(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4936715号
(P4936715)
(45)発行日　平成24年5月23日(2012.5.23)　(24)登録日　平成24年3月2日(2012.3.2)

(51)Int.Cl.　F I
G06F 17/21 (2006.01)　G06F 17/21 570D
G06F 17/21 501T

請求項の数 19　(全 23 頁)

(21)出願番号　特願2005-352580 (P2005-352580)
(22)出願日　平成17年12月6日 (2005.12.6)
(65)公開番号　特開2006-323813 (P2006-323813A)
(43)公開日　平成18年11月30日 (2006.11.30)
審査請求日　平成20年11月19日 (2008.11.19)
(31)優先権主張番号　11/030,423
(32)優先日　平成17年1月6日 (2005.1.6)
(33)優先権主張国　米国 (US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　トリスタン　エー．デービス
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】ワードプロセッシングアプリケーションにおけるデータバインディング

(57)【特許請求の範囲】
【請求項１】
コンピュータがワードプロセッサアプリケーションにおいてワードプロセッサドキュメントを提供する方法であって、
前記ワードプロセッサドキュメントを開くことであって、前記ワードプロセッサドキュメントの表示表面は、
前記ワードプロセッサドキュメントのドキュメントフォーマットにおいて前記ワードプロセッサドキュメントのコンテンツを表示するために、１つまたは複数のコンテンツ領域を含み、
前記ワードプロセッサドキュメントの前記コンテンツに対する変更と関係のない、前記ワードプロセッサドキュメントの前記ドキュメントフォーマットに対する変更を受け取る
ように構成されていることと、
前記ワードプロセッサドキュメントを開いた後に、データストアを作成することであって、前記データストアは、
前記ワードプロセッサドキュメントの前記コンテンツを表すドキュメントデータの１つまたは複数のノードを格納し、
前記ワードプロセッサドキュメントの前記ドキュメントフォーマットに対する変更と関係のない、前記ワードプロセッサドキュメントの前記コンテンツに対する変更を受け取る

ように構成されていることと、
前記ドキュメントデータと共に前記データストアをロードすることと、
前記ドキュメントデータの前記１つまたは複数のノードに前記１つまたは複数のコンテンツ領域を関連付ける１つまたは複数のバインディングをロードすることと、
前記バインディングの１つが前記データストア内の存在しない場所を参照すると判定した場合に、前記１つまたは複数のコンテンツ領域を、前記１つまたは複数の前記ドキュメントデータの他のドキュメントデータにバインドすること
を含むことを特徴とする方法。
【請求項２】
前記データストア内の前記ドキュメントデータへのアクセスを提供することをさらに含むことを特徴とする請求項１に記載の方法。
【請求項３】
前記データストア内で直接前記ドキュメントデータを編集することをさらに含むことを特徴とする請求項２に記載の方法。
【請求項４】
前記ドキュメントデータの編集に関連付けられた前記１つまたは複数のコンテンツ領域を更新することをさらに含むことを特徴とする請求項３に記載の方法。
【請求項５】
前記１つまたは複数のコンテンツ領域の編集に基づいて、前記データストア内の関連付けられたドキュメントデータを更新することをさらに含むことを特徴とする請求項１に記載の方法。
【請求項６】
前記バインディングが前記データストア内の存在しない場所を参照する場合に、未帰着参照を作成することをさらに含むことを特徴とする請求項１に記載の方法。
【請求項７】
前記ワードプロセッサアプリケーションへのプログラミングインターフェースを使用して作成されたコードに基づいて、前記データストア内で、あるいは前記表示表面の１つまたは複数のコンテンツ領域内で変更を行う／変更に対応することをさらに含むことを特徴とする請求項１に記載の方法。
【請求項８】
前記データストア内の少なくとも１つのノードに関連付けられた修正に基づいて、複数のコンテンツ領域のコンテンツを更新することをさらに含むことを特徴とする請求項１に記載の方法。
【請求項９】
ワードプロセッサドキュメントを提供するワードプロセッサシステムであって、
表示表面であって、
前記ワードプロセッサドキュメントのドキュメントフォーマットにおいて前記ワードプロセッサドキュメントのコンテンツを表示するために１つまたは複数のコンテンツ領域を含み、
前記ワードプロセッサドキュメントの前記コンテンツに対する変更と関係のない、前記ワードプロセッサドキュメントの前記ドキュメントフォーマットに対する変更を受け取る
ように構成された前記表示表面と、
前記ワードプロセッサドキュメントを開いた後に作成されるデータストアであって、
前記ワードプロセッサドキュメントの前記コンテンツを表すドキュメントデータを保存し、前記ワードプロセッサドキュメントの前記コンテンツは、前記表示表面から別個に保存され、
前記ワードプロセッサドキュメントの前記ドキュメントフォーマットに対する変更と関係のない、前記ワードプロセッサドキュメントの前記コンテンツに対する変更を受け取る

ように構成された前記データストアと、
前記データストア内の前記ドキュメントデータに関連付けられたスキーマファイルであって、前記ドキュメントデータの構造を定義する前記スキーマファイルと、
前記データストア内の前記ドキュメントデータにおける１つまたは複数のノードを、前記ドキュメントのコンテンツ領域に関連付けるバインディングを作成し、前記バインディングの１つが前記データストア内の存在しない場所を参照すると判定した場合に、前記１つまたは複数の前記ドキュメントデータの他のドキュメントデータにバインドするように構成されたバインダと
を備えることを特徴とするワードプロセッサシステム。
【請求項１０】
前記１つまたは複数のリンクを監視するコードを作成するプログラミング手段をさらに備えることを特徴とする請求項９に記載のワードプロセッサシステム。
【請求項１１】
前記ワードプロセッサドキュメント内の複数のリンクされたエリアを編集するように動作することができ、これによって前記データストア内で関連する変更が行われることを特徴とする請求項９に記載のワードプロセッサシステム。
【請求項１２】
前記データストア内のノードを編集するように動作することができ、これによって前記ワードプロセッサドキュメントの１つまたは複数のコンテンツ領域内で関連する編集が行われることを特徴とする請求項９に記載のワードプロセッサシステム。
【請求項１３】
前記バインダは、未帰着の、またはバインドされたリンクを作成するようにさらに動作することができることを特徴とする請求項９に記載のワードプロセッサシステム。
【請求項１４】
コンピュータ実行可能コンポーネントを格納する記録媒体であって、前記コンピュータ実行可能コンポーネントは、
ワードプロセッサ表示表面を格納する表示表面ストアであって、ワードプロセッサドキュメントのフォーマットにおいて前記ワードプロセッサドキュメントのコンテンツを表示し、前記ワードプロセッサドキュメントの前記コンテンツに対する変更に関係のない、前記ワードプロセッサドキュメントの前記ドキュメントフォーマットに対する変更を受け取る前記表示表面ストアと、
前記ワードプロセッサドキュメントを開いた後に作成され、前記ワードプロセッサドキュメントの前記コンテンツを含む第１のデータを保存するコンテンツストアであって、前記第１のデータは、前記コンテンツの部分を含み、前記ワードプロセッサドキュメントの前記ドキュメントフォーマットに対する変更に関係のない、前記ワードプロセッサドキュメントの前記コンテンツに対する変更を受け取る１つまたは複数のノードを含み、前記表示表面ストアから別個に保存される前記コンテンツストアと、
前記コンテンツストア内の前記第１のデータの前記１つまたは複数のノードを、前記ワードプロセッサドキュメントの１つのエリアにバインドするバインディングであって、前記バインディングが前記コンテンツストア内の存在しない場所を参照すると判定された場合に、前記１つまたは複数のノードを再びバインドする前記バインディングと
を含むことを特徴とする記録媒体。
【請求項１５】
第２のデータを保存する第２のストアをさらに含むことを特徴とする請求項１４に記載の記録媒体。
【請求項１６】
前記コンテンツストアに関連付けられた第１のスキーマと、前記第２のストアに関連付けられた第２のスキーマとをさらに含むことを特徴とする請求項１５に記載の記録媒体。
【請求項１７】
前記コンテンツストアを監視するコードを作成するプログラミングインターフェースを

