(19)日本国特許庁(JP)

再公表特許(A1)

(11)国際公開番号
WO2006/051712

発行日　平成20年5月29日(2008.5.29)

(43)国際公開日　平成18年5月18日(2006.5.18)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 17/21	(2006.01)	G06F 17/21	564P	5B009
		G06F 17/21	501Z	5B109
		G06F 17/21	570R	

審査請求 有　予備審査請求 未請求　（全 52 頁）

出願番号	特願2006-544847 (P2006-544847)
(21)国際出願番号	PCT/JP2005/020028
(22)国際出願日	平成17年10月31日 (2005.10.31)
(31)優先権主張番号	特願2004-329844 (P2004-329844)
(32)優先日	平成16年11月12日 (2004.11.12)
(33)優先権主張国	日本国(JP)

(71)出願人　390024350
株式会社ジャストシステム
徳島県徳島市川内町平石若松１０８番地４
(74)代理人　100105924
弁理士　森下　賢樹
(74)代理人　100109047
弁理士　村田　雄祐
(72)発明者　大島　教雄
徳島県徳島市川内町平石若松１０８番地４
株式会社ジャストシステム内
(72)発明者　廣庭　雅一
徳島県徳島市川内町平石若松１０８番地４
株式会社ジャストシステム内

Fターム(参考)　5B009 QA06　RB31　SA14
5B109 QA06　RB31　SA14

最終頁に続く

(54)【発明の名称】文書処理装置及び文書処理方法

(57)【要約】
マークアップ言語により記述された文書にアノーテーションをつける技術を提供する。文書処理装置１００は、マークアップ言語により記述された文書を取得すると、その文書に含まれる、アノーテーションをつけるために定義された要素を検出し、その要素を、他の要素とは異なる形式で表示する。アノーテーションをつけるための要素は、アスタリスク（＊）、コロン（：）、ヌル（要素名なし）など、いずれのボキャブラリにも属さない要素として定義される。

【特許請求の範囲】
【請求項１】
マークアップ言語により記述された文書を取得する文書取得部と、
前記文書に含まれる、アノーテーションをつけるために定義された要素を検出する検出部と、
前記要素を検出したときに、前記要素を、他の要素とは異なる形式で表示する表示部と、
を備えることを特徴とする文書処理装置。
【請求項２】
前記要素は、処理系から無視される要素であることを特徴とする請求項１に記載の文書処理装置。
【請求項３】
前記要素は、ボキャブラリにおいて、意味を有しないことを定義された要素であることを特徴とする請求項１に記載の文書処理装置。
【請求項４】
前記要素の要素名は、アスタリスク（＊）、コロン（：）、ヌル（要素名なし）のいずれかであることを特徴とする請求項１に記載の文書処理装置。
【請求項５】
前記要素の処理方法を記述した定義ファイルを取得する定義ファイル取得部を更に備え、
前記表示部は、前記定義ファイルに記述された処理方法にしたがって、前記要素を表示することを特徴とする請求項１から４のいずれかに記載の文書処理装置。
【請求項６】
マークアップ言語により記述された文書を取得するステップと、
前記文書に含まれる、アノーテーションをつけるために定義された要素を検出するステップと、
前記要素を検出したときに、前記要素を、他の要素とは異なる形式で表示するステップと、
を含むことを特徴とする文書処理方法。
【請求項７】
マークアップ言語により記述された文書を取得する機能と、
前記文書に含まれる、アノーテーションをつけるために定義された要素を検出する機能と、
前記要素を検出したときに、前記要素を、他の要素とは異なる形式で表示する機能と、
をコンピュータに実現させることを特徴とする文書処理プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、文書処理技術に関し、特に、マークアップ言語により記述された文書を処理する文書処理装置及び文書処理方法に関する。
【背景技術】
【０００２】
ＸＭＬは、ネットワークなどを介して他者とデータを共有するのに適した形式として注目されており、ＸＭＬ文書を作成、表示、編集するためのアプリケーションが開発されている（たとえば、特許文献１参照）。ＸＭＬ文書は、文書型定義などにより定義されたボキャブラリ（タグセット）に基づいて作成されている。
【特許文献１】特開２００１－２９０８０４号公報
【発明の開示】
【発明が解決しようとする課題】
【０００３】

ボキャブラリは、任意に定義することが許されており、理論上、無限に多くのボキャブラリが存在しうる。これらのボキャブラリの全てに対応して専用の表示・編集環境を提供するのは現実的ではない。従来、専用の編集環境が用意されていないボキャブラリにより記述された文書を編集する場合、テキストデータにより構成された文書のソースを直接テキストエディタなどで編集していた。

【０００４】

本発明はこうした状況に鑑みてなされたものであり、その目的は、マークアップ言語により構造化されたデータを処理する際の、ユーザの利便性を向上させる技術を提供することにある。

【課題を解決するための手段】

【０００５】

本発明のある態様は、文書処理装置に関する。この文書処理装置は、マークアップ言語により記述された文書を取得する文書取得部と、前記文書に含まれる、アノーテーションをつけるために定義された要素を検出する検出部と、前記要素を検出したときに、前記要素を、他の要素とは異なる形式で表示する表示部と、を備えることを特徴とする。

【０００６】

アノーテーションは、注釈であってもよいし、メモであってもよいし、その他、任意の文字列や図形などで表現されるものであってもよい。文書処理装置は、アノーテーションをつけるために定義された要素を、太字で表示したり、斜体で表示したりするなど、強調表示してもよい。このような構成により、文書に新たな情報を付加し、付加した情報を識別可能に表示させることができる。

【０００７】

前記要素は、処理系から無視される要素であってもよい。例えば、コメントやＰＩであってもよい。これにより、他の処理系でも処理できる形式を保ちつつ、文書にアノーテーションを付加することができる。前記要素は、ボキャブラリにおいて、意味を有しないことを定義された要素であってもよい。例えば、ＸＨＴＭＬの「span」要素であってもよい。この場合、大域属性を利用してアノーテーションをつけてもよい。前記要素の要素名は、アスタリスク（＊）、コロン（：）、ヌル（要素名なし）のいずれかであってもよい。その他、他の処理系ではエラーとなるが、この文書処理装置ではアノーテーションであると解釈されるような要素であってもよい。

【０００８】

文書処理装置は、前記要素の処理方法を記述した定義ファイルを取得する定義ファイル取得部を更に備えてもよく、前記表示する手段は、前記定義ファイルに記述された処理方法にしたがって、前記要素を表示してもよい。

