる機器または位置(location)、ならびに電話機とワイヤレス通信可能に（例えば、IEEE802.11空中変調技法）動作的に配置されているワイヤレス・デバイスというような、IEEE802系の標準規格を用いる有線およびワイヤレス・デバイスまたはエンティティと通信するように動作することができる。これは、少なくとも、Ｗｉ－Ｆｉ（即ち、ワイヤレス・フィデリティ）、ＷｉＭａｘ、およびBluetooth（登録商標）ワイヤレス技術を含む。このように、通信は、従来のネットワークと同様に、既定の構造とすることができ、または単に少なくとも２つのデバイス間におけるアドホック通信とすることもできる。Ｗｉ－Ｆｉネットワークは、IEEE802.11x（a、b、g等）を用いて、安全で信頼性があり高速のワイヤレス接続を提供する。Ｗｉ－Ｆｉネットワークは、コンピューターを互いに接続するため、インターネットに接続するため、そして有線ネットワーク（IEEE802.3に関係する媒体および機能を用いる）に接続するために用いることができる。

【０１０１】

[00116] 図１１は、既に説明した種々の実施形態を実現するのに適した通信アーキテクチャー例１１００のブロック図を示す。通信アーキテクチャー１１００は、送信機、受信機、送受信機、無線機、ネットワーク・インターフェース、ベースバンド・プロセッサー、アンテナ、増幅器、フィルター等というような、種々の一般的な通信エレメントを含む。しかしながら、実施形態は、通信アーキテクチャー１１００による実施態様に限定されるのではない。

【０１０２】

[00117] 図１１に示すように、通信アーキテクチャー１１００は、１つ以上のクライアント１１０２およびサーバー１１０４を備えている。クライアント１１０２は、ウェブ・クライアント４３０を実装することができる。サーバー１１０４は、インタプリター型ランタイム・エンジン４１２を実装することができる。クライアント１１０２およびサーバー１１０４は、１つ以上のそれぞれのクライアント・データー・ストア１１０８およびサーバー・データー・ストア１１１０に動作的に接続されている。データー・ストア１１０８、１１１０は、クッキーおよび／または関連するコンテキスト情報というような、それぞれのクライアント１１０２およびサーバー１１０４にローカルな情報を格納するために用いることができる。

【０１０３】

[00118] クライアント１１０２およびサーバー１１０４は、通信フレームワーク１１０６を用いて、互いの間で情報を伝達し合うことができる。通信フレームワーク１１０６は、パケット交換ネットワーク（例えば、インターネットのような公開ネットワーク、企業のイントラネットのような私有ネットワーク等）、回線交換ネットワーク（例えば、公衆電話交換ネットワーク）、またはパケット交換ネットワークおよび回線交換ネットワークの組み合わせ（適したゲートウェイおよびトランスレーターを用いる）というような、周知の通信技法のいずれでも実現することができる。クライアント１１０２およびサーバー１１０４は、１つ以上の通信インターフェース、ネットワーク・インターフェース、ネ

ットワーク・インターフェース・カード（ＮＩＣ）、無線機、ワイヤレス送信機／受信機（送受信機）、有線および／またはワイヤレス通信媒体、物理コネクター等というような、通信フレームワーク１１０６と相互動作可能であるように設計された種々のタイプの標準的な通信エレメントを含むことができる。一例として、そして限定ではなく、通信媒体は、有線通信媒体およびワイヤレス通信媒体を含む。有線通信媒体の例は、ワイヤ、ケーブル、金属線、印刷回路ボード（ＰＣＢ）、バックプレーン、スイッチ・ファブリック、半導体材料、撚り線対ワイヤ、同軸ケーブル、光ファイバー、伝搬信号等を含むことができる。ワイヤレス通信媒体の例は、音響、無線周波（ＲＦ）スペクトル、赤外線、およびその他のワイヤレス媒体を含むことができる。クライアント１１０２とサーバー１１０４との間において可能な１つの通信は、２つ以上のコンピューター・プロセス間で送信されるように構成されたデーター・パケットの形態とすることができる。データー・パケットは、例えば、クッキーおよび／または関連するコンテキスト情報を含むことができる。