さらに含むことを特徴とする請求項１４に記載の記録媒体。
【請求項１８】
前記ワードプロセッサは、リンクが前記コンテンツストア内の不明の場所を参照する場合に未帰着参照を作成するようにさらに動作することができることを特徴とする請求項１４に記載の記録媒体。
【請求項１９】
前記ワードプロセッサは、未帰着の、またはバインドされたリンクを作成するようにさらに動作することができることを特徴とする請求項１４に記載の記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は一般に、ワードプロセッシングアプリケーションに関し、より詳細には、ＸＭＬデータとその表示とを切り離すことに関する。
【背景技術】
【０００２】
近年、マークアップ言語が広く人気を博している。マークアップ言語の一種である拡張マークアップ言語（ＸＭＬ）は、さまざまな種類のデータを識別し、交換し、処理するための方法を提供する汎用言語である。たとえばＸＭＬは、さまざまなアプリケーションプログラムが利用できるドキュメントを作成するために使用される。ＸＭＬファイルの要素は通常、関連付けられたネームスペースおよびスキーマを有する。
【０００３】
ネームスペースは、要素／属性の名前およびタイプを定義するためにＸＭＬドキュメント内で使用される、名前のコレクション用の一意の識別子である。ネームスペースの名前は一般に、ＸＭＬドキュメントのそれぞれのクラスを、一意に識別するために使用される。この一意のネームスペースによって、別々のソースに由来するが偶然に同じ名前を有する、マークアップ要素が区別される。
【０００４】
ＸＭＬスキーマは、ＸＭＬ環境内でデータを記述し、認証するための方法を提供する。スキーマは、ＸＭＬドキュメント内のコンテンツを記述するためにどんな要素および属性が使用されるか、それぞれの要素がどこで許可されるか、その中でどんなタイプのコンテンツが許可されるか、およびどの要素が他のどの要素の中に現れることができるかについて記載する。スキーマを使用することによって、ドキュメントは必ず一貫した予測可能な方法で構造化される。スキーマは、ユーザによって作成され、一般にＸＭＬなどの関連付けられたマークアップ言語によってサポートされることができる。ＸＭＬエディタを使用することによって、ユーザはＸＭＬファイルを操作し、自分が作成したスキーマに準拠するＸＭＬドキュメントを作成することができる。従来のワードプロセッサアプリケーションでは、カスタムＸＭＬスキーマ用のサポートがアプリケーションに追加され、ユーザは、カスタムＸＭＬマークアップ（たとえば＜ｔｉｔｌｅ＞）によってドキュメントのコンテンツに「タグを付ける」ことができるようになり、いわゆる従来のテキストの無差別な実行に対して意味（semantic meaning）を基本的に与えるようになった。つまり、それまではフォーマット設定を有する単なるテキストであり、他のアプリケーションが処理する上では何の意味も持たなかったドキュメントが、他の任意のＸＭＬ対応型アプリケーションが見つけ出して理解できる任意のユーザ定義のＸＭＬスキーマからのＸＭＬマークアップの特定の断片を含む、構造化されたＸＭＬドキュメントになることができるようになった。
【０００５】
基本的な例では、あるドキュメントの最上部にあるテキストには、ユーザ定義のＸＭＬスキーマからの＜ｔｉｔｌｅ＞というＸＭＬ要素によってタイトルとして「タグを付ける」ことができ、つまり他のＸＭＬ対応型アプリケーションは、テキストのこの範囲には「タイトル」が含まれていることを容易に理解し、それを適切に抽出できるようになる。こ

れによってバックエンドプロセスは、ドキュメントの一部を適切なセマンティクスおよびコンテキスト（たとえば、このテキストは＜ｔｉｔｌｅ＞である）と共に、インテリジェントに抽出することができる。

【０００６】

しかし、従来のワードプロセッサアプリケーションに付随する欠点は、カスタムＸＭＬマークアップの追加および維持が、ドキュメントの表示と結び付いているという事実に端を発する。すなわち既存の実装形態では、ワードプロセッサドキュメントのＸＭＬマークアップ（たとえば、ＸＭＬフォーマットで表記された顧客のインボイスの詳細）と、ドキュメント表面上のその表示（たとえば、３パラグラフのプレーンテキストの後に、特定のテーブル形式を有する５列×４行のテーブルが続く表示）の間に、不動のリンクがある。したがって、従来のワードプロセッサアプリケーションにおいて表記されるＸＭＬデータは、（その表示と結び付いているため）ドキュメントのコンテンツと厳密に一致しなければならない。たとえば、インボイス用のＸＭＬスキーマが、＜ｄａｔｅ＞の次に＜ａｄｄｒｅｓｓ＞が来て、その次に＜ｐｈｏｎｅＮｕｍｂｅｒ＞が来ると記載している場合、これら３つのＸＭＬ要素は、ドキュメント内に表示される際、厳密にその順序で現れなければならない。つまり、表示フォーマットが変更されると（たとえば、＜ｄａｔｅ＞を含むテーブルの行を移動すると）、そのドキュメント内に含まれるＸＭＬデータの構造も変化することになり、このデータを、関連するＸＭＬスキーマの構造に必ず準拠させるために、ソリューション開発者の側に余分なステップが必要となる。したがって、ドキュメントのエンドユーザには、表示を制約なく操作する自由は与えられない。これは、そうすれば必ずデータのセマンティクスが変更される可能性が生じ、そのデータ用のＸＭＬスキーマに違反するおそれがあるためである。

【０００７】

その上、従来のワードプロセッサアプリケーションに追加して開発されるソリューションは、バックエンドアプリケーション用のドキュメントとの間でデータの読み取り／書き込みを試みる際の表示の持つ意味を、より注意深く考慮する必要がある。したがって、ボールドテキストのパラグラフにタイトルとしてタグを付けると、結果として従来のワードプロセッサアプリケーションによって保存されるＸＭＬは、次のようになるだろう。

【０００８】

【表１】

```
<w:p>
    <Title>
        <w:r>
            <w:rPr>
                <w:b/>
            </w:rPr>
            <w:t>This is the title.</w:t>
        </w:r>
    </Title>
<w:p>
```

【０００９】

上に示されているように、カスタムＸＭＬのタグ付けは、従来のワードプロセッサアプリケーションにきわめて固有のＸＭＬタグ、この例ではｗ：ｐやｗ：ｒなどによって両側を囲まれている。つまり、このデータを処理しているＸＭＬ対応型ソリューションは、（＜Ｔｉｔｌｅ＞要素を含む）自分自身のデータフォーマットを理解するだけでなく、従来

のワードプロセッサアプリケーションのフォーマット設定の厳密な詳細も理解しなければならず、そうすることによって、自分自身のデータを探している際にその情報を飛び越して無視することを知るようになる。したがって、この種の実装形態では、ドキュメント内のテキストの外観をわずかに変更すると（たとえば、＜Ｔｉｔｌｅ＞要素のコンテンツをテーブルのセルにドラッグするなどすると）、結果として、周囲のワードプロセッサの元のタグ内において、カスタムＸＭＬタグの位置が大幅に変わる可能性があるため、依然としてユーザにはいくつかの要件が課されている。したがって、プログラマ／コード開発者は、従来のワードプロセッサアプリケーションが表示に基づいてカスタムＸＭＬ要素をどこに置くかを予測および理解して、さまざまな順列（permutations）のすべてに対処するための追加のコードを書くことが必要となる場合が多い。つまり、結果としてでき上がるソリューションは、依然として従来のワードプロセッサアプリケーションの特定のニーズに対処するための重要な論理コードを含む必要がある可能性がある。

【００１０】

従来のワードプロセッサアプリケーションと共に作業するプログラマ／コード開発者は、読み取りおよび書き込みオペレーションについて考慮する際、ドキュメントのレイアウトフォーマットの持つ意味についても考慮に入れる必要がある。たとえばユーザは、企業レポートを書くユーザ用の簡単な機能強化として、＜ＳｔｏｃｋＳｙｍｂｏｌ＞要素の値を取り込み、同じドキュメント内の＜ＣｏｍｐａｎｙＮａｍｅ＞要素内に企業のフルネームを記入するためにその値を使用することを試みる場合がある。ドキュメントの整合性を維持するために、ユーザは、ドキュメントからの希望のデータの読み取りおよび書き込みの双方において、ドキュメントの現在のレイアウトフォーマットを考慮する必要があり、その後でなければ、これらのアクションを実行するための機能的なコードを書くことができなかった。たとえばユーザは、従来のワードプロセッサアプリケーションのフォーマット設定情報を、ドキュメント内に挿入された際に所望の結果を生み出すように構築するために、自分が記入していた値がテーブルのセルや黒丸付きのリストなどの中にあったかどうかを知っておくことが必要となる場合がある。これが、ワードプロセッサアプリケーションの表示セマンティクスを理解するために追加のコーディングが必要となる別の潜在的な理由である。

【００１１】

従来のワードプロセッサアプリケーションのさらに別の制限は、ＸＭＬ要素の編集行為が「脆弱」とみなされる場合があるという点である。これは一部には、前述のように、ドキュメント表面上のタグの配置が、ユーザ定義のスキーマに基づいてＸＭＬデータの構造を決定するという事実によって、編集行為が制限されるためである。したがって、複数の問題が生じる場合がある。第１に、典型的なユーザオペレーション（たとえば、１つのセクションから別のセクションへのコピー／ペースト）によって、ＸＭＬ構造が変更され、関連するＸＭＬスキーマに従ってドキュメントが無効になる可能性がある。第２に、このようなワードプロセッサの実装形態では、カスタマー定義のＸＭＬスキーマによって必要とされるすべての要素は、何らかの形態でドキュメント表面上に含まれる必要がある。つまり開発者は、ドキュメント表面上に表示されずに、むしろメタデータとして機能するドキュメントに関する追加情報を保持するための方法として、関連するＸＭＬデータを作成する際に苦労する可能性がある。そして第３に、ドキュメント表面上では意味論的に不要な要素（たとえば、混合コンテンツをマークしていないノンリーフ要素）も、このようなワードプロセッサの実装形態に含む必要があり、一般的なユーザオペレーションによってＸＭＬデータが変更される可能性がさらに高まる。

【００１２】

多くの場合、ＸＭＬデータ（たとえば、メモドキュメントを構成するデータ）を定義するスキーマは、複数の異種の処理システム間でこのデータの通信を容易にするために、単一の標準化団体によって厳格に定義される傾向がある。しかしそのようにバックエンド通信を容易にする際に、ドキュメントデータに対する人間による可読性および編集可能性が犠牲にされることが多く、そのためにユーザにとっては、このデータを理解および解析す

ることが困難になる。たとえばＸＭＬ標準は、dd-mm-yyyyThh:mm:ss.ssssなど、日付用の標準的なフォーマットを定義することがある。ＸＭＬ対応型アプリケーションによって解析するには、すべての日付をこのフォーマットで表記する必要がある。明らかにこのフォーマットは、ユーザにとって正確に入力することが困難であり、ユーザが一般的に日付を入力する方法と一致しない場合が多い（たとえば多くの場面では、通常はdd-mm-yyyyではなくmm-dd-yyyyが使用される、といった具合である）。