【０００９】

なお、以上の構成要素の任意の組合せ、本発明の表現を方法、装置、システムなどの間で変換したものもまた、本発明の態様として有効である。

【発明の効果】

【００１０】

本発明によれば、マークアップ言語により構造化されたデータを処理する際の、ユーザの利便性を向上させることができる。

【図面の簡単な説明】

【００１１】

【図１】前提技術に係る文書処理装置の構成を示す図である。

【図２】処理対象となるＸＭＬ文書の例を示す図である。

【図３】図２に示したＸＭＬ文書をＨＴＭＬで記述された表にマッピングする例を示す図である。

【図４（ａ）】図２に示したＸＭＬ文書を図３に示した表にマッピングするための定義ファイルの例を示す図である。

【図４（ｂ）】図２に示したＸＭＬ文書を図３に示した表にマッピングするための定義フ

ァイルの例を示す図である。
【図５】図２に示した成績管理ボキャブラリで記述されたＸＭＬ文書を、図３に示した対応によりＨＴＭＬにマッピングして表示した画面の例を示す図である。
【図６】ユーザが定義ファイルを生成するために、定義ファイル生成部がユーザに提示するグラフィカルユーザインターフェースの例を示す図である。
【図７】定義ファイル生成部により生成された画面レイアウトの他の例を示す図である。
【図８】文書処理装置によるＸＭＬ文書の編集画面の一例を示す図である。
【図９】文書処理装置により編集されるＸＭＬ文書の他の例を示す図である。
【図１０】図９に示した文書を表示した画面の例を示す図である。
【図１１（ａ）】文書処理システムの基本構成を示す図である。
【図１１（ｂ）】文書処理システム全体のブロック図を示す図である。
【図１１（ｃ）】文書処理システム全体のブロック図を示す図である。
【図１２】文書管理部の詳細を示す図である。
【図１３】ボキャブラリコネクションサブシステムの詳細を示す図である。
【図１４】プログラム起動部と他の構成の関係の詳細を示す図である。
【図１５】プログラム起動部によりロードされたアプリケーションサービスの構造の詳細を示す図である。
【図１６】コアコンポーネントの詳細を示す図である。
【図１７】文書管理部の詳細を示す図である。
【図１８】アンドゥフレームワークとアンドゥコマンドの詳細を示す図である。
【図１９】文書処理システムにおいて文書がロードされる様子を示す図である。
【図２０】文書とその表現の例を示す図である。
【図２１】モデルとコントローラの関係を示す図である。
【図２２】プラグインサブシステム、ボキャブラリコネクション、及びコネクタの詳細を示す図である。
【図２３】ＶＣＤファイルの例を示す図である。
【図２４】文書処理システムにおいて複合文書をロードする手順を示す図である。
【図２５】文書処理システムにおいて複合文書をロードする手順を示す図である。
【図２６】文書処理システムにおいて複合文書をロードする手順を示す図である。
【図２７】文書処理システムにおいて複合文書をロードする手順を示す図である。
【図２８】文書処理システムにおいて複合文書をロードする手順を示す図である。
【図２９】コマンドの流れを示す図である。
【図３０】実施の形態に係る文書処理装置の構成を示す図である。
【図３１】文書処理装置により処理される文書の例を示す図である。
【図３２】図３１に示した文書を文書処理装置により表示した画面の例を示す図である。
【図３３】文書処理装置により処理される文書の例を示す図である。
【図３４】実施の形態に係る文書処理装置の構成を示す図である。
【図３５】文書処理装置により処理される文書の例を示す図である。
【図３６】文書処理装置により処理される文書の例を示す図である。
【図３７】図３６に示した文書に対応づけられたアノーテーション用ファイルの例を示す図である。
【図３８】図３８（ａ）（ｂ）（ｃ）は、それぞれ、文書が表示されたレイヤー、アノーテーションレイヤー、それらが重畳された表示画面の例を示す。
【図３９】ユーザが個別にウェブページにアノーテーションをつける場合の構成例を示す図である。
【図４０】複数のユーザがウェブページのアノーテーションを共有する場合の構成例を示す図である。
【符号の説明】
【００１２】
２０　文書処理装置、２２　主制御ユニット、２４　編集ユニット、３０　ＤＯＭユニ

ット、３２　ＤＯＭ提供部、３４　ＤＯＭ生成部、３６　出力部、４０　ＣＳＳユニット、４２　ＣＳＳ解析部、４４　ＣＳＳ提供部、４６　レンダリング部、５０　ＨＴＭＬユニット、５２，６２　制御部、５４，６４　編集部、５６，６６　表示部、６０　ＳＶＧユニット、７０　アノーテーションユニット、７２　アノーテーション検出部、７４　アノーテーション表示部、７６　アノーテーション付加部、７８　取得部、８０　ＶＣユニット、８２　マッピング部、８４　定義ファイル取得部、８６　定義ファイル生成部、８８　定義ファイル合成部、１００　文書処理装置。

【発明を実施するための最良の形態】

【００１３】

以下、本発明の前提となる技術の説明を行った上で、実施の形態の詳細を説明する。

【００１４】

（前提技術）

図１は、前提技術に係る文書処理装置２０の構成を示す。文書処理装置２０は、文書内のデータが階層構造を有する複数の構成要素に分類された構造化文書を処理するが、本前提技術では構造化文書の一例としてＸＭＬ文書を処理する例について説明する。文書処理装置２０は、主制御ユニット２２、編集ユニット２４、ＤＯＭユニット３０、ＣＳＳユニット４０、ＨＴＭＬユニット５０、ＳＶＧユニット６０、及び変換部の一例であるＶＣユニット８０を備える。これらの構成は、ハードウエアコンポーネントでいえば、任意のコンピュータのＣＰＵ、メモリ、メモリにロードされたプログラムなどによって実現されるが、ここではそれらの連携によって実現される機能ブロックを描いている。したがって、これらの機能ブロックがハードウエアのみ、ソフトウエアのみ、またはそれらの組合せによっていろいろな形で実現できることは、当業者には理解されるところである。

【００１５】

主制御ユニット２２は、プラグインのロードや、コマンド実行のフレームワークを提供する。編集ユニット２４は、ＸＭＬ文書を編集するためのフレームワークを提供する。文書処理装置２０における文書の表示及び編集機能は、プラグインにより実現されており、文書の種別に応じて必要なプラグインが主制御ユニット２２又は編集ユニット２４によりロードされる。主制御ユニット２２又は編集ユニット２４は、処理対象となるＸＭＬ文書の名前空間を参照して、ＸＭＬ文書がいずれのボキャブラリにより記述されているかを判別し、そのボキャブラリに対応した表示又は編集用のプラグインをロードして表示や編集を実行させる。例えば、文書処理装置２０には、ＨＴＭＬ文書の表示及び編集を行うＨＴＭＬユニット５０、ＳＶＧ文書の表示及び編集を行うＳＶＧユニット６０など、ボキャブラリ（タグセット）ごとに表示系及び編集系がプラグインとして実装されており、ＨＴＭＬ文書を編集するときはＨＴＭＬユニット５０が、ＳＶＧ文書を編集するときはＳＶＧユニット６０が、それぞれロードされる。後述するように、ＨＴＭＬとＳＶＧの双方の構成要素を含む複合文書が処理対象となっている場合は、ＨＴＭＬユニット５０とＳＶＧユニット６０の双方がロードされる。

【００１６】

このような構成によれば、ユーザは、必要な機能のみを選択してインストールし、後から適宜機能を追加又は削除することができるので、プログラムを格納するハードディスクなどの記録媒体の記憶領域を有効に活用することができ、また、プログラム実行時にも、メモリの浪費を防ぐことができる。また、機能拡張性に優れており、開発主体としても、プラグインの形で新たなボキャブラリに対応することが可能なので開発が容易となり、ユーザとしても、プラグインの追加により容易かつ低コストにて機能を追加することができる。

【００１７】

編集ユニット２４は、ユーザインターフェースを介してユーザから編集指示のイベントを受け付け、そのイベントを適切なプラグインなどに通知するとともに、イベントの再実行（リドゥ）又は実行の取消（アンドゥ）などの処理を制御する。

【００１８】

ＤＯＭユニット３０は、ＤＯＭ提供部３２、ＤＯＭ生成部３４、及び出力部３６を含み、ＸＭＬ文書をデータとして扱うときのアクセス方法を提供するために定められた文書オブジェクトモデル（Document Object Model：ＤＯＭ）に準拠した機能を実現する。ＤＯＭ提供部３２は、編集ユニット２４に定義されているインタフェースを満たすＤＯＭの実装である。ＤＯＭ生成部３４は、ＸＭＬ文書からＤＯＭツリーを生成する。後述するように、処理対象となるＸＭＬ文書が、ＶＣユニット８０により他のボキャブラリにマッピングされる場合は、マッピング元のＸＭＬ文書に対応するソースツリーと、マッピング先のＸＭＬ文書に対応するデスティネーションツリーが生成される。出力部３６は、例えば編集終了時に、ＤＯＭツリーをＸＭＬ文書として出力する。

【００１９】

ＣＳＳユニット４０は、ＣＳＳ解析部４２、ＣＳＳ提供部４４、及びレンダリング部４６を含み、ＣＳＳに準拠した表示機能を提供する。ＣＳＳ解析部４２は、ＣＳＳの構文を解析するパーサの機能を有する。ＣＳＳ提供部４４は、ＣＳＳオブジェクトの実装であり、ＤＯＭツリーに対してＣＳＳのカスケード処理を行う。レンダリング部４６は、ＣＳＳのレンダリングエンジンであり、ＣＳＳを用いてレイアウトされるＨＴＭＬなどのボキャブラリで記述された文書の表示に用いられる。