【０１０４】

[00119] 種々の実施形態は、ハードウェア・エレメント、ソフトウェア・エレメント、または双方の組み合わせを用いて実現することができる。ハードウェア・エレメントの例には、デバイス、論理デバイス、コンポーネント、プロセッサー、マイクロプロセッサー、回路、回路エレメント（例えば、トランジスター、抵抗器、キャパシター、インダクター等）、集積回路、特定用途集積回路（ＡＳＩＣ）、プログラマブル論理デバイス（ＰＬＤ）、ディジタル信号プロセッサー（ＤＳＰ）、フィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ）、メモリー・ユニット、論理ゲート、レジスター、半導体デバイス、チップ、マイクロチップ、チップセット等を含むことができる。ソフトウェア・エレメントの例には、ソフトウェア・コンポーネント、プログラム、アプリケーション、コンピューター・プログラム、アプリケーション・プログラム、システム・プログラム、機械プログラム、オペレーティング・システム・ソフトウェア、ミドルウェア、ファームウェア、ソフトウェア・モジュール、ルーチン、サブルーチン、関数、メソッド、手順、ソフトウェア・インターフェース、アプリケーション・プログラム・インターフェース（ＡＰＩ）、命令セット、計算コード、コンピューター・コード、コード・セグメント、コンピューター・コード・セグメント、ワード(word)、値、記号、またはこれらのあらゆる組み合わせを含むことができる。ハードウェア・エレメントおよび／またはソフトウェア・エレメントのどちらを用いて実施形態を実現するか決定するのは、所望の計算率、電力レベル、熱許容度、処理サイクルの予算、入力データー・レート、出力データー・レート、メモリー・リソース、データー・バス速度、および、所与の実施態様に望まれる通りの他の設計または性能制約というような、 いかなる数の要因にしたがって多様に変化するのでもよい。

【０１０５】

[00120] 実施形態の中には、製造品目(article of manufacture)を構成するものもある。製造品目は、ロジックを格納するように構成されたコンピューター読み取り可能記憶媒体を含むことができる。コンピューター読み取り可能記憶媒体の例には、電子データーを格納することができるあらゆる記憶媒体が含まれ、揮発性メモリーまたは不揮発性メモリー、リムーバブルまたは非リムーバブル・メモリー、消去可能メモリーまたは消去可能でないメモリー、書き込み可能メモリーまたは再書き込み可能メモリー等が含まれる。ロジックの例には、ソフトウェア・コンポーネント、プログラム、アプリケーション、コンピューター・プログラム、アプリケーション・プログラム、システム・プログラム、機械プログラム、オペレーティング・システム・ソフトウェア、ミドルウェア、ファームウェア、ソフトウェア・モジュール、ルーチン、サブルーチン、関数、メソッド、手順、ソフトウェア・インターフェース、アプリケーション・プログラム・インターフェース（ＡＰＩ）、命令セット、計算コード、コンピューター／コード、コード・セグメント、コンピューター・コード・セグメント、ワード、値、記号、またはこれらのあらゆる組み合わせというような、種々のソフトウェア・エレメントを含むことができる。一実施形態では、例えば、製造品目は、実行可能コンピューター・プログラム命令を格納するのでもよく、

この命令をコンピューターによって実行すると、以上で説明した実施形態にしたがって方法および／または動作をこのコンピューターに実行させる。実行可能コンピューター・プログラム命令は、ソース・コード、コンパイル・コード、インタプリター・コード、実行可能コード、スタティック・コード、ダイナミック・コード等のような、適したタイプのコードであればいずれでも含むことができる。実行可能コンピューター・プログラム命令は、既定のコンピューター言語、様式(manner)または構文(syntax)にしたがって、一定の機能を実行するようにコンピューターに命令するために、実装することができる。 命令は、適した高級プログラム言語、低級プログラム言語、オブジェクト指向プログラム言語、ビジュアル・プログラム言語、コンパイル型プログラム言語および／またはインタプリター型プログラム言語であればいずれを用いても実装することができる。

【０１０６】

[00121] 実施形態の中には、「一実施形態」または「実施形態」という表現をその派生語と共に用いて説明するとよい場合がある。これらの用語は、当該実施形態と関連付けて説明された特定の特徴、構造、または特性が、少なくとも１つの実施形態に含まれることを意味する。本明細書の種々の場所において「一実施形態では」という句が出てくる場合、必ずしも全てが同じ実施形態を指す訳ではない。