【００１３】

【特許文献１】2004年9月30日に出願された「Methods, System, and Computer-Readable Medium For Managing Specific Types of Content In An Electronic Document」という名称の本譲受人による譲渡された米国特許出願第10/955,612号明細書

【発明の開示】

【発明が解決しようとする課題】

【００１４】

したがって必要とされているのは、開発者がワードプロセッサアプリケーションなどのアプリケーションにおいて、ＸＭＬデータとそのようなデータの表示とを切り離すことができるようにする方法である。

【課題を解決するための手段】

【００１５】

本発明の実施形態は、データと表示を切り離すことができるワードプロセッサドキュメントを作成するためのワードプロセッサアプリケーションを提供する。より詳細には、データは、ワードプロセッサドキュメントの表示フォーマットとは別の場所に保存されるワードプロセッサドキュメントへ入力され、そこから抽出されることができる。本発明の実施形態によれば、ワードプロセッサアプリケーションのユーザは、ワードプロセッサドキュメントデータ用の別個の記憶場所を作成し、そのデータのコンテンツと表示表面の間にリンク（またはバインディング）を確立することができる。ワードプロセッシングアプリケーションのユーザは、絶え間なく変化している可能性のある表示フォーマットの複雑さに対処する必要はなく、リンクされているデータを直接変更することによって、ワードプロセッサドキュメントの表示の内容を修正することができる。ビュー／表示とデータを切り離すことによって、ユーザは、自由な形式でドキュメントの編集を継続することができ、その一方で、単一の既知の構造的なフォーマットにおける持続的なファイルフォーマットにおいてデータを表すことができる。ワードプロセッサドキュメントのコンテンツを修正するために外部プロセスを使用する場合、ユーザは、表示表面またはフォーマットの現在のレイアウトをまったく意識せずに、ワードプロセッサファイル内に保存されているＸＭＬデータの切り離された断片を編集することができる。またこれらのリンク（またはバインディング）は、表示表面の構造に対する変更によってＸＭＬデータの構造が決定されないように、また逆にＸＭＬデータの構造に対する変更によってワードプロセッサドキュメントの表示表面のレイアウトが決定されるように作成され、これらのリンクは、ドキュメントのコンテンツに悪影響を与えることなく作成または破棄することができる。本発明の実施形態は、ワードプロセッサドキュメント内における構造化されたデータの追加、編集、および抽出を簡略化する上で役に立つ。

【００１６】

本発明の実施形態によれば、ワードプロセッサドキュメントには、テキストの特定の領域用のセマンティクスによってタグを付けることができるが、関連付けられたデータは、そのドキュメント内の別のＸＭＬデータファイルに保存される。このデータは、データバインディングを使用して、そのドキュメント内のテキストの範囲、すなわちタグ付けされた領域にリンクされる。したがってデータは、データバインディングがそのコンテンツをドキュメントの表示表面内のどこに置くか、またはそのデータがどのように表示されるかにかかわらず、自分自身のＸＭＬデータファイル内の一貫した場所（「データストア」）に保存することができる。したがってユーザは、ドキュメント内におけるデータの移動／変更／削除などについて心配する必要はない。これは、そのデータ用のＸＭＬ構造は、一

貫した場所におけるカスタムＸＭＬ内に配置され、表示表面への変更によって影響を受けることはありえないためである。さらに本発明の実施形態は、ワードプロセッサアプリケーション用のプログラミングインターフェースを提供する。ユーザは、このプログラミングインターフェースを使用して、データストア、またはドキュメントの１つまたは複数のエリア内の変更に対応するためのコードを作成することができる。

【発明を実施するための最良の形態】

【００１７】

本明細書および特許請求の範囲を通じて、次の用語は、ここで明確に関連付けられた意味を有するものとする。ただし文脈によってその他の意味が明示される場合は除く。

【００１８】

「データ」という用語は、ワードプロセッシングドキュメントに伴う、ワードプロセッシングドキュメントを参照する、またはワードプロセッシングドキュメントによって使用される任意の補足的な情報を指す。この情報は、しばしば膨大であり、ドキュメントの表示層上にその全体が露出される可能性は低い。

【００１９】

「マークアップ言語」または「ＭＬ」という用語は、アプリケーションによってドキュメントの部分がどのように解釈されるかを指定する、ドキュメント内の特別なコード用の言語を指す。ワードプロセッサファイル内で、マークアップ言語は、テキストがどのようにフォーマットされ、またはレイアウトされるかを指定する。

【００２０】

「要素」という用語は、ＸＭＬドキュメントの基本的なユニットを指す。要素は、ＸＭＬドキュメント用の属性、他の要素、テキスト、および他のコンテンツ領域を含むことができる。

【００２１】

「表示」という用語は、ドキュメントの見える部分、すなわちそのドキュメントが印刷された場合に表示されるテキストおよびレイアウトを指す。

【００２２】

「タグ」という用語は、ドキュメント内に挿入されて、ＸＭＬドキュメント内の要素を描写する文字を指す。それぞれの要素は、２つのタグ、すなわち開始タグおよび終了タグしか有することができない。空の（すなわちコンテンツのない）要素を有することも可能であり、この場合は１つのタグが認められる。

【００２３】

タグとタグの間にあるＸＭＬコンテンツは、要素の「子」（または子孫）とみなされる。したがって、要素のコンテンツ内に組み込まれた他の要素は、「子要素」、または「子ノード」、あるいは要素と呼ばれる。要素のコンテンツ内に直接組み込まれたテキストは、要素の「子テキストノード」とみなされる。子要素と要素内のテキストとが合わさって、その要素の「コンテンツ」を構成する。

【００２４】

「属性」という用語は、特定の値に設定され、要素に関連付けられている追加のプロパティを指す。要素は、自分に関連付けられた任意の数の属性設定を有することができ、そうした属性設定をまったく有さない場合もある。属性は、さらなる要素を含まない要素、またはテキストノードとして扱われる要素に、さらなる情報を関連付けるために使用される。

【００２５】

「コンテンツ領域」という用語は、ユーザによって入力されるタイプのコンテンツ用のコンテナとして機能する、ドキュメント内の境界で区切られた領域、および／または任意選択でラベル付けされた領域を指す（特許文献１参照）。

【００２６】

「ＸＰａｔｈ」は、パターン式（pattern expression）を使用して、ＸＭＬドキュメント内のノードを識別する演算子である。ＸＰａｔｈパターンは、ＸＭＬドキュメントを通

るパスを記述する、子要素名の切り離されたリストである。このパターンは、パスにマッチする要素を「選択」する。

【００２７】

「ＸＭＬデータストア」という用語は、ファイルが開いている間にワードプロセッサドキュメント内に（たとえばＸＭＬフォーマットで）保存されているデータを格納および修正するためのアクセスを提供する、ワードプロセッシングドキュメント内のコンテナを指す。

【００２８】

「データバインディング」という用語は、コンテンツ領域のコンテンツを保存できるワードプロセッサドキュメント内のＸＭＬデータの１つまたは複数の断片内においてＸＰａｔｈの位置を確定する、コンテンツ領域上のプロパティを指す。本明細書で使用される際、

「ｒｅｆ」は、個々のバインディングによって使用される、それぞれのバインドされたＸＭＬドキュメント用の一意の整数を指し、

「ＩＤ」は、ＸＭＬデータストア内の個々の項目用の一意のＩＤを指し、

「ｓｅｌｅｃｔｉｏｎＮａｍｅｓｐａｃｅｓ」は、ＸＭＬデータストア内において関連付けられたＸＭＬドキュメント用の（ネームスペースと略称を結び付ける）プレフィックスマッピングを指し、

「ｒｏｏｔＵＲＩ」は、ＸＭＬデータストア内において関連付けられたＸＭＬドキュメントのルートネームスペースを指す。

【００２９】

（例示的な動作環境）

本発明の実施形態は、ＸＭＬデータの保存と表示を切り離すことができるワードプロセッサドキュメントを作成するためのワードプロセッサアプリケーションを提供する。より詳細には、ワードプロセッサドキュメントとの間で入力および抽出できるデータは、ワードプロセッサドキュメントの表示フォーマットとは別の場所に保存される。したがって、ワードプロセッサアプリケーションのユーザは、ワードプロセッサドキュメント内に含まれるＸＭＬデータ用の別個の記憶場所を作成し、そのデータのコンテンツと表示表面の間にリンク（またはバインディング）を確立することができ、これによってユーザは、表示の内容を編集することによって、関連付けられているＸＭＬデータを編集することができ、その一方で同じトークンによって、関連付けられているＸＭＬデータの構造をユーザが変更することが防止される。たとえば、インボイス用のデータは、ワードプロセッサファイルフォーマット内にＸＭＬとして別個に保存することができ、これによって、ドキュメント内でリンクの位置を移動しても、切り離されているデータストアの構造は変更されない。したがって、この構造化されたデータは、ユーザのドキュメント編集方法によって影響を受けることのない既知の構造を有するようになるため、この構造化されたデータのバックエンド処理はより容易になる。ユーザは、ワードプロセッサドキュメント内でデータを編集し、リッチ表示フォーマットの設定などを含めてデータのフォーマットを設定することができ、テキストコンテンツに対する変更のみが、ドキュメントの背後に保存されているＸＭＬデータへ「押し」戻される。しかし、本発明によれば、ユーザとワードプロセッサドキュメントとの対話を介して更新されるすべてのデータは、ＸＭＬの当初の元のストリーム内において利用可能である。本発明はまた、リンクされているＸＭＬデータを直接変更することによって、ワードプロセッサドキュメントのコンテンツを修正できるようにし、絶え間なく変化している可能性のあるそのデータの表示フォーマットの複雑さには対処する必要はない。その際、ワードプロセッサドキュメント内での構造化されたデータの追加、編集、および抽出が大幅に簡略化される。