【００２０】

ＨＴＭＬユニット５０は、ＨＴＭＬにより記述された文書を表示又は編集する。ＳＶＧユニット６０は、ＳＶＧにより記述された文書を表示又は編集する。これらの表示／編集系は、プラグインの形で実現されており、それぞれ、文書を表示する表示部（Canvas）５６、６６、編集指示を含むイベントを送受信する制御部（Editlet）５２、６２、編集コマンドを受けてＤＯＭに対して編集を行う編集部（Zone）５４、６４を備える。制御部５２又は６２が外部からＤＯＭツリーの編集コマンドを受け付けると、編集部５４又は６４がＤＯＭツリーを変更し、表示部５６又は６６が表示を更新する。これらは、ＭＶＣ（Model-View-Controller）と呼ばれるフレームワークに類似する構成をとっており、概ね、表示部５６及び６６が「View」に、制御部５２及び６２が「Controller」に、編集部５４及び６４とＤＯＭの実体が「Model」に、それぞれ対応する。本前提技術の文書処理装置２０では、ＸＭＬ文書をツリー表示形式で編集するだけでなく、それぞれのボキャブラリに応じた編集を可能とする。例えば、ＨＴＭＬユニット５０は、ＨＴＭＬ文書をワードプロセッサに類似した方式で編集するためのユーザインターフェースを提供し、ＳＶＧユニット６０は、ＳＶＧ文書を画像描画ツールに類似した方式で編集するためのユーザインターフェースを提供する。

【００２１】

ＶＣユニット８０は、マッピング部８２、定義ファイル取得部８４、及び定義ファイル生成部８６を含み、あるボキャブラリにより記述された文書を、他のボキャブラリにマッピングすることにより、マッピング先のボキャブラリに対応した表示編集用プラグインで文書を表示又は編集するためのフレームワークを提供する。本前提技術では、この機能を、ボキャブラリコネクション（Vocabulary Connection：ＶＣ）と呼ぶ。定義ファイル取得部８４は、マッピングの定義を記述したスクリプトファイルを取得する。この定義ファイルは、ノードごとに、ノード間の対応（コネクション）を記述する。このとき、各ノードの要素値や属性値の編集の可否を指定してもよい。また、ノードの要素値や属性値を用いた演算式を記述してもよい。これらの機能については、後で詳述する。マッピング部８２は、定義ファイル取得部８４が取得したスクリプトファイルを参照して、ＤＯＭ生成部３４にデスティネーションツリーを生成させ、ソースツリーとデスティネーションツリーの対応関係を管理する。定義ファイル生成部８６は、ユーザが定義ファイルを生成するためのグラフィカルユーザインターフェースを提供する。

【００２２】

ＶＣユニット８０は、ソースツリーとデスティネーションツリーの間のコネクションを監視し、表示を担当するプラグインにより提供されるユーザインタフェースを介してユーザから編集指示を受け付けると、まずソースツリーの該当するノードを変更する。ＤＯＭ

ユニット３０が、ソースツリーが変更された旨のミューテーションイベントを発行すると、ＶＣユニット８０は、そのミューテーションイベントを受けて、ソースツリーの変更にデスティネーションツリーを同期させるべく、変更されたノードに対応するデスティネーションツリーのノードを変更する。デスティネーションツリーを表示／編集するプラグイン、例えばＨＴＭＬユニット５０は、デスティネーションツリーが変更された旨のミューテーションイベントを受けて、変更されたデスティネーションツリーを参照して表示を更新する。このような構成により、少数のユーザにより利用されるローカルなボキャブラリにより記述された文書であっても、他のメジャーなボキャブラリに変換することで、文書を表示することができるとともに、編集環境が提供される。

【００２３】

文書処理装置２０により文書を表示又は編集する動作について説明する。文書処理装置２０が処理対象となる文書を読み込むと、ＤＯＭ生成部３４が、そのＸＭＬ文書からＤＯＭツリーを生成する。また、主制御ユニット２２又は編集ユニット２４は、名前空間を参照して文書を記述しているボキャブラリを判別する。そのボキャブラリに対応したプラグインが文書処理装置２０にインストールされている場合は、そのプラグインをロードして、文書を表示／編集させる。プラグインがインストールされていない場合は、マッピングの定義ファイルが存在するか否かを確認する。定義ファイルが存在する場合、定義ファイル取得部８４が定義ファイルを取得し、その定義に従って、デスティネーションツリーが生成され、マッピング先のボキャブラリに対応するプラグインにより文書が表示／編集される。複数のボキャブラリを含む複合文書である場合は、後述するように、それぞれのボキャブラリに対応したプラグインにより、文書の該当箇所がそれぞれ表示／編集される。定義ファイルが存在しない場合は、文書のソース又はツリー構造を表示し、その表示画面において編集が行われる。

【００２４】

図２は、処理対象となるＸＭＬ文書の例を示す。このＸＭＬ文書は、生徒の成績データを管理するために用いられる。ＸＭＬ文書のトップノードである構成要素「成績」は、配下に、生徒ごとに設けられた構成要素「生徒」を複数有する。構成要素「生徒」は、属性値「名前」と、子要素「国語」、「数学」、「理科」、「社会」を有する。属性値「名前」は、生徒の名前を格納する。構成要素「国語」、「数学」、「理科」、「社会」は、それぞれ、国語、数学、理科、社会の成績を格納する。例えば、名前が「Ａ」である生徒の国語の成績は「９０」、数学の成績は「５０」、理科の成績は「７５」、社会の成績は「６０」である。以下、この文書で使用されているボキャブラリ（タグセット）を、「成績管理ボキャブラリ」と呼ぶ。

【００２５】

本前提技術の文書処理装置２０は、成績管理ボキャブラリの表示／編集に対応したプラグインを有しないので、この文書をソース表示、ツリー表示以外の方法で表示するためには、前述したＶＣ機能が用いられる。すなわち、成績管理ボキャブラリを、プラグインが用意された別のボキャブラリ、例えば、ＨＴＭＬやＳＶＧなどにマッピングするための定義ファイルを用意する必要がある。ユーザ自身が定義ファイルを作成するためのユーザインターフェースについては後述することにして、ここでは、既に定義ファイルが用意されているとして説明を進める。

【００２６】

図３は、図２に示したＸＭＬ文書をＨＴＭＬで記述された表にマッピングする例を示す。図３の例では、成績管理ボキャブラリの「生徒」ノードを、ＨＴＭＬにおける表（「TABLE」ノード）の行（「TR」ノード）に対応づけ、各行の第１列には属性値「名前」を、第２列には「国語」ノードの要素値を、第３列には「数学」ノードの要素値を、第４列には「理科」ノードの要素値を、第５列には「社会」ノードの要素値を、それぞれ対応付ける。これにより、図２に示したＸＭＬ文書を、ＨＴＭＬの表形式で表示することができる。また、これらの属性値及び要素値は、編集可能であることが指定されており、ユーザがＨＴＭＬによる表示画面上で、ＨＴＭＬユニット５０の編集機能により、これらの値を編

集することができる。第６列には、国語、数学、理科、社会の成績の加重平均を算出する演算式が指定されており、生徒の成績の平均点が表示される。このように、定義ファイルに演算式を指定可能とすることにより、より柔軟な表示が可能となり、編集時のユーザの利便性を向上させることができる。なお、第６列は、編集不可であることが指定されており、平均点のみを個別に編集することができないようにしている。このように、マッピング定義において、編集の可否を指定可能とすることにより、ユーザの誤操作を防ぐことができる。