【０１０７】

[00122] 実施形態の中には、「結合される」(coupled)および「接続される」(connected)という表現を、その派生語と共に用いて説明するとよい場合がある。これらの用語は、必ずしも互いに対する同義語であることを意図しているのではない。例えば、実施形態の中には、２つ以上のエレメントが直接物理的にまたは電気的に互いに接触していることを示すために、「接続される」および／または「結合される」という用語を用いて説明するとよい場合がある。しかしながら、「結合される」という用語は、２つ以上のエレメントが互いに直接接触していないが、それでも互いに協働するまたは相互作用することを意味することもできる。

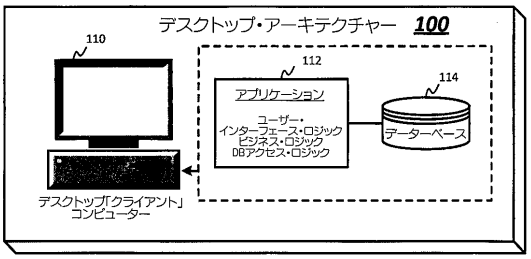
【０１０８】

[00123] 開示の要約は、読み手が本技術的開示の固有性を素早く確認することを可能にするために設けられていることを強調しておく。尚、これは、請求項の範囲または意味を解釈するためや限定するために用いられるのではないことを前提として、申し述べることとする。加えて、以上の詳細な説明では、開示を簡素化する目的に限って、１つの実施形態において種々の特徴が一緒に纏められていることが分かるであろう。この開示方法は、特許請求する実施形態が各請求項において明示的に記載される特徴よりも多くの特徴を必要とするという意図を表す(reflect) というように解釈してはならない。逆に、以下の請求項が表すように、発明の主題は、１つの開示された実施形態の全ての特徴に存在する訳ではない。つまり、以下の請求項は、詳細な説明に含まれることとし、各請求項が別個の実施形態としてそれ自体を成り立たせている(stand on its own)。添付した特許請求の範囲において、「含む」(including)および「において」(in which)という用語は、それぞれ、「備えている」(comprising)および「において」(wherein)というそれぞれの用語の平素な英語の同義語(equivalent)として用いられるものとする。更に、「第１」、「第２」、「第３」等は、単に名称として用いられるのであり、それらの目的語に対して数値的な要件を強制することは意図していない。

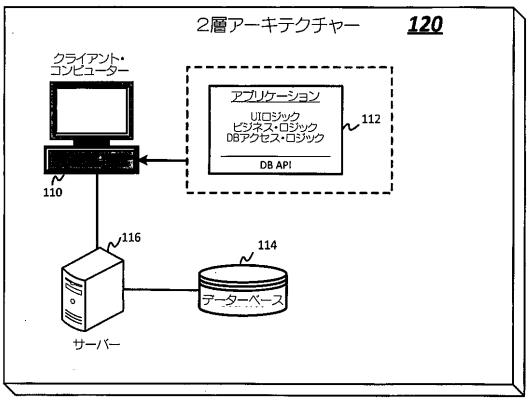
【０１０９】

[00124] 以上、構造的特徴および／または方法論的動作に特定的な文言で本主題について説明したが、添付した特許請求の範囲において定義されている主題は、必ずしも以上で説明した具体的な特徴や動作には限定されないことは理解されてしかるべきである。逆に、以上で説明した具体的な特徴や動作は、特許請求の範囲を実現する形態例として開示したまでである。