【００３０】

図１を参照すると、本発明を実施するための１つの典型的なシステムは、コンピューティングデバイス１００などのコンピューティングデバイスを含む。非常に基本的な構成では、コンピューティングデバイス１００は通常、少なくとも１つの処理装置１０２および

システムメモリ１０４を含む。コンピューティングデバイスの厳密な構成およびタイプに応じて、システムメモリ１０４は、（ＲＡＭなどの）揮発性、（ＲＯＭやフラッシュメモリなどの）不揮発性、またはその双方の組合せとすることができる。システムメモリ１０４は通常、オペレーティングシステム１０５、１つまたは複数のアプリケーション１０６を含み、プログラムデータ１０７を含む場合もある。一実施形態では、アプリケーション１０６は、ワードプロセッサアプリケーション１２０を含むことができる。この基本的な構成が、破線１０８内のこれらのコンポーネントによって、図１に示されている。

【００３１】

コンピューティングデバイス１００は、追加の機能または機能性を有することができる。たとえばコンピューティングデバイス１００は、磁気ディスク、光ディスク、テープなどの（取り外し可能および／または取り外し不可能な）追加のデータストレージデバイスを含むこともできる。このような追加のストレージは、取り外し可能なストレージ１０９、および取り外し不可能なストレージ１１０として図１に示されている。コンピュータストレージメディアは、コンピュータ読取り可能命令、データ構造、プログラムモジュール、他のデータなどの情報を記憶するための任意の方法または技術において実装される揮発性メディアおよび不揮発性メディア、ならびに取り外し可能なメディアおよび取り外し不可能なメディアを含むことができる。システムメモリ１０４、取り外し可能なストレージ１０９、および取り外し不可能なストレージ１１０は、すべてコンピュータストレージメディアの例である。コンピュータストレージメディアは、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）または他の光ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージまたは他の磁気ストレージデバイス、あるいは希望の情報を記憶するために使用可能で、コンピューティングデバイス１００によってアクセス可能な他の任意のメディアを含むが、これらには限定されない。このような任意のコンピュータストレージメディアは、デバイス１００の一部とすることができる。コンピューティングデバイス１００は、キーボード、マウス、ペン、音声入力デバイス、タッチ入力デバイスなどの（１つまたは複数の）入力デバイス１１２を有することもできる。またディスプレイ、スピーカ、プリンタなどの（１つまたは複数の）出力デバイス１１４を含むこともできる。これらのデバイスは、当技術分野でよく知られており、本明細書において詳細に論じる必要はない。

【００３２】

また、コンピューティングデバイス１００は通信接続１１６も含み、これによってネットワークを介するなどして、他のコンピューティングデバイス１１８と通信することができる。通信接続１１６は、通信メディアの一例である。通信メディアは通常、コンピュータ読取り可能命令、データ構造、プログラムモジュール、または搬送波や他の伝送機構などの被変調データ信号内の他のデータによって具体化することができ、任意の情報伝達メディアを含む。「被変調データ信号」という用語は、情報をその信号内で符号化するような方法で設定または変更された、その特性の１つまたは複数を有する信号を意味する。たとえば通信メディアは、有線ネットワークや直接有線接続などの有線メディアと、音波メディア、無線周波数（ＲＦ）メディア、赤外線メディア、他の無線メディアなどの無線メディアとを含むが、これらには限定されない。本明細書で使用するコンピュータ読取り可能記憶媒体という用語は、ストレージメディアと通信メディアの両方を含む。

【００３３】

ワシントン州レドモンドのＭＩＣＲＯＳＯＦＴ　ＣｏｒｐｏｒａｔｉｏｎによるＷＩＮＤＯＷＳ（登録商標）　ＸＰオペレーティングシステムなど、ネットワーク化されたパーソナルコンピュータの動作を制御するのに適したオペレーティングシステム１０５を含む、複数のプログラムモジュールおよびデータファイルを、コンピューティングデバイス１００のシステムメモリ１０４内に格納することができる。またシステムメモリ１０４は、後述するワードプロセッサアプリケーション１２０、およびその他ものなどの１つまたは複数のプログラムモジュールを格納することもできる。ワードプロセッサアプリケーション１２０は、電子ドキュメントを作成し、編集し、および処理するための機能を提供する

ように動作する。

【００３４】

本発明の一実施形態によれば、ワードプロセッサアプリケーション１２０は、ＭＩＣＲＯＳＯＦＴ　Ｃｏｒｐｏｒａｔｉｏｎの ＷＯＲＤプログラムを含む。しかし本発明のさまざまな態様を具体化するために、他の製造業者によるワードプロセッサアプリケーションプログラムも利用できることが理解できるはずである。さらに本発明のさまざまな態様は、ワードプロセッサアプリケーションプログラムに限定されるものではなく、スプレッドシートアプリケーションプログラム、データベースアプリケーションプログラム、プレゼンテーションアプリケーションプログラム、描画またはコンピュータ支援アプリケーションプログラムなど、さまざまな形態のコンテンツ（たとえばテキスト、イメージ、画像など）を処理できる他のアプリケーションプログラム１０６も利用できることが理解できるはずである。

【００３５】

本発明の実施形態は、コンピュータプロセスやコンピューティングシステムとして、またはコンピュータプログラム製品やコンピュータ読取り可能記憶媒体などの製品として実装することができる。コンピュータプログラム製品は、コンピュータシステムによって読み取り可能な、コンピュータプロセスを実行するための命令のコンピュータプログラムをコード化している、コンピュータストレージメディアとすることができる。コンピュータプログラム製品は、コンピューティングシステムによって読取り可能な、コンピュータプロセスを実行するための命令のコンピュータプログラムをコード化している、搬送波上で伝搬される信号とすることもできる。

【００３６】

（ワードプロセッサアプリケーションにおけるデータのバインディング）

図２は、本発明の実施形態を実施するための典型的な環境を示すブロック図である。図２に示されている典型的な環境は、ワードプロセッサアプリケーション２０２と、ワードプロセッサドキュメント２０４と、プログラミング環境２０６と、データストア２０８と、スキーマファイル２１０と、ＸＭＬ処理モジュール２１２とを含むワードプロセッサ環境２００である。しかし前述のように、本発明は、スプレッドシートアプリケーションプログラム、データベースアプリケーションプログラム、プレゼンテーションアプリケーションプログラム、描画またはコンピュータ支援アプリケーションプログラム、およびその他のプログラムなどの、さまざまな形態のコンテンツ（たとえばテキスト、イメージ、画像など）を処理できるその他のアプリケーションプログラム１０６に適用することもできる。プログラミング環境２０６は、ＸＭＬ処理モジュール２１２用の簡単なアプリケーションプログラミングインターフェース（ＡＰＩ）を提供することができ、これによって、ドキュメント２０４またはＸＭＬデータストア２０８のコンテンツのいずれかを修正するコードを作成することができる。本発明は、本明細書に記載されているいかなる特定の実施形態または実施例によっても限定されることを意図するものではないことが理解できるであろう。たとえばワードプロセッサ環境は、複数のワードプロセッサドキュメント２０４、データストア２０８、および／またはスキーマファイル２１０を含むことができる。本発明の一実施形態によれば、ＸＭＬ処理モジュール２１２は、拡張マークアップ言語に従ってフォーマット設定されたデータを処理するために、ワードプロセッサアプリケーション２０２によって使用される。適切なＸＭＬ処理モジュール２１２は、ワシントン州レドモンドのＭＩＣＲＯＳＯＦＴ　Ｃｏｒｐｏｒａｔｉｏｎによって製造および市販されているＭＳＸＭＬである。

【００３７】

ワードプロセッサアプリケーション２０２は、１つまたは複数の自分自身のネームスペースと、ワードプロセッサアプリケーション２０２に関連付けられた（１つまたは複数の）ドキュメント２０４と共に使用するために定義される、スキーマ２１０またはスキーマのセットとを含む。スキーマ２１０によってワードプロセッサアプリケーション２０２用に定義されたタグおよび属性のセットは、ドキュメント２０４のフォーマットを定義する

。後述するように、本発明の実施形態によれば、データストア２０８はデータ２０９を含むことができる。スキーマ２１０は、データストア２０８内でデータ２０９に添付されることが好ましい。ワードプロセッサドキュメント２０４はまた、後述するように、ユーザによって作成されたコンテンツ領域２０７を含む。複数のデータストア２０８、関連付けられたＸＭＬデータ２０９、およびスキーマ２１０は、ワードプロセッサアプリケーション２０２の一部として含むことができる。所与のＸＭＬデータ２０９内に含むことができるデータのタイプおよび構造を規定する、データのタイプに関する文法的なルールのセットをＸＭＬデータ２０９に提供するために、ユーザが所与のＸＭＬデータ２０９に注釈を付ける際に使用できる、ＸＭＬの要素およびタグのそれぞれを規定するルールを提供するための、１つまたは複数のＸＭＬスキーマ２１０を、ＸＭＬデータ２０９に関連付けることができる。スキーマ２１０は、それらの要素をＸＭＬデータ２０９に適用できる順序を規定するルールと、ＸＭＬデータ２０９に適用される個々の要素に関連付けられた特定のルールとを含む。