【００２７】

図４（ａ）及び図４（ｂ）は、図２に示したＸＭＬ文書を図３に示した表にマッピングするための定義ファイルの例を示す。この定義ファイルは、定義ファイル用に定義されたスクリプト言語により記述される。定義ファイルには、コマンドの定義と、表示のテンプレートが記述されている。図４（ａ）（ｂ）の例では、コマンドとして、「生徒の追加」と「生徒の削除」が定義されており、それぞれ、ソースツリーにノード「生徒」を挿入する操作と、ソースツリーからノード「生徒」を削除する操作が対応付けられている。また、テンプレートとして、表の第１行に「名前」、「国語」などの見出しが表示され、第２行以降に、ノード「生徒」の内容が表示されることが記述されている。ノード「生徒」の内容を表示するテンプレート中、「text-of」と記述された項は「編集可能」であることを意味し、「value-of」と記述された項は「編集不可能」であることを意味する。また、ノード「生徒」の内容を表示する行のうち、第６列には、「(src:国語 + src:数学 + src:理科 + src:社会) div 4」という計算式が記述されており、生徒の成績の平均が表示されることを意味する。

【００２８】

図５は、図２に示した成績管理ボキャブラリで記述されたＸＭＬ文書を、図３に示した対応によりＨＴＭＬにマッピングして表示した画面の例を示す。表９０の各行には、左から、各生徒の名前、国語の成績、数学の成績、理科の成績、社会の成績、及び平均点が表示されている。ユーザは、この画面上で、ＸＭＬ文書を編集することができる。たとえば、第２行第３列の値を「７０」に変更すると、このノードに対応するソースツリーの要素値、すなわち、生徒「Ｂ」の数学の成績が「７０」に変更される。このとき、ＶＣユニット８０は、デスティネーションツリーをソースツリーに追従させるべく、デスティネーションツリーの該当箇所を変更し、ＨＴＭＬユニット５０が、変更されたデスティネーションツリーに基づいて表示を更新する。したがって、画面上の表においても、生徒「Ｂ」の数学の成績が「７０」に変更され、更に、平均点が「５５」に変更される。

【００２９】

図５に示した画面には、図４（ａ）（ｂ）に示した定義ファイルに定義されたように、「生徒の追加」及び「生徒の削除」のコマンドがメニューに表示される。ユーザがこれらのコマンドを選択すると、ソースツリーにおいて、ノード「生徒」が追加又は削除される。このように、本前提技術の文書処理装置２０では、階層構造の末端の構成要素の要素値を編集するのみではなく、階層構造を編集することも可能である。このようなツリー構造の編集機能は、コマンドの形でユーザに提供されてもよい。また、例えば、表の行を追加又は削除するコマンドが、ノード「生徒」を追加又は削除する操作に対応づけられてもよい。また、他のボキャブラリを埋め込むコマンドがユーザに提供されてもよい。この表を入力用テンプレートとして、穴埋め形式で新たな生徒の成績データを追加することもできる。以上のように、ＶＣ機能により、ＨＴＭＬユニット５０の表示／編集機能を利用しつつ、成績管理ボキャブラリで記述された文書を編集することが可能となる。

【００３０】

図６は、ユーザが定義ファイルを生成するために、定義ファイル生成部８６がユーザに提示するグラフィカルユーザインタフェースの例を示す。画面左側の領域９１には、マッピング元のＸＭＬ文書がツリー表示されている。画面右側の領域９２には、マッピング先のＸＭＬ文書の画面レイアウトが示されている。この画面レイアウトは、ＨＴＭＬユニット５０により編集可能となっており、ユーザは、画面右側の領域９２において、文書を表

示するための画面レイアウトを作成する。そして、例えば、マウスなどのポインティングデバイスにより、画面左側の領域９１に表示されたマッピング元のＸＭＬ文書のノードを、画面右側の領域９２に表示されたＨＴＭＬによる画面レイアウト中へドラッグ＆ドロップ操作を行うことにより、マッピング元のノードと、マッピング先のノードとのコネクションが指定される。例えば、要素「生徒」の子要素である「数学」を、ＨＴＭＬ画面の表９０の第１行第３列にドロップすると、「数学」ノードと、３列目の「ＴＤ」ノードの間にコネクションが張られる。各ノードには、編集の可否が指定できるようになっている。また、表示画面中には、演算式を埋め込むこともできる。画面の編集が終わると、定義ファイル生成部８６は、画面レイアウトとノード間のコネクションを記述した定義ファイルを生成する。

【００３１】

ＸＨＴＭＬ、ＭａｔｈＭＬ、ＳＶＧなどの主要なボキャブラリに対応したビューワやエディタは既に開発されているが、図２に示した文書のようなオリジナルなボキャブラリで記述された文書に対応したビューワやエディタを開発するのは現実的でない。しかし、上記のように、他のボキャブラリにマッピングするための定義ファイルを作成すれば、ビューワやエディタを開発しなくても、ＶＣ機能を利用して、オリジナルなボキャブラリで記述された文書を表示・編集することができる。

【００３２】

図７は、定義ファイル生成部８６により生成された画面レイアウトの他の例を示す。図７の例では、成績管理ボキャブラリで記述されたＸＭＬ文書を表示するための画面に、表９０と、円グラフ９３が作成されている。この円グラフ９３は、ＳＶＧにより記述される。後述するように、本前提技術の文書処理装置２０は、一つのＸＭＬ文書内に複数のボキャブラリを含む複合文書を処理することができるので、この例のように、ＨＴＭＬで記述された表９０と、ＳＶＧで記述された円グラフ９３とを、一つの画面上に表示することができる。

【００３３】

図８は、文書処理装置２０によるＸＭＬ文書の編集画面の一例を示す。図８の例では、一つの画面が複数に分割されており、それぞれの領域において、処理対象となるＸＭＬ文書を異なる複数の表示形式により表示している。領域９４には、文書のソースが表示されており、領域９５には、文書のツリー構造が表示されており、領域９６には、図５に示したＨＴＭＬにより記述された表が表示されている。これらのいずれの画面上においても、文書の編集が可能であり、いずれかの画面上でユーザが編集を行うと、ソースツリーが変更され、それぞれの画面の表示を担当するプラグインが、ソースツリーの変更を反映すべく画面を更新する。具体的には、ソースツリーの変更を通知するミューテーションイベントのリスナーとして、それぞれの編集画面の表示を担当するプラグインの表示部を登録しておき、いずれかのプラグイン又はＶＣユニット８０によりソースツリーが変更されたときに、編集画面を表示中の全ての表示部が、発行されたミューテーションイベントを受け取って画面を更新する。このとき、プラグインがＶＣ機能により表示を行っている場合は、ＶＣユニット８０がソースツリーの変更に追従してデスティネーションツリーを変更した後、変更されたデスティネーションツリーを参照してプラグインの表示部が画面を更新する。

【００３４】

例えば、ソース表示及びツリー表示を、専用のプラグインにより実現している場合は、ソース表示用プラグインとツリー表示用プラグインは、デスティネーションツリーを用いず、直接ソースツリーを参照して表示を行う。この場合、いずれかの画面において編集が行われると、ソース表示用プラグインとツリー表示用プラグインは、変更されたソースツリーを参照して画面を更新し、領域９６の画面を担当しているＨＴＭＬユニット５０は、ソースツリーの変更に追従して変更されたデスティネーションツリーを参照して画面を更新する。

【００３５】

ソース表示及びツリー表示は、ＶＣ機能を利用して実現することもできる。すなわち、ソース、ツリー構造をＨＴＭＬによりレイアウトし、そのＨＴＭＬにＸＭＬ文書をマッピングして、ＨＴＭＬユニット５０により表示してもよい。この場合、ソース形式、ツリー形式、表形式の３つのデスティネーションツリーが生成されることになる。いずれかの画面において編集が行われると、ＶＣユニット８０は、ソースツリーを変更した後、ソース形式、ツリー形式、表形式の３つのデスティネーションツリーをそれぞれ変更し、ＨＴＭＬユニット５０は、それらのデスティネーションツリーを参照して、３つの画面を更新する。