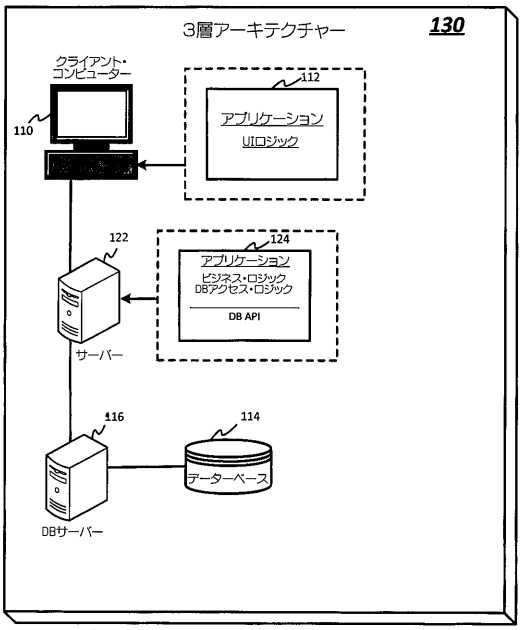
【図１Ａ】



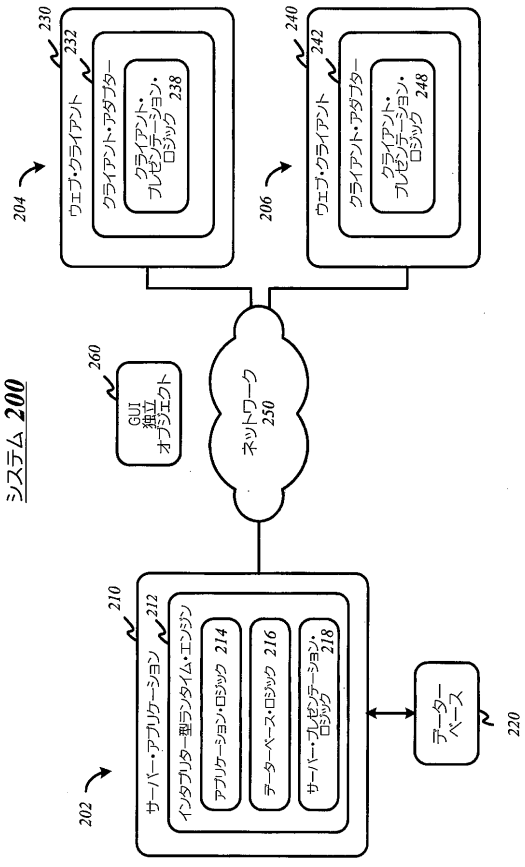
【図１Ｂ】



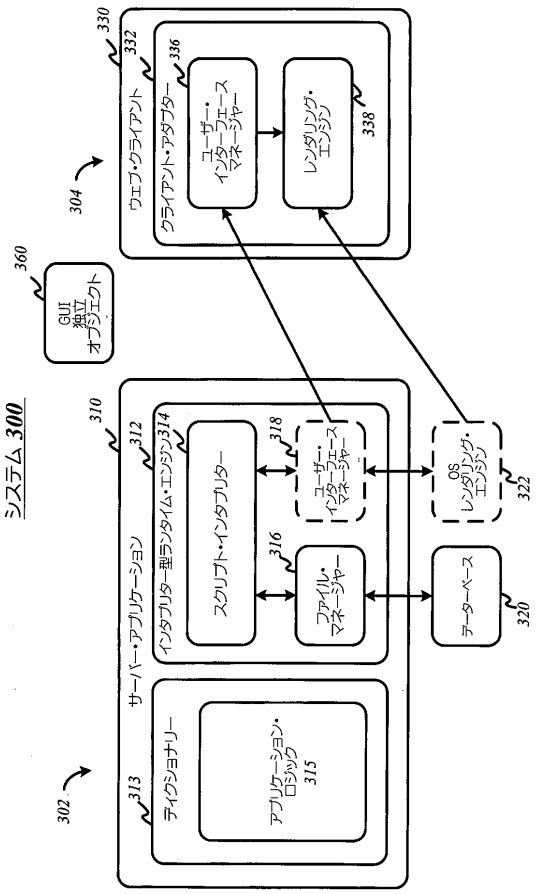
【図１Ｃ】



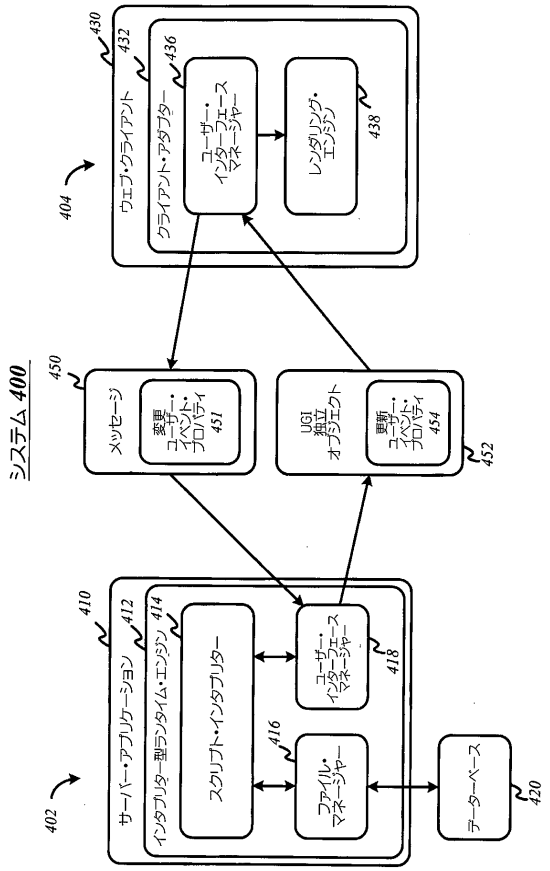
【図２】



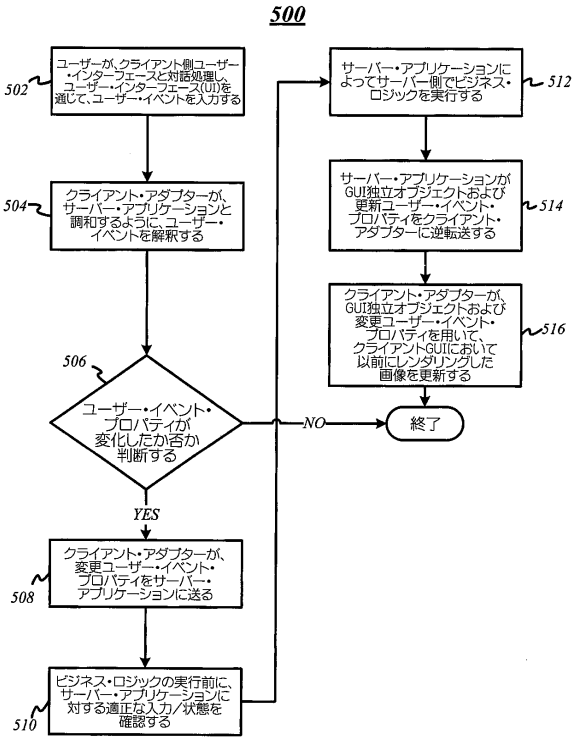
【図３】



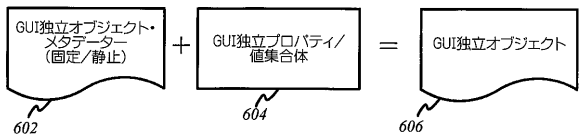
【図４】



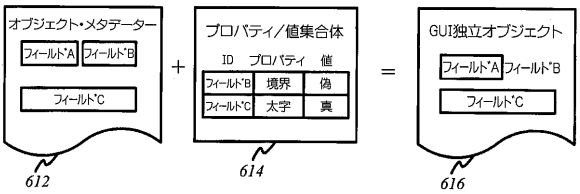
【図５】



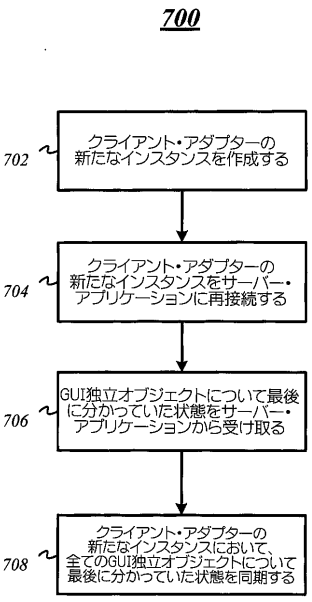
【図６Ａ】



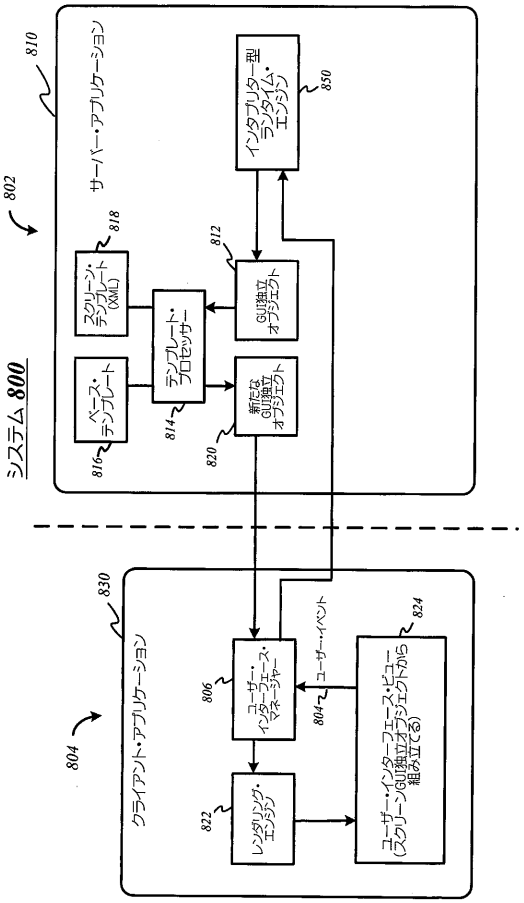
【図６Ｂ】



【図７】

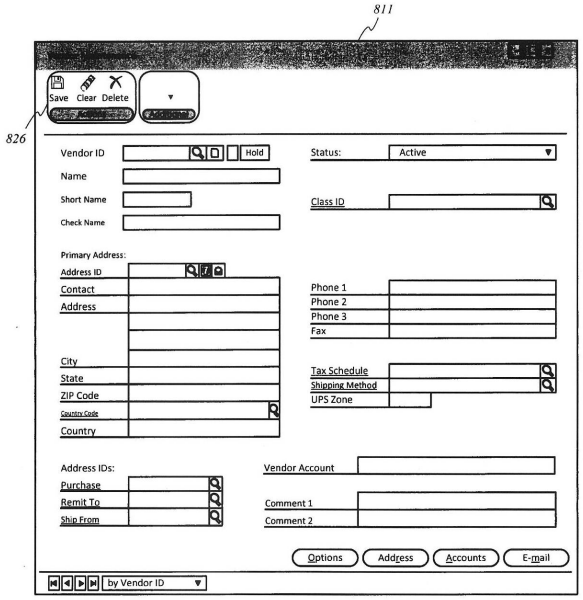


【図８Ａ】



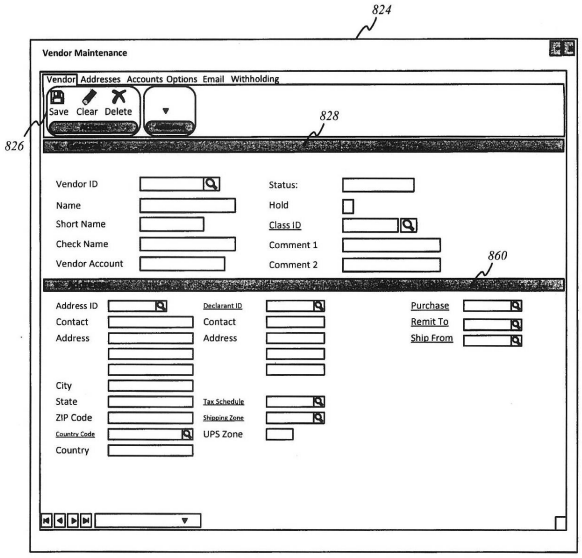
【図８Ｂ】

GUIビュー *811*

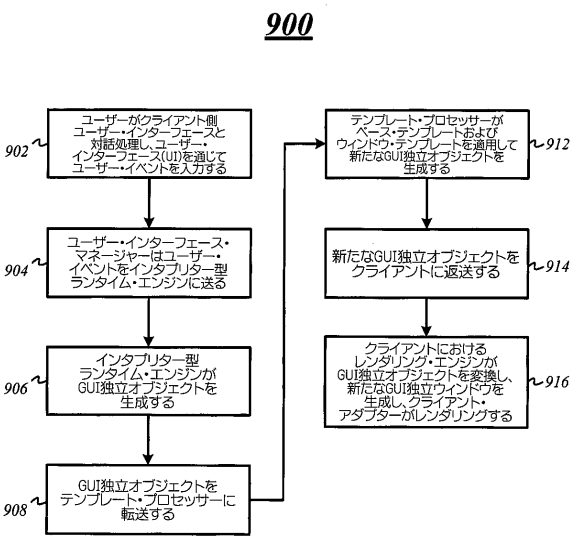


【図８Ｃ】