【００３８】

本発明の実施形態は、ワードプロセッシングドキュメントと共に保存される別個のデータストア２０８の存在を介してデータと表示を切り離すことができる、ワードプロセッサドキュメント２０４を作成するように動作できるワードプロセッサアプリケーション２０２を提供する。より詳細には、ワードプロセッサドキュメント２０４との間で入力および抽出できるデータは、そのドキュメントのデータストア２０８内の１つまたは複数のＸＭＬデータ２０９のファイルに保存され、これによってそのデータは、ワードプロセッサドキュメント２０４の表示フォーマットから切り離される。したがってワードプロセッサアプリケーション２０２のユーザは、ワードプロセッサドキュメント２０４のデータ用の別個の記憶場所を作成し、そのデータのコンテンツと表示表面２０３の間に１つまたは複数のコンテンツ領域２０７に関連付けられたリンク（またはバインディング）２０５を確立することができる。これによってユーザは、表示の内容を編集することによってそのデータを編集することができ、その一方で同じトークンによって、データ２０９の構造をユーザが変更することが防止される。ドキュメント２０４内でコンテンツ領域２０７の位置を移動しても、切り離されているデータストア２０８内のＸＭＬデータ２０９の構造は変更されない。さらにデータの表示、ボールド、イタリック、行端ぞろえなどに対して行われる変更は、データの構造に影響を与えない。したがってＸＭＬデータ２０９は、ユーザがドキュメント２０４を編集する方法によって影響を受けることのない既知の構造と一致するため、この構造化されたデータのバックエンド処理は簡略化される。

【００３９】

本発明の実施形態では、絶え間なく変化している可能性のある表示フォーマットの複雑さに対処する必要はなく、リンクされているデータを直接変更することによって、ワードプロセッサドキュメント２０４のコンテンツを修正することができる。これを行う際に、ワードプロセッサドキュメント２０４内での構造化されたデータの追加、編集、および抽出が、大幅に簡略化される。さらに、構造化されたＸＭＬデータ２０９にバインドされたデータバインディング２０５は、データの構造に影響を与えることなくドキュメント２０４内で移動することができる。コンテンツ領域２０７上のデータバインディング２０５は、ユーザインターフェースまたはプログラミングウィンドウ２０６を介して定義できるＸＰａｔｈ式を使用して有効にすることが好ましい。

【００４０】

ユーザは、ＸＰａｔｈ式（ＸＭＬツリー内のノード２１３を識別するための標準的なＸＭＬメソッド）を使用して、ドキュメントコンテンツ領域のバインド先となるべき所望のＸＭＬノード２１３を一意に識別する。ワードプロセッシングアプリケーション２０２は、データバインディング領域用の所望のターゲットを見つけ出すためにＸＰａｔｈを自動的に解析して使用するように動作する。つまり、ＸＰａｔｈの規格に精通している開発者は、事実上半動的（semi-dynamic）なデータバインディング２０５を作成するためのＸＭＬのこの使用法を、利用できるということでもある。すなわち、データ２０９に対する他

の変更、または表示２０３に対する変更に基づいて、異なるターゲットノード２１３を識別することである。たとえばユーザが、与えられた期間内に最大の売り上げを達成した従業員の名前を表示したいと仮定する。この情報が、ドキュメント２０４に関連付けられたＸＭＬデータ２０９内にある場合、ユーザは、達成された明細の最高の数字を有する個人の名前にリンクするＸＰａｔｈ式を作成することができ、データが変更されると、そのリンクは適切な場所（ノード２１３）へ自動的に移動する。このリンクはまた、コード２１１、ユーザインターフェース、またはプログラミング環境２０６を使用することによって、複数のノード２１３の間で変更することもできる。

【００４１】

あるいは、ユーザは、ドキュメントコンテンツ領域２０７のバインド先とすることができるデータ２０９内のノード２１３を表すオブジェクトを一意に識別するデータバインディングを作成することもできる。ワードプロセッシングアプリケーション２０２は、データバインディング領域用の所望のターゲットを見つけ出すために、ＸＰａｔｈを自動的に確定するように動作する。しかしこれは、この場合、同じＸＰａｔｈ式とは対照的に、データバインディング２０５は、自分が必ず同一のオブジェクトを指し示すようにそのＸＰａｔｈを自動的に更新させることを意味する。

【００４２】

プログラミングコード２１１は、簡単に前述したように、ＸＭＬ処理モジュール２１２を利用して、いずれの方向へも（すなわち、ドキュメント表面２０３上のコンテンツ領域２０７から、データストア２０８内のＸＭＬデータ２０９内におけるノード２１３へ、およびその逆へ）移動する変更に、対応するように作成することができる。ユーザは、ドキュメント表面２０３とデータストア２０８内の特定のコンテンツとの関係を定義する、コード２１１を作成することができる。さらに編集、追加、削除など、ドキュメント２０４のバインドされた領域内またはデータストア２０８内での変更、トラッピングまたはインターセプティングイベントに対応するコード２１１を作成することができる。たとえばユーザが、必ず１つのドキュメントのみが特定のタイトルを使用できるようにしたいと仮定する。コード２１１は、タイトルノードに何が入力されるかに基づいて、そのタイトルが既に使用されているかどうかについて中央データベースをチェックすることができる。そのタイトルが使用されている場合、コード２１１は、別のタイトルを入力するようユーザに促すか、および／またはそのタイトルが利用できないことをユーザに警告することができる。本発明の実施形態によって、ユーザは、関連付けられたＸＭＬによってコード２１１を１回だけ書けばよく、このコードは、ターゲットアプリケーションの厳密なセマンティクスについて心配することなく、ＸＭＬ構造の使用をサポートするすべてのドキュメントタイプに移植することができ、これによってアプリケーション開発が大幅に簡略化および合理化される。

【００４３】

本発明の実施形態によれば、ワードプロセッサドキュメント２０４には、テキストの特定の領域（たとえばＴｉｔｌｅやＢｏｄｙなど）用のセマンティクスを表すコンテンツ領域２０７によってタグを付けることができ、データバインディング２０５を追加することによって、そのコンテンツ領域２０７内の関連付けられたテキストは、ドキュメント２０４内のデータストア２０８内におけるあるＸＭＬデータ２０９の内部のノード２１３内に保存される。データ２０９は、１つまたは複数のデータバインディング２０５を使用して、ドキュメント内のコンテンツ領域２０７、すなわちタグ付けされた領域にリンクされる。データ２０９は、表示２０３内の関連付けられたコンテンツ、すなわちドキュメント２０４の特定のドキュメントを編集する際にユーザが対話するデジタル表示（たとえば、特定のユーザドキュメントを示すＷＯＲＤウィンドウ）の位置を、データバインディング２０５がどこに示すか、またはそのデータ２０９がどのように表示されるかを問わず、自分自身のＸＭＬストリーム内の一貫した場所（「データストア」）に保存することができる。したがってユーザは、ＸＭＬデータ２０９が一貫した場所のデータストア２０８内に配置されるため、ドキュメント２０４内で移動するデータについて心配する必要はない。さ

らにデータストア２０８は、ドキュメント２０４内のデータバインディング２０５によって参照されないデータ２０９を含むことができる。メタデータなどの「特別な」データは、ドキュメントユーザには関係のない可能性のある追加の情報を、ソリューション開発者などのユーザに提供する。

【００４４】

本発明の実施形態によれば、データの構造は、切り離された場所、すなわちドキュメントのデータストア２０８内のＸＭＬデータ２０９の１つまたは複数の断片内で保持され、これによってユーザは、データ構造に影響を与えることなく、表示２０３内でリンク（すなわちデータバインディング２０５）を移動することができる。したがってＸＭＬデータ２０９の構造は変更されず、ワードプロセッサドキュメント２０４に関連付けられたＸＭＬデータ２０９の表示のみが変更される。したがってドキュメント２０４内のデータ表示のフォーマットを変更しても、データストア２０８の構造は影響を受けない。ユーザがドキュメント表面２０３を操作しても、実際のデータ２０９は移動しておらず、したがってユーザは、ストア２０８内に別個に保持されているデータ２０９に対する悪影響について心配する必要はなく、表示を完全に制御することができる。このように本発明の実施形態によって、ユーザは、表示情報から切り離されたカスタムＸＭＬ情報にアクセスすることができる。

【００４５】

１つまたは複数のデータバインディング２０５を使用して、本明細書ではＸＭＬデータ２０９と呼ばれるデータソースのコンテンツ（データストア２０８内のＸＭＬデータ２０９）をドキュメント２０４内の場所へ「バインド」することができる。本明細書で使用されるＸＭＬデータ２０９は、（プレーンまたはリッチフォーマットの）テキスト、イメージ、特定のタイプのコンテンツ（日付）など、任意のタイプのコンテンツを含む。データバインディング２０５の構造は、ＸＰａｔｈリンクとして記述することができ、これによってワードプロセッサアプリケーション２０２は、ドキュメント２０４に関連付けられたＸＭＬデータストア２０８内のＸＭＬノード２１３に対して接続する／同期化する／リンクを維持することができる。ＸＭＬデータストア２０８は、ドキュメント２０４の一部であること（すなわちストア２０８は、ワードプロセッサファイルと共に保存され、特定のドキュメント２０４と共に移動するか、または特定のドキュメント２０４に関連付けられること）が最も好ましい。データバインディング２０５はまた、データ２０９が、表示（たとえば、ワードプロセッサのドキュメント表面２０３）とデータストア２０８との間でどのように変換されるべきかを制御するための情報を含むこともできる。ワードプロセッサアプリケーション２０２のユーザは、データストア２０８内に保存されるデータ２０９（バックエンドアプリケーションによって操作されるデータ）を標準的なフォーマットで保存し、その一方で同じ情報をワードプロセッサドキュメント２０４内により分かりやすいフォーマットで表示することができる（たとえばユーザには２００４年１月２９日と見えるが、データは２９－０１－００４Ｔ１２：００：００．０００００としてｄａｔｅＴｉｍｅフォーマットで保存されている）。別の例として、データバインディング情報はイメージを含むことができ、データ２０９にとって、このイメージは外見的には意味のない文字列として表されるが、上述の同じ変換原理によれば、ユーザにはワードプロセッサドキュメント２０４のコンテンツ領域２０７にイメージが見えることになる。ユーザは、このイメージに追加／変更を行うことができ、ＸＭＬデータ２０９にとってはＸＭＬでコード化された表示が維持され、これによっていかなるバックエンドプロセスも、その情報を保存／操作することができる。