【００３６】

このように、一つの画面上に複数の表示形式で文書を表示することにより、ユーザの利便性を向上させることができる。例えば、ユーザは、ソース表示又はツリー表示により文書の階層構造を把握しつつ、表９０などを用いて視覚的に分かりやすい形式で文書を表示し、編集することができる。上記の例では、一つの画面を分割して複数の表示形式による画面を同時に表示したが、一つの画面に一つの表示形式による画面を表示し、表示形式をユーザの指示により切り替え可能としてもよい。この場合、主制御ユニット２２が、ユーザから表示形式の切り替え要求を受け付け、各プラグインに指示して表示を切り替える。

【００３７】

図９は、文書処理装置２０により編集されるＸＭＬ文書の他の例を示す。図９に示したＸＭＬ文書では、ＳＶＧ文書の「foreignObject」タグの中にＸＨＴＭＬ文書が埋め込まれており、さらに、ＸＨＴＭＬ文書の中にＭａｔｈＭＬで記述された数式が入っている。このような場合、編集ユニット２４が、名前空間を参照して、適切な表示系に描画作業を振り分ける。図９の例では、編集ユニット２４は、まず、ＳＶＧユニット６０に四角形を描画させ、つづいて、ＨＴＭＬユニット５０にＸＨＴＭＬ文書を描画させる。さらに、図示しないＭａｔｈＭＬユニットに、数式を描画させる。こうして、複数のボキャブラリを包含する複合文書が適切に表示される。表示結果を図１０に示す。

【００３８】

文書編集中、カーソル（キャリッジ）の位置に応じて、表示されるメニューを切り替えてもよい。すなわち、カーソルが、ＳＶＧ文書が表示された領域内に存在するときは、ＳＶＧユニット６０が提供するメニュー、又はＳＶＧ文書をマッピングするための定義ファイルに定義されたコマンドを表示し、カーソルが、ＸＨＴＭＬ文書が表示された領域内に存在するときは、ＨＴＭＬユニット５０が提供するメニュー、又はＸＨＴＭＬ文書をマッピングするための定義ファイルに定義されたコマンドを表示する。これにより、編集位置に応じて適切なユーザインターフェースを提供することができる。

【００３９】

複合文書において、あるボキャブラリに対応する適切なプラグイン又はマッピング定義ファイルがなかった場合は、そのボキャブラリにより記述された部分は、ソース表示又はツリー表示されてもよい。従来、ある文書に他の文書を埋め込んだ複合文書を開くとき、埋め込まれた文書を表示するアプリケーションがインストールされていないと、その内容を表示することができなかったが、本前提技術では、表示用のアプリケーションが存在しなくても、テキストデータにより構成されたＸＭＬ文書をソース表示又はツリー表示することにより内容を把握することができる。これは、テキストベースであるＸＭＬなどの文書ならではの特徴といえる。

【００４０】

データがテキストベースで記述されることの他の利点として、例えば、複合文書中の、あるボキャブラリにより記述される部分において、同一文書内の他のボキャブラリで記述された部分のデータを参照してもよい。また、文書内で検索を実行する時に、ＳＶＧなどの図に埋め込まれた文字列も検索対象とすることができる。

【００４１】

あるボキャブラリにより記述された文書内に、他のボキャブラリのタグを用いてもよい。このＸＭＬ文書は、妥当（valid）ではないが、整形式（well-formed）であれば、有効

なＸＭＬ文書として処理可能である。この場合、挿入された他のボキャブラリのタグは、定義ファイルによりマッピングされてもよい。例えば、ＸＨＴＭＬ文書中に、「重要」、「最重要」などのタグを使用し、これらのタグで囲まれた部分を強調表示してもよいし、重要度の順にソートして表示してもよい。

【００４２】

図１０に示した編集画面において、ユーザにより文書が編集されると、編集された部分を担当するプラグイン又はＶＣユニット８０がソースツリーを変更する。ソースツリーには、ノードごとにミューテーションイベントのリスナーを登録できるようになっており、通常は、各ノードが属するボキャブラリに対応したプラグインの表示部又はＶＣユニット８０がリスナーとして登録される。ＤＯＭ提供部３２は、ソースツリーが変更されると、変更されたノードから上位の階層へたどって、登録されたリスナーがあれば、そのリスナーへミューテーションイベントを発行する。例えば、図９に示した文書において、＜ｈｔｍｌ＞ノードの下位のノードが変更された場合、＜ｈｔｍｌ＞ノードにリスナーとして登録されたＨＴＭＬユニット５０にミューテーションイベントが通知されるとともに、その上位の＜ｓｖｇ＞ノードにリスナーとして登録されたＳＶＧユニット６０にもミューテーションイベントが通知される。このとき、ＨＴＭＬユニット５０は、変更されたソースツリーを参照して表示を更新する。ＳＶＧユニット６０は、自身のボキャブラリに属するノードが変更されていないので、ミューテーションイベントを無視してもよい。

【００４３】

編集の内容によっては、ＨＴＭＬユニット５０による表示の更新に伴って、全体のレイアウトが変わる可能性がある。この場合は、画面のレイアウトを管理する構成、例えば最上位のノードの表示を担当するプラグインにより、プラグインごとの表示領域のレイアウトが更新される。例えば、ＨＴＭＬユニット５０による表示領域が以前より大きくなった場合、ＨＴＭＬユニット５０は、まず自身の担当する部分を描画して、表示領域の大きさを決定する。そして、画面のレイアウトを管理する構成に、変更後の表示領域の大きさを通知し、レイアウトの更新を依頼する。画面のレイアウトを管理する構成は、通知を受けて、プラグインごとの表示領域を再レイアウトする。こうして、編集された部分の表示が適切に更新されるとともに、画面全体のレイアウトが更新される。

【００４４】

つづいて、前提技術の文書処理装置２０を実現する機能構成について更に詳細に説明する。以下の説明では、クラス名などを記載する際には、英字をそのまま用いて記載することにする。

【００４５】

Ａ．概要

インターネットの出現により、ユーザによって処理され管理される文書の数が、ほぼ指数関数的に増加してきた。インターネットの核を形成するウェブ（World Wide Web）は、そのような文書データの大きな受け皿となっている。ウェブは、文書に加えて、このような文書の情報検索システムを提供する。これらの文書は、通常、マークアップ言語により記述される。マークアップ言語のシンプルかつポピュラーな例の一つにＨＴＭＬ（HyperText Markup Language）がある。このような文書は、ウェブの他の位置に格納されている他の文書へのリンクをさらに含む。ＸＭＬ（eXtensible Markup Language）は、さらに高度でポピュラーなマークアップ言語である。ウェブ文書にアクセスし、閲覧するためのシンプルなブラウザが、Ｊａｖａ（登録商標）のようなオブジェクト指向のプログラミング言語で開発されている。

【００４６】

マークアップ言語により記述された文書は、通常、ブラウザや他のアプリケーションの中では、ツリーデータ構造の形で表現される。この構造は、文書を構文解析した結果のツリーに相当する。ＤＯＭ（Document Object Model）は、文書を表現し、操作するために使用される、よく知られたツリーベースのデータ構造モデルである。ＤＯＭは、ＨＴＭＬやＸＭＬ文書などを含む文書を表現するための標準的なオブジェクトのセットを提供する

。ＤＯＭは、文書内のコンポーネントを表現するオブジェクトがどのようにつながっているかという標準モデルと、それらのオブジェクトにアクセスしたり操作したりするための標準インタフェイスという、２つの基本的なコンポーネントを含む。

【００４７】

アプリケーション開発者は、独自のデータ構造やＡＰＩ（Application Program Interface）へのインタフェイスとしてＤＯＭをサポートすることができる。他方、文書を作成するアプリケーション開発者は、彼らのＡＰＩの独自インタフェイスではなく、ＤＯＭの標準インタフェイスを使用することができる。したがって、標準を提供するというその能力により、ＤＯＭは、様々な環境、特にウェブにおいて、文書の相互利用を促進させるために有効である。ＤＯＭのいくつかのバージョンが定義されており、異なるプログラミング環境及びアプリケーションによって使用されている。