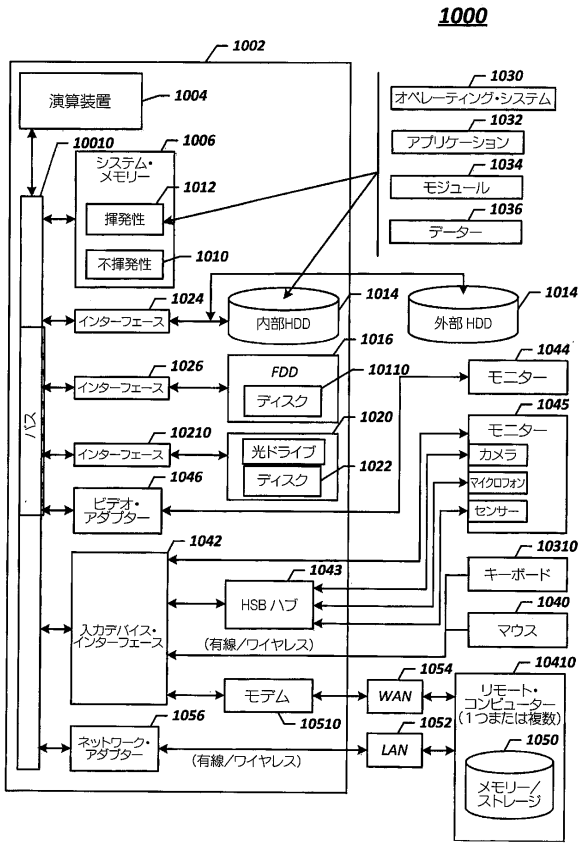
GUIビュー *824*



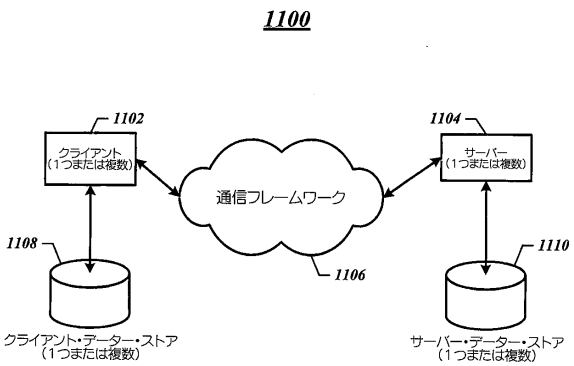
【図９】



【図１０】


1000
1002
演算装置
1004
10010
システム・メモリー
1006
揮発性
1012
不揮発性
1010
1030
オペレーティング・システム
1032
アプリケーション
1034
モジュール
1036
データー
1024
インターフェース
内部HDD
1014
外部 HDD
1014
1016
1026
インターフェース
FDD
ディスク
10110
1044
モニター
1045
モニター
カメラ
マイクロフォン
センサー
バス
10210
インターフェース
光ドライブ
ディスク
1020
1022
1046
ビデオ・アダプター
1042
1043
HSB ハブ
10310
キーボード
1040
マウス
入力デバイス・インターフェース
(有線/ワイヤレス)
10410
1054
WAN
リモート・コンピューター (1つまたは複数)
モデム
10510
1056
ネットワーク・アダプター
1052
LAN
1050
メモリー/ストレージ
(有線/ワイヤレス)

【図１１】


1100
1102
クライアント (1つまたは複数)
1104
サーバー (1つまたは複数)
通信フレームワーク
1106
1108
1110