【００４６】

一実施形態によれば、ユーザは、データバインディング情報２０５をコンテンツ領域２０７へ追加する際、ＸＰａｔｈ式を指定することによって、興味のある（たとえば、１つまたは複数のノード２１３を識別する）リンクされたＸＭＬデータ２０９を提供する。いったんバインドされると、このコンテンツ領域２０７のコンテンツは、そのＸＰａｔｈによって返されたノード２１３のコンテンツ（ＸＭＬデータ）にリンクまたはバインドされ

る。したがって、つまりＸＰａｔｈによって返されたＸＭＬノード２１３を変更するような方法でそのＸＭＬノード２１３が追加／削除／変更される場合、ドキュメント２０４内のコンテンツ領域２０７のコンテンツは自動的に更新される。あるいは変更が生じた結果、特定のデータバインディング２０５によって返されるノード２１３がない場合、そのデータバインディング２０５は、後述する「未帰着参照（dangling reference）」の状態へと移行する。

【００４７】

たとえば、ドキュメント２０４は次のパラグラフを含み、その中の「Ｍｉｃｒｏｓｏｆｔ　Ｃｏｒｐｏｒａｔｉｏｎ」は、そのドキュメント用のデータストア２０８内におけるいくつかのリンクされたＸＭＬデータ２０９内のＸＰａｔｈ／ｃｏｎｔｒａｃｔ（１）／ｈｅａｄｅｒ（１）／ｃｏｍｐａｎｙ（１）にバインドされた（下線で表示されている）プレーンテキストコンテンツ領域２０７に相当すると仮定する。表示２０３上に表示されるパラグラフは、次のようになる。

"Microsoft Corporation is located at One Microsoft Way."

【００４８】

一実施形態によれば、プログラミング環境２０６において、（たとえば）次のようなコードを１行だけ指定することによって、リンクをセットアップすることができる。

【００４９】

【表２】

```
Document.ContentRegions.Add().DataBinding.Add("/contract(1)/header(1)
/company(1)")
```

【００５０】

対応するリンクされたＸＭＬデータ２０９は、次のような体裁とすることができる（リンク先のノード２１３は、シングルクォーテーションマークで示されている）。

【００５１】

【表３】

```
<contract>
    <header>
        '<company>Microsoft Corporation</company>'
        <company>Contoso Corporation</company>
    </header>
</contract>
```

【００５２】

次いで、ユーザは、データストア２０８のＡＰＩを使用して、新たな＜ｃｏｍｐａｎｙ＞ノード２１３を＜ｈｅａｄｅｒ＞の第１の子として追加すると仮定する（新たなノード２１３は、シングルクォーテーションマークで示されている）。

【００５３】

【表４】

```
<contract>
    <header>
        '<company>Fabrikam Corporation</company>'
        <company>Microsoft Corporation</company>
        <company>Contoso Corporation</company>
    </header>
</contract>
```

【００５４】

コンテンツ領域２０７上に結果として作成されるバインディング２０５は、依然として同じＸＰａｔｈ（「/contract(1)/header(1)/company(1)」）にバインドされており、したがってドキュメントのコンテンツはすぐに更新されて、次のようにそのノード２１３の新たなコンテンツが表示される。

"Fabrikam Corporation is located at One Microsoft Way."

【００５５】

本発明によれば、ワードプロセッサドキュメント２０４の１つまたは複数の領域が、データがバインドされたコンテンツ領域２０７を含む場合、ドキュメント２０４は、リンクされたコンテンツのいずれのソースに対する変更にも対応する。したがって、ドキュメント２０４のある範囲にデータがバインドされている場合、関連付けられているＸＭＬデータ２０９内のＸＭＬノード２１３のコンテンツを変更すると、コンテンツ領域２０７のテキストが自動的に変更される結果となる。これと同様に、ドキュメント２０４のある範囲にデータがバインドされている場合、ドキュメント２０４内のそのバインドされたコンテンツ領域２０７のテキストを変更すると、対応するＸＭＬデータ２０９内のＸＭＬノード２１３のコンテンツが自動的に変更される結果となる。すなわち、同一のバインディング２０５を有する複数のコンテンツ領域２０７が、ドキュメント２０４内の複数の場所に存在することができる。たとえば、ある名前に対するデータバインディング２０５を有するコンテンツ領域２０７を、ドキュメント２０４のヘッダならびに本文に追加することができる。これらの位置のいずれを変更しても、そのテキストはＸＭＬデータストア２０８と同期化され、ドキュメント２０４内においてノード２１３に対するデータバインディング２０５を有するコンテンツ領域２０７が存在するすべての場所で、その変更が反映されることになる。

【００５６】

ＸＭＬデータ２０９内のＸＭＬノード２１３は、ドキュメント２０４に対して一対多の関係を有することができ、つまりＸＭＬデータ２０９内の同じＸＭＬノード２１３を複数のデータバインディング２０５によって参照することができる。ドキュメント２０４内の、データがバインドされたコンテンツ領域２０７が更新されると常に、ＸＭＬデータ２０９内の該当するＸＭＬノード２１３に対する変更が行われ、ドキュメント２０４内の他のコンテンツ領域２０７内における他のすべての関連付けられたバインディング２０５が、その新たなテキストによって更新される。たとえば、ドキュメント２０４のヘッダ内のコンテンツ領域２０７が、いくつかのＸＭＬデータ２０９内の＜ｔｉｔｌｅ／＞ノード用のＸＰａｔｈ式を指定するデータバインディング２０５を含み、ドキュメント２０４の本文内の別のコンテンツ領域２０７も、その同じ要素に対するデータバインディング２０５を含むと仮定する。本発明によれば、双方は、たとえ異なるフォーマット設定を有する可能性があるとしても、同じコンテンツを表示することになる。ユーザが、ドキュメント２０４の本文内のコンテンツ領域２０７内のコンテンツを編集すると、その更新は、データス

トア２０８内の該当するＸＭＬデータ２０９内の該当するＸＭＬノード２１３に対して保持され、そのＸＭＬノード２１３を指定するドキュメント２０４内の関連付けられたバインディング２０５を有する他のすべてのコンテンツ領域２０７（たとえばヘッダ内、フッタ内など）が更新される。本発明の実施形態は、ドキュメント内の複数の場所をデータストア２０８内の単一のＸＭＬノード２１３にバインドし、次いでそれらの場所の３つすべてのコンテンツをデータの単一のソースにリンクするためのメカニズムを提供する。したがって、ＸＭＬデータ２０９内の同じノード２１１にリンクされている、ドキュメント２０４内のすべてのコンテンツ領域２０７のコンテンツは同じである。

【００５７】

この具体的な一例が、典型的なレポートドキュメントであり、ユーザは通常、複数の場所にタイトルを表示させることができる。たとえば、（大きなボールドのテキストで）表紙上に、（小さなテキストで）ヘッダ内に、また（小さなイタリックのテキストで）フッタ内になどである。通常ユーザは、それぞれの場所にタイトルを打ち込み、これら３つの場所のいずれか１つでタイトルが変更された場合、（コンテンツの整合性を保つために）必ず他の２つの場所でも忘れずにタイトルを変更しなければならない。しかし、これらの場所の３つすべての整合性を保つことは、あまりにも容易に忘れられてしまう。本発明の一実施形態によれば、ユーザがドキュメント２０４内に表示したいＸＭＬデータ２０９を含むデータストア２０８が配置されると、ドキュメント内の複数の場所（たとえば上記の３つの場所）はすべて、データストア２０８内の単一のＸＭＬノード２１３にデータをバインドされたコンテンツ領域２０７となることができる。

【００５８】

したがって、３つすべての場所のコンテンツは、データ２０９の単一のソースにリンクされ、またはバインドされている。つまり、ユーザがこれらの領域のいずれか１つ（たとえば表紙）のコンテンツを変更することによって、ユーザのテキストは、基礎をなすＸＭＬデータ２０９へ自動的に押し出され、次いで対応するデータバインディング２０５を伴うコンテンツ領域２０７を有するドキュメント２０４内の他の場所（この場合、ヘッダおよびフッタ）へ押し出される。つまりユーザは、ドキュメントとの対話の範囲で、コンテンツのこれら３つの範囲をリンクし、またはバインドしたことになり、これらはすべて同じになる。本発明の実施形態によれば、ドキュメントの領域は複数の方法（大きなボールドや、小さなイタリックなど）で表示することができるが、データストア２０８内のデータ構造は変更されない。