【００４８】

ＤＯＭツリーは、対応するＤＯＭの内容に基づいた文書の階層的表現である。ＤＯＭツリーは「根（ルート）」、及びルートから発生する１つ以上の「節（ノード）」を含む。ルートが文書全体を表す場合もある。中間のノードは、例えば、テーブル及びそのテーブル中の行及び列のような要素を表すことができる。ＤＯＭツリーの「葉」は、通常、それ以上分解できないテキストや画像のようなデータを表す。ＤＯＭツリーの各ノードは、フォント、サイズ、色、インデントなど、ノードによって表される要素のパラメータを記述する属性に関連付けられてもよい。

【００４９】

ＨＴＭＬは、文書を作成するために一般に用いられる言語であるが、フォーマット及びレイアウト用の言語であり、データ記述のための言語ではない。ＨＴＭＬドキュメントを表現するＤＯＭツリーのノードは、ＨＴＭＬのフォーマッティングタグとして予め定義されたエレメントであって、通常、ＨＴＭＬは、データの詳述や、データのタギング／ラベリングのための機能を提供しないので、ＨＴＭＬドキュメント中のデータに対するクエリを定式化することは多くの場合困難である。

【００５０】

ネットワーク設計者たちの目指すものは、ウェブ上の文書がソフトウェアアプリケーションによってクエリされたり処理されたりできるようにすることである。表示方法とは無関係で、階層的に構造化された言語であれば、そのようにクエリされ処理されることができる。ＸＭＬ（eXtensible Markup Language）のようなマークアップ言語は、これらの特徴を提供することができる。

【００５１】

ＨＴＭＬとは逆に、ＸＭＬのよく知られた利点は、文書の設計者が自由に定義可能な「タグ」を使用して、データ要素にラベルを付けることが可能である点である。このようなデータ要素は、階層的に構造化することができる。さらに、ＸＭＬ文書は、文書内で用いられるタグ及びそれらの相互関係の「文法」を記述した文書型定義を含むことができる。構造化されたＸＭＬ文書の表示方法を定義するために、ＣＳＳ（Cascading Style Sheet）又はＸＳＬ（XML Style Language）が使用される。ＤＯＭ、ＨＴＭＬ、ＸＭＬ、ＣＳＳ、ＸＳＬ及び関連する言語の特徴に関する付加的な情報は、ウェブからも得ることができる。（例えば、http://www.w3.org/TR/）

【００５２】

Ｘｐａｔｈは、ＸＭＬ文書の部分の位置を指定するために共通のシンタックス及びセマンティクスを提供する。機能性の例として、ＸＭＬ文書に対応するＤＯＭツリーのトラバース（移動）がある。それは、ＸＭＬ文書の様々な表現に関連した文字列、数、及びブーリアン文字の操作のための基本的な機能を提供する。Ｘｐａｔｈは、ＸＭＬ文書の見た目のシンタックス、例えば、テキストとしてみたときに何行目であるとか何文字目であるとかといった文法ではなく、ＤＯＭツリーなどの抽象的・論理的な構造において動作する。Ｘｐａｔｈを使用することにより、例えばＸＭＬ文書のＤＯＭツリー内の階層的構造を通じて場所を指定することができる。アドレシングのための使用の他に、Ｘｐａｔｈは、Ｄ

ＯＭツリー中のノードがパターンにマッチするか否かをテストするために使用されるようにも設計されている。ＸＰａｔｈに関する更なる詳細は、http://www.w3.org/TR/xpathで得ることができる。

【００５３】

ＸＭＬの既知の利点及び特徴により、マークアップ言語（例えばＸＭＬ）で記述された文書を扱うことができ、文書を作成及び修正するためのユーザフレンドリーなインタフェイスを提供することができる、効果的な文書処理システムが求められる。

【００５４】

ここで説明されるシステムの構成のうちのいくつかは、ＭＶＣ（Model-View-Controller）と呼ばれる、よく知られたＧＵＩ（Graphical User Interface）パラダイムを用いて説明される。ＭＶＣパラダイムは、アプリケーション又はアプリケーションのインタフェイスの一部を、３つの部分、すなわち、モデル、ビュー、コントローラに分割する。ＭＶＣは、元は、ＧＵＩの世界に、従来の入力、処理、出力の役割を割り当てるために開発された。

［入力］ → ［処理］ → ［出力］

［コントローラ］→ ［モデル］ → ［ビュー］

【００５５】

ＭＶＣパラダイムによれば、外界のモデリング、ユーザへの視覚的なフィードバック、及びユーザの入力は、モデル（Ｍ）、ビュー（Ｖ）、及びコントローラ（Ｃ）オブジェクトにより分離されて扱われる。コントローラは、ユーザからのマウスとキーボード入力のような入力を解釈し、これらのユーザアクションを、適切な変更をもたらすためにモデル及び／又はビューに送られるコマンドにマップするように作用する。モデルは、１以上のデータ要素を管理するように作用し、その状態に関するクエリに応答し、状態を変更する指示に応答する。ビューは、ディスプレイの長方形の領域を管理するように作用し、グラフィクスとテキストの組合せによりユーザにデータを提示する機能を有する。

【００５６】

Ｂ．文書処理システムの全体構成

文書処理システムの実施例は、図１１－２９に関連して明らかにされる。

【００５７】

図１１（ａ）は、後述するタイプの文書処理システムの基礎として機能する要素の従来の構成例を示す。構成１０は、通信経路１３によりメモリ１２に接続されたＣＰＵ又はマイクロプロセッサ１１などの形式のプロセッサを含む。メモリ１２は、現在又は将来に利用可能な任意のＲＯＭ及び／又はＲＡＭの形式であってもよい。通信経路１３は、典型的にはバスとして設けられる。マウス、キーボード、音声認識システムなどのユーザ入力装置１４及び表示装置１５（又は他のユーザインタフェイス）に対する入出力インタフェイス１６も、プロセッサ１１とメモリ１２の通信のためのバスに接続される。この構成は、スタンドアロンであってもよいし、複数の端末及び１以上のサーバが接続されてネットワーク化された形式であってもよいし、既知のいかなる方式により構成されてもよい。本発明は、これらのコンポーネントの配置、集中又は分散されたアーキテクチャー、あるいは様々なコンポーネントの通信方法により制限されない。

【００５８】

さらに、本システム及びここで議論される実施例は、様々な機能性を提供するいくつかのコンポーネント及びサブコンポーネントを含むものとして議論される。これらのコンポーネント及びサブコンポーネントは、注目された機能性を提供するために、ハードウェアとソフトウェアの組合せだけでなく、ハードウェアのみ、ソフトウェアのみによっても実現されうる。さらに、ハードウェア、ソフトウェア、及びそれらの組合せは、汎用の計算装置、専用のハードウェア、又はそれらの組合せにより実現されうる。したがって、コンポーネント又はサブコンポーネントの構成は、コンポーネント又はサブコンポーネントの機能性を提供するための特定のソフトウェアを実行する汎用／専用の計算装置を含む。

【００５９】

図１１（ｂ）は、文書処理システムの一例の全体のブロック図を示す。このような文書処理システムにおいて文書が生成され編集される。これらの文書は、例えばＸＭＬなど、マークアップ言語の特徴を有する任意の言語により記述されてもよい。また、便宜上、特定のコンポーネント及びサブコンポーネントの用語及び表題を創造した。しかしながら、これらは、この開示の一般的な教示の範囲を制限するために解釈されるべきではない。

【００６０】

文書処理システムは、２つの基本的な構成を有するものととらえることができる。第１の構成は、文書処理システムが動作する環境である「実行環境」１０１である。例えば、実行環境は、文書の処理中及び管理中に、ユーザだけでなくシステムも支援する、基本的なユーティリティ及び機能を提供する。第２の構成は、実行環境において走るアプリケーションから構成される「アプリケーション」１０２である。これらのアプリケーションは、文書自身及び文書の様々な表現を含む。