クライアント・データー・ストア (1つまたは複数)
サーバー・データー・ストア (1つまたは複数)

フロントページの続き

(74)代理人 100153028
弁理士 上田 忠
(74)代理人 100120112
弁理士 中西 基晴
(74)代理人 100196508
弁理士 松尾 淳一
(74)代理人 100147991
弁理士 鳥居 健一
(74)代理人 100119781
弁理士 中村 彰吾
(74)代理人 100162846
弁理士 大牧 綾子
(74)代理人 100173565
弁理士 末松 亮太
(74)代理人 100138759
弁理士 大房 直樹
(72)発明者 パテル，ルシ
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，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ラーソン，カート
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，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 マレスカ，ルイス
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，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ロニー，ブライアン
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，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ニッセン，エリック
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，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ
(72)発明者 ナネンガ，ジョン
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，マイクロソフト コーポレーション，エルシーエイ－インターナショナル・パテンツ

審査官 三坂 敏夫

(56)参考文献 特開２００７－０８００６３（ＪＰ，Ａ）
特開２００４－００５５６８（ＪＰ，Ａ）
特開２００８－１１２４６０（ＪＰ，Ａ）
米国特許出願公開第２００５／０００５２５９（ＵＳ，Ａ１）
米国特許出願公開第２００３／００１４４４２（ＵＳ，Ａ１）
特表２０１０－５０３０５２（ＪＰ，Ａ）
米国特許出願公開第２００８／００５２３４８（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １５／００
９／０６

9／44－9／445
9／48－9／50
9／54－11／00
11／36