【００５９】

（未帰着参照（Dangling References））

本発明の一実施形態によれば、ユーザは、ターゲットを有さないＸＰａｔｈ式、すなわち自分の指定したターゲットＸＭＬノード２１３がデータストア２０８内のＸＭＬデータ２０９内に存在しないＸＰａｔｈ式を指定することもできる。データバインディング２０５は、その所望のターゲットノード２１３を「忘れる」ことはなく、むしろ「待機」状態へと移行し、いかなる特定のＸＭＬデータ２０９にも接続されずに、背後にあるＸＭＬデータストア２０８内のＸＭＬデータ２０９内に所望のノード２１３が現れるかどうかを注視しつつ待機する。これは、ドキュメントアセンブリシナリオにとって特に有用であり、このシナリオでは、ドキュメント２０４のそれぞれの部分（たとえば標準的な表紙、最終ページ、および再利用される節）が、これらの部分を単一の最終ドキュメント２０４へと組み立てる場合にのみ移植されるべきデータバインディング２０５を含むことができる。この場合、ドキュメントの作成者は、ドキュメントのそれぞれの「部分」におけるコンテンツ領域２０７内に、その部分に存在しなかったＸＭＬデータ２０９内のＸＭＬノード２１３へのデータバインディング２０５を指定する（たとえば表紙は、＜Ｔｉｔｌｅ／＞ＸＭＬ要素および＜Ｄａｔｅ／＞ＸＭＬ要素に対するバインディング２０５を有するコンテンツ領域を含むことができる）。その部分がそのターゲットドキュメントの外側に表示されている場合、ＸＭＬデータ２０９が存在しないため、それらのバインディングは接続されていないが、その部分が、実際に所望のデータ２０９を含むドキュメントに追加される

と、すぐにデータバインディング２０５はデータ２０９へ接続（同期化）して、正しいコンテンツを表示し、これによってドキュメントの作成者は、バインディング２０５を指定して、データ２０９がまだ作成されていない場合でも、それらを保存することができる。

【００６０】

あるタイプの未帰着参照２１５は、コンテンツ領域２０７上のデータバインディング２０５が、リンクされたＸＭＬストリーム内のノード２１３に首尾よくリンクできない場合に（すなわちコンテンツ領域内でのバインディングの状態で）発生する。ノード２１３が、リンクされたＸＭＬストリームから置換／削除されると、結果として、１つまたは複数のデータバインディング２０５は未帰着参照２１５になる場合がある。データバインディング２０５が、そのＸＰａｔｈによって未帰着参照２１５を有する場合、ワードプロセッサアプリケーション２０２は、引き続きノード２１３用の最新の既知のＸＰａｔｈをデータバインディング２０５上に保存することが好ましい。これは、もはやＸＰａｔｈがいかなるノード２０９にも帰着できない場合に発生する可能性がある。データストア２０８が、いくつかのＸＭＬデータ２０９を更新した旨のメッセージをワードプロセッサドキュメント２０４に伝えるたびに、ワードプロセッサアプリケーション２０２は、いずれかの未帰着参照２１５が最新の更新によって帰着するかどうか（すなわちＸＰａｔｈが、ＸＭＬツリー内の有効なノード２１３を指し示しているかどうか）をチェックする。ワードプロセッサアプリケーション２０２によって未帰着参照が帰着する場合、データストア２０８のコンテンツは、その時点でデータバインディング２０５内にあるコンテンツに優先すること、すなわちデータバインディング２０５のコンテンツは、データストア２０８内におけるノード２１３内のコンテンツによって置換されることが好ましい。未帰着参照は、１つまたは複数のプログラミング環境２０６を介してアクセスできる簡単なＡＰＩ層を使用して公開されることが好ましい。

【００６１】

一例として、ワードプロセッサドキュメント２０４は次のパラグラフを含み、その中のＭｉｃｒｏｓｏｆｔ　Ｃｏｒｐｏｒａｔｉｏｎは、いくつかのＸＭＬデータ２０９内のＸＰａｔｈ／ｃｏｎｔｒａｃｔ／ｈｅａｄｅｒ／ｃｏｍｐａｎｙ（３）にデータがバインドされたプレーンテキストコンテンツ領域２０７に相当すると仮定する。

"Microsoft Corporation is located at One Microsoft Way."

【００６２】

対応するＸＭＬデータ２０９は、次のような体裁とすることができる（リンク先のノード２１３は、シングルクォーテーションマークで示されている）。

【００６３】

【表５】

```
<contract>
    <header>
        <company>Fabrikam Corporation</company>
        <company>Contoso Corporation</company>
        ‘<company>Microsoft Corporation</company>’
    </header>
</contract>
```

【００６４】

開発者などのユーザが、データストア２０８のＡＰＩを使用して、＜ｈｅａｄｅｒ＞の下の第１の＜ｃｏｍｐａｎｙ＞ノード２１３を削除する場合は、次のようになる（ノード２１３は、シングルクォーテーションマークで示されている）。

【００６５】

【表６】

```
<contract>
    <header>
        ‘<company>Fabrikam Corporation</company>’
        <company>Contoso Corporation</company>
        <company>Microsoft Corporation</company>
    </header>
</contract>
```

【００６６】

ドキュメント２０４内のコンテンツ領域２０７上に結果として作成されるデータバインディング２０５は、同じＸＰａｔｈへのリンクを保持し、したがってデータバインディング２０５は、もはや存在しない＼ｃｏｎｔｒａｃｔ＼ｈｅａｄｅｒ＼ｃｏｍｐａｎｙ（３）に対する未帰着参照２１５となる。

【００６７】

【表７】

```
<contract>
    <header>
        <company>Ford Corporation</company>
        <company>Intel Corporation</company>
    </header>
</contract>
```

【００６８】

つまり、内部では途切れたリンクを有することになるが、本発明によれば、コンテンツ領域２０７のコンテンツは変更されず、またエラーも発生しない。すなわち、次のように表示される。

"Microsoft Corporation is located at One Microsoft Way."

【００６９】

いくつかのＸＭＬデータ２０９が置換または削除されると（またはリンクが、あるドキュメントから別のドキュメントへ移動すると）、そのＸＭＬデータ２０９を参照するすべてのデータバインディング２０５はすぐに、削除されたＸＭＬデータ２０９を指し示す未帰着参照２１５になる。データバインディング２０５が未帰着参照２１５を含む場合、ワードプロセッサアプリケーション２０２は、引き続きデータバインディング２０５に関連付けられた最新の既知のＸＰａｔｈ／ネームスペースのリンクを保存する。本発明の一実施形態によれば、データバインディング２０５のセットが未帰着参照２１５になると、ワードプロセッサアプリケーション２０２は、関連付けられたＸＭＬデータストア２０８内の他の任意の利用可能なＸＭＬデータ２０９にこれらのリンクを再び連結しようと試みる。これらのデータバインディング２０５のいずれかが、実際に別のＸＭＬデータ２０９内のノード２１３に帰着した場合、未帰着参照２１５のすべては、このＸＭＬデータ２０９に関連付けられ、その時点ではデータバインディング２０５が接続されている関連付けられたコンテンツ領域２０７を更新する。このＸＭＬデータ２０９が、結果的に未帰着参照２１５のいずれにとっても有効なデータバインディング２０５に結び付かない場合、ワードプロセッサアプリケーション２０２は、データストア２０８などにおけるそれぞれのＸＭＬデータ２０９に対して同様のチェックを実行する。ＸＭＬデータ２０９のいずれも未帰着参照２１５用として使用できない場合、これらのバインディングは、引き続き元のＸ

ＭＬデータ２０９に対する未帰着参照２１５として残る。

【００７０】

図３に示されている流れ図を参照しつつ、引き続き図２を参照して、本発明の一実施形態について説明する。図３に示されているプロセス３００は、ユーザがワードプロセッシングアプリケーション２０２を使用してワードプロセッサドキュメント２０４を開いたときに、３０２において開始する。３０４では、ワードプロセッシングアプリケーションはデータストア２０８を作成し、次いでこれには、３１０において、ワードプロセッサドキュメント２０４内に保存された、あるいはユーザインターフェースまたはプログラミングウィンドウ２０６を使用することによって要求された、任意のＸＭＬデータ２０９が移植される。データストア２０８は、ドキュメント２０４の一部として含まれ、ドキュメント編集表面２０３上には表示されないことが好ましい。データストア２０８は、コンテンツ領域２０７およびデータバインディング２０５を作成する前にロードできることが理解できるであろう。同様にコンテンツ領域２０７は、データストア２０８の前に作成することができる。すなわち、図３に示されているさまざまなオペレーションは、何らかの特定の順序で実行する必要はなく、ユーザ独自の好みに従って実施することができる。

【００７１】

３０６において、ユーザは、ドキュメント２０４の表面２０３上に存在する１つまたは複数のコンテンツ領域２０７を作成する。これらのコンテンツ領域は、ドキュメント２０４の既存のコンテンツから読み込むこともできるという点に留意されたい。３０８では、ユーザは、特定のリンクされたＸＭＬデータ２０９と、ターゲットノード２１３を指定するＸＰａｔｈ式とを提供することによって、データバインディング情報をコンテンツ領域２０７に関連付けることができる。１つまたは複数のデータバインディングは、データストア２０８内のＸＭＬデータ２０９の１つまたは複数のノード２１３を１つまたは複数のコンテンツ領域２０７へリンクする。データバインディング２０５は、バインドされるか、または未帰着となる。前述のように、各ノード２１３は、複数のコンテンツ領域２０７にバインドすることができ、各コンテンツ領域は、同じＸＭＬノード２１３へのデータバインディング２０５を指定する。さらにデータバインディング２０５は、複数のデータストア２０８へリンクすることができる。３０９では、ユーザは、コンテンツ領域２０７またはデータストア２０８内にＸＭＬデータを作成することができる。３１０では、ワードプロセッサアプリケーション２０２は、すべてのＸＭＬデータをデータストア２０８にロードする。３１２では、ワードプロセッサアプリケーション２０２は、ドキュメント２０４から、または３０６においてユーザによって要求されたようにコンテンツ領域２０７をロードし、３１４では、ワードプロセッサアプリケーション２０２は、ドキュメント２０４から、または３０８においてユーザによって要求されたようにデータバインディング２０５をロードする。３１６では、ワードプロセッサアプリケーション２０２は、特定のデータバインディング２０５によって指定されたノード２１３に関連付けられたＸＭＬデータ２０９が存在するかどうかをチェックする。