【００６１】

１．実行環境

実行環境１０１のキーとなるコンポーネントはProgramInvoker（プログラムインボーカ：プログラム起動部）１０３である。ProgramInvoker１０３は、文書処理システムを起動するためにアクセスされる基本的なプログラムである。例えば、ユーザが文書処理システムにログオンして開始するとき、ProgramInvoker１０３が実行される。ProgramInvoker１０３は、例えば、文書処理システムにプラグインとして加えられた機能を読み出して実行させたり、アプリケーションを開始して実行させたり、文書に関連するプロパティを読み出すことができる。ProgramInvoker１０３の機能はこれらに限定されない。ユーザが実行環境内で実行されるように意図されたアプリケーションを起動したいとき、ProgramInvoker１０３は、そのアプリケーションを見つけ、それを起動して、アプリケーションを実行する。

【００６２】

ProgramInvoker１０３には、プラグインサブシステム１０４、コマンドサブシステム１０５、及びResource（リソース）モジュール１０９などのいくつかのコンポーネントがアタッチされている。これらの構成については、以下に詳述する。

【００６３】

ａ）プラグインサブシステム

プラグインサブシステム１０４は、文書処理システムに機能を追加するための高度に柔軟で効率的な構成として使用される。プラグインサブシステム１０４は、また、文書処理システムに存在する機能を修正又は削除するために使用することができる。さらに、種々様々の機能をプラグインサブシステムを使用して追加又は修正することができる。例えば、画面上への文書の描画を支援するように作用するEditlet（エディットレット：編集部）機能を追加することもできる。Editletプラグインは、システムに追加されるボキャブラリの編集も支援する。

【００６４】

プラグインサブシステム１０４は、ServiceBroker（サービスブローカ：サービス仲介部）１０４１を含む。ServiceBroker１０４１は、文書処理システムに加えられるプラグインを管理することにより、文書処理システムに加えられるサービスを仲介する。

【００６５】

所望の機能性を実現する個々の機能は、Service（サービス）１０４２の形でシステムに追加される。利用可能なService１０４２のタイプは、Application（アプリケーション）サービス、ZoneFactory（ゾーンファクトリ：ゾーン生成部）Service、Editlet（エディットレット：編集部）Service、CommandFactory（コマンドファクトリ：コマンド生成部）Service、ConnectXPath（コネクトＸＰａｔｈ：ＸＰａｔｈ管理部）Service、CSSComputation（ＣＳＳコンピューテーション：ＣＳＳ計算部）Serviceなどを含むが、これらに限定されない。これらのService、及びシステムの他の構成とそれらとの関係は、文書処理システムについてのよりよい理解のために、以下に詳述される。

【００６６】
プラグインとServiceの関係は以下の通りである。プラグインは、１以上のServiceProvider（サービスプロバイダ：サービス提供部）を含むことができるユニットである。それぞれのServiceProviderは、それに関連したServiceの１以上のクラスを有する。例えば、適切なソフトウェアアプリケーションを有する単一のプラグインを使用することにより、１以上のServiceをシステムに追加することができ、これにより、対応する機能をシステムに追加することができる。

【００６７】
ｂ）コマンドサブシステム
コマンドサブシステム１０５は、文書の処理に関連したコマンドの形式の命令を実行するために使用される。ユーザは、一連の命令を実行することにより、文書に対する操作を実行することができる。例えば、ユーザは、コマンドの形で命令を発行することにより、文書処理システム中のＸＭＬ文書に対応するＸＭＬのＤＯＭツリーを編集し、ＸＭＬ文書を処理する。これらのコマンドは、キーストローク、マウスクリック、又は他の有効なユーザインタフェイスアクションを使用して入力されてもよい。１つのコマンドにより１以上の命令が実行されることもある。この場合、これらの命令が１つのコマンドにラップ（包含）され、連続して実行される。例えば、ユーザが、誤った単語を正しい単語に置換したいとする。この場合、第１の命令は、文書中の誤った単語を発見することであり、第２の命令は、誤った単語を削除することであり、第３の命令は、正しい単語を挿入することであってもよい。これらの３つの命令が１つのコマンドにラップされてもよい。

【００６８】
コマンドは、関連した機能、例えば、後で詳述する「アンドゥ」機能を有してもよい。これらの機能は、オブジェクトを生成するために使用されるいくつかの基本クラスにも割り当てられてもよい。

【００６９】
コマンドサブシステム１０５のキーとなるコンポーネントは、選択的にコマンドを与え、実行するように作用するCommandInvoker（コマンドインボーカ：コマンド起動部）１０５１である。図１１（ｂ）には、１つのCommandInvokerのみが示されているが、１以上のCommandInvokerが使用されてもよく、１以上のコマンドが同時に実行されてもよい。CommandInvoker１０５１は、コマンドを実行するために必要な機能及びクラスを保持する。動作において、実行されるべきCommand（コマンド：命令）１０５２は、Queue（キュー）１０５３に積まれる。CommandInvokerは、連続的に実行するコマンドスレッドを生成する。CommandInvoker内で既に実行中のCommandがなければ、CommandInvoker１０５１により実行されるように意図されたCommand１０５２が実行される。CommandInvokerが既にコマンドを実行している場合、新しいCommandは、Queue１０５３の最後に積まれる。しかしながら、それぞれのCommandInvoker１０５１では、一度に１つのCommandのみが実行される。指定されたCommandの実行に失敗した場合、CommandInvoker１０５１は例外処理を実行する。

【００７０】
CommandInvoker１０５１により実行されるCommandの型は、UndoableCommand（取消可能コマンド）１０５４、AsynchronousCommand（非同期コマンド）１０５５、及びVCCommand（ＶＣコマンド）１０５６を含むが、これらに限定されない。UndoableCommand１０５４は、ユーザが望めば、そのCommandの結果を取り消すことが可能なCommandである。UndoableCommandの例として、切り取り、コピー、テキストの挿入、などがある。動作において、ユーザが文書の一部を選択し、その部分に切り取りコマンドを適用するとき、UndoableCommandを用いることにより、切り取られた部分は、必要であれば、「切り取られていない」ようにすることができる。

【００７１】
VCCommand１０５６は、ボキャブラリコネクション記述子（Vocabulary Connection Descriptor：ＶＣＤ）スクリプトファイルに格納される。これらは、プログラマにより定義

されうるユーザ指定のCommandである。Commandは、例えば、ＸＭＬフラグメントを追加したり、ＸＭＬフラグメントを削除したり、属性を設定したりするための、より抽象的なCommandの組合せであってもよい。これらのCommandは、特に、文書の編集に焦点を合わせている。

【００７２】

AsynchronousCommand１０５５は、文書のロードや保存など、システムよりのCommandであり、UndoableCommandやVCCommandとは別に、非同期的に実行される。AsynchronousCommandは、UndoableCommandではないので、取り消すことはできない。

【００７３】

ｃ）リソース

Resource１０９は、様々なクラスに、いくつかの機能を提供するオブジェクトである。例えば、ストリングリソース、アイコン、及びデフォルトキーバインドは、システムで使用されるResourceの例である。

【００７４】

２．アプリケーションコンポーネント

文書処理システムの第２の主要な特徴であるアプリケーションコンポーネント１０２は、実行環境１０１において実行される。アプリケーションコンポーネント１０２は、実際の文書と、システム内における文書の様々な論理的、物理的な表現を含む。さらに、アプリケーションコンポーネント１０２は、文書を管理するために使用されるシステムの構成を含む。アプリケーションコンポーネント１０２は、さらに、UserApplication（ユーザアプリケーション）１０６、アプリケーションコア１０８、ユーザインタフェイス１０７、及びCoreComponent（コアコンポーネント）１１０を含む。

【００７５】

ａ）ユーザアプリケーション

UserApplication１０６は、ProgramInvoker１０３と共にシステム上にロードされる。UserApplication１０６は、文書と、文書の様々な表現と、文書と対話するために必要なユーザインタフェイスとをつなぐ接着剤となる。例えば、ユーザが、プロジェクトの一部である文書のセットを生成したいとする。これらの文書がロードされると、文書の適切な表現が生成される。ユーザインタフェイス機能は、UserApplication１０６の一部として追加される。言いかえれば、UserApplication１０６は、ユーザがプロジェクトの一部を形成する文書と対話することを可能とする文書の表現と、文書の様々な態様とを、共に保持する。一旦UserApplication１０６が生成されると、ユーザがプロジェクトの一部を形成する文書との対話を望むたびに、ユーザは簡単に実行環境上にUserApplication１０６をロードすることができる。