【００７２】

ＸＭＬデータ２０９が存在しない場合、３１８において、ワードプロセッサアプリケーション２０２は、他のＸＭＬデータ２０９が同じＸＭＬネームスペース内に存在するかどうかを判定する。３１６において、ＸＭＬデータ２０９がデータストア内で見つかった場合、３２０において、ワードプロセッサアプリケーション２０２は、指定されたＸＰａｔｈ用の関連付けられたＸＭＬノード２１３が存在するかどうかを判定する。ＸＰａｔｈが存在する場合、３２２において、ワードプロセッサアプリケーション２０２は、さまざまなドキュメントコンテンツ、すなわちコンテンツ領域２０７および他の任意のリンクされたコンテンツを、データバインディングを介して１つまたは複数の関連付けられたＸＭＬノード２１３に接続する。３２０においてＸＰａｔｈが見つからない場合、ワードプロセッサアプリケーション２０２は、３２４において、特定のデータバインディング２０５に未帰着参照２１５というラベルを付ける（未帰着状態と入力する）。３１８において、他のＸＭＬデータ２０９が、データストア２０８内の同じＸＭＬネームスペース内で見つか

った場合、３２６において、ワードプロセッサアプリケーション２０２は、ＸＭＬノード２１３がそのデータ内に存在するかどうかを再びチェックする。

【００７３】

ＸＭＬデータ２０９が存在しない場合、ワードプロセッサアプリケーション２０２は、３２４において、特定のデータバインディング２０５に未帰着参照というラベルを付ける（未帰着状態と入力する）。３２６においてＸＭＬデータ２０９が存在していると分かった場合、３２０において、ワードプロセッサアプリケーション２０２は、そのＸＭＬデータ２０９内の所望のＸＰａｔｈを検索する。３２０においてノード２１３が見つかった場合、ワードプロセッサアプリケーション２０２は、３２２において、ドキュメントコンテンツ、すなわちコンテンツ領域２０７および他の任意のリンクされたコンテンツを１つまたは複数の関連付けられたＸＭＬノード２１３に接続する。

【００７４】

本発明のさまざまな実施形態の論理的オペレーションは、（１）コンピュータによって実施される一連の行為や、コンピューティングシステム上で実行されるプログラムモジュールとして、および／または（２）コンピューティングシステム内で相互に接続された機械論理回路や回路モジュールとして実装されることが理解できるはずである。この実装は、本発明を実施するコンピューティングシステムのパフォーマンス要件に応じた選択の問題である。したがって、本明細書に記載されている本発明の実施形態を構成する論理的オペレーションは、オペレーション、構造的なデバイス、行為、あるいはモジュールなど、さまざまに呼ばれる。これらのオペレーション、構造的なデバイス、行為、およびモジュールは、本明細書で説明されている特許請求の範囲に記載された本発明の趣旨および範囲から逸脱することなく、ソフトウェア、ファームウェア、専用のデジタル論理、およびそれらの任意の組合せにおいて実装できることを当業者なら認識するであろう。

【００７５】

前述の仕様、例、およびデータは、本発明の構成の製造および使用に関する完全な説明を提供する。本発明の多くの実施形態は、本発明の趣旨および範囲から逸脱することなく作成することができるため、本発明は添付の特許請求の範囲に属するものである。

【図面の簡単な説明】

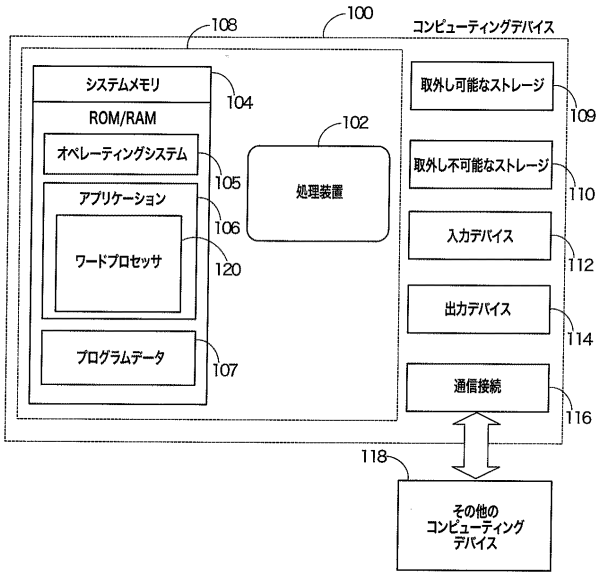
【００７６】

【図１】本発明の一例示的実施形態において使用できる例示的なコンピューティングデバイスを示す図である。

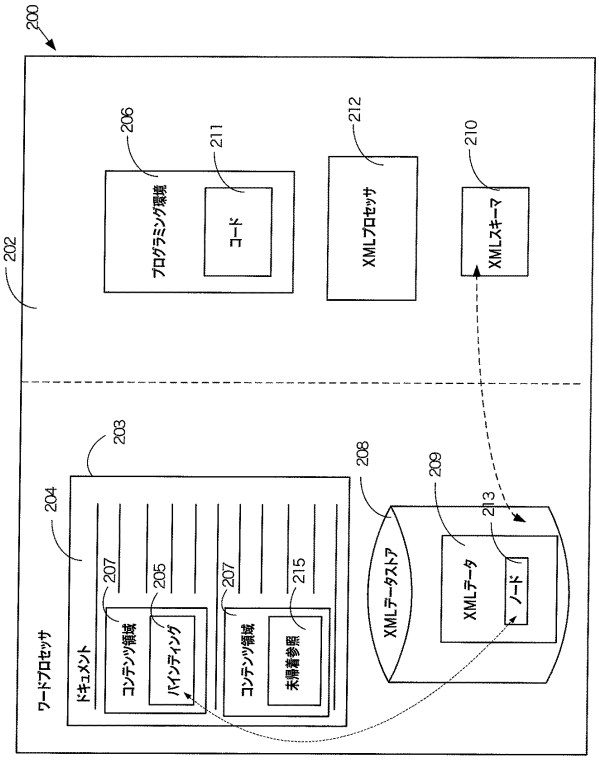
【図２】本発明を実施するための例示的な環境を示すブロック図である。

【図３】本発明の一実施形態による流れ図である。

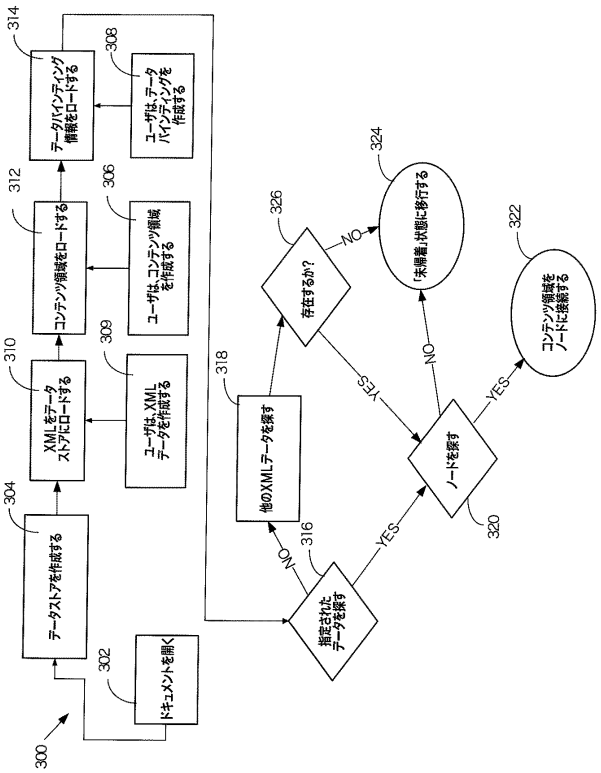
【図１】


100
コンピューティングデバイス
108
システムメモリ
104
ROM/RAM
オペレーティングシステム
105
アプリケーション
106
ワードプロセッサ
120
プログラムデータ
107
102
処理装置
取外し可能なストレージ
109
取外し不可能なストレージ
110
入力デバイス
112
出力デバイス
114
通信接続
116
118
その他の
コンピューティング
デバイス

【図２】


200
202
プログラミング環境
206
コード
211
XMLプロセッサ
212
XMLスキーマ
210
ワードプロセッサ
ドキュメント
204
203
コンテンツ領域
207
バインディング
205
コンテンツ領域
207
未解着参照
215
XMLデータストア
208
XMLデータ
209
ノード
213

【図３】


300
ドキュメントを開く
302
データストアを作成する
304
XMLをデータストアにロードする
310
ユーザは、XMLデータを作成する
309
コンテンツ領域をロードする
312
ユーザは、コンテンツ領域を作成する
306
データバインディング情報をロードする
314
ユーザは、データバインディングを作成する
308
指定されたデータを探す
316
他のXMLデータを探す
318
存在するか?
326
NO
YES
「未解着」状態に移行する
324
ノードを探す
320
NO
YES
コンテンツ領域をノードに接続する
322

フロントページの続き

(72)発明者 マーシン サウィッキー
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内
(72)発明者 ブライアン エム. ジョーンズ
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内
(72)発明者 ロバート エー. リトル
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内
(72)発明者 マーク サンダーランド
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内

審査官 岩田 淳

(56)参考文献 特開２０００－２２７９１４（ＪＰ，Ａ）
米国特許第０６０１４６７７（ＵＳ，Ａ）
米国特許出願公開第２００３／０００７００９（ＵＳ，Ａ１）
米国特許出願公開第２００４／００２１６７９（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １７／２１
ＪＳＴＰｌｕｓ（ＪＤｒｅａｍＩＩ）