【００７６】

ｂ）コアコンポーネント

CoreComponent１１０は、複数のPane（ペイン）の間で文書を共有する方法を提供する。後で詳述するように、Paneは、ＤＯＭツリーを表示し、画面の物理的なレイアウトを扱う。例えば、物理的な画面は、個々の情報の断片を描写する画面内の複数のPaneからなる。ユーザから画面上に見える文書は、１又はそれ以上のPaneに出現しうる。また、２つの異なる文書が画面上で２つの異なるPaneに現れてもよい。

【００７７】

図１１（ｃ）に示されるように、画面の物理的なレイアウトもツリーの形式になっている。Paneは、RootPane（ルートペイン）１０８４にもなり得るし、SubPane（サブペイン）１０８５にもなり得る。RootPane１０８４は、Paneのツリーの根に当たるPaneであり、SubPane１０８５は、RootPane１０８４以外の任意のPaneである。

【００７８】

CoreComponent１１０は、さらに、フォントを提供し、ツールキットなど、文書のための複数の機能的な操作のソースの役割を果たす。CoreComponent１１０により実行されるタスクの一例に、複数のPane間におけるマウスカーソルの移動がある。実行されるタスク

の他の例として、あるPane中の文書の一部をマークし、それを異なる文書を含む別のPane上にコピーする。

【００７９】

ｃ）アプリケーションコア

上述したように、アプリケーションコンポーネント１０２は、システムにより処理され管理される文書から構成される。これは、システム内における文書の様々な論理的及び物理的な表現を含む。アプリケーションコア１０８は、アプリケーションコンポーネント１０２の構成である。その機能は、実際の文書を、それに含まれる全てのデータとともに保持することである。アプリケーションコア１０８は、DocumentManager（ドキュメントマネージャ：文書管理部）１０８１及びDocument（ドキュメント：文書）１０８２自身を含む。

【００８０】

DocumentManager１０８１の様々な態様を以下に詳述する。DocumentManager１０８１は、Document１０８２を管理する。DocumentManager１０８１は、RootPane１０８４、SubPane１０８５、ClipBoard（クリップボード）ユーティリティ１０８７、及びSnapShot（スナップショット）ユーティリティ１０８８にも接続される。ClipBoardユーティリティ１０８７は、ユーザがクリップボードに加えることを決定した文書の部分を保持する方法を提供する。例えば、ユーザが、文書の一部を切り取り、後で再考するために新規文書にそれを保存することを望んだとする。このような場合、切り取られた部分がClipBoardに追加される。

【００８１】

つづいて、SnapShotユーティリティ１０８８についても説明する。SnapShotユーティリティ１０８８は、アプリケーションがある状態から別の状態まで移行するときに、アプリケーションの現在の状態を記憶することを可能とする。

【００８２】

ｄ）ユーザインタフェイス

アプリケーションコンポーネント１０２の別の構成は、ユーザがシステムと物理的に対話する手段を提供するユーザインタフェイス１０７である。例えば、ユーザインタフェイスは、ユーザが文書をアップロードしたり、削除したり、編集したり、管理したりするために使用される。ユーザインタフェイスは、Frame（フレーム）１０７１、MenuBar（メニューバー）１０７２、StatusBar（ステータスバー）１０７３、及びURLBar（ＵＲＬバー）１０７４を含む。

【００８３】

Frame１０７１は、一般に知られているように、物理的な画面のアクティブな領域であるとみなされる。MenuBar１０７２は、ユーザに選択を提供するメニューを含む画面領域である。StatusBar１０７３は、アプリケーションの実行状態を表示する画面領域である。URLBar１０７４は、インターネットをナビゲートするためにＵＲＬアドレスを入力する領域を提供する。

【００８４】

Ｃ．文書管理及び関連するデータ構造

図１２は、DocumentManager１０８１の詳細を示す。これは、文書処理システム内で文書を表現するために用いられるデータ構造及び構成を含む。分かりやすくするために、このサブセクションで説明される構成は、ＭＶＣパラダイムを用いて説明される。

【００８５】

DocumentManager１０８１は、文書処理システム内にある全ての文書を保持しホストするDocumentContainer（ドキュメントコンテナ：文書コンテナ）２０３を含む。DocumentManager１０８１にアタッチされたツールキット２０１は、DocumentManager１０８１により使用される様々なツールを提供する。例えば、DomService（ＤＯＭサービス）は、文書に対応するＤＯＭを生成し、保持し、管理するために必要とされる全ての機能を提供するために、ツールキット２０１により提供されるツールである。ツールキット２０１により

提供される別のツールであるIOManager（入出力管理部）は、システムへの入力及びシステムからの出力を管理する。同様に、StreamHandler（ストリームハンドラ）は、ビットストリームによる文書のアップロードを扱うツールである。これらのツールは、図中に特に示さず、参照番号を割り当てないが、ツールキット２０１のコンポーネントを形成する。

【００８６】

ＭＶＣパラダイムの表現によれば、モデル（Ｍ）は、文書のＤＯＭツリーモデル２０２を含む。前述したように、全ての文書は、文書処理システムにおいてＤＯＭツリーとして表現される。文書は、また、DocumentContainer２０３の一部を形成する。

【００８７】

１．ＤＯＭモデル及びゾーン

文書を表現するＤＯＭツリーは、Node（ノード）２０２１を有するツリーである。ＤＯＭツリーの部分集合であるZone（ゾーン）２０９は、ＤＯＭツリー内の１以上のNodeの関連領域を含む。例えば、画面上で文書の一部のみを表示し得るが、この可視化された文書の一部はZone２０９を用いて表示される。Zoneは、ZoneFactory（ゾーンファクトリ：ゾーン生成部）２０５と呼ばれるプラグインを用いて、生成され、取り扱われ、処理される。ZoneはＤＯＭの一部を表現するが、１以上の「名前空間」を使用してもよい。よく知られているように、名前空間は、名前空間内でユニークな名前の集合である。換言すれば、名前空間内に同じ名前は存在しない。

【００８８】

２．Facet及びFacetとZoneとの関係

Facet（ファセット）２０２２は、ＭＶＣパラダイムのモデル（Ｍ）部分内の別の構成である。Facetは、ZoneにおいてNodeを編集するために使用される。Facet２０２２は、Zone自身の内容に影響を与えずに実行することができる手続（プロシージャ）を使用して、ＤＯＭへのアクセスを編成する。次に説明するように、これらの手続は、Nodeに関連した重要で有用な操作を実行する。

【００８９】

各Nodeは、対応するFacetを有する。ＤＯＭの中のNodeを直接操作する代わりに、操作を実行するためにFacetを使用することによって、ＤＯＭの保全性は保護される。操作がNode上で直接実行される場合、いくつかのプラグインがＤＯＭを同時に変更することができ、その結果矛盾を引き起こす。

【００９０】

Ｗ３Ｃが策定したＤＯＭの標準規格は、Nodeを操作するための標準的なインタフェイスを定義するが、実際には、ボキャブラリごと又はNodeごとに特有の操作があるので、これらの操作をＡＰＩとして用意しておくのが好都合である。文書処理システムでは、このような各Nodeに特有のＡＰＩをFacetとして用意し、各Nodeにアタッチする。これにより、ＤＯＭの標準規格に準拠しつつ、有用なＡＰＩを付加することができる。また、ボキャブラリごとに特有のＤＯＭを実装するのではなく、標準的なＤＯＭの実装に、後から特有のＡＰＩを付加するようにすることで、多様なボキャブラリを統一的に処理することができるとともに、複数のボキャブラリが任意の組合せで混在した文書を適切に処理することができる。

【００９１】

ボキャブラリは、名前空間に属するタグ（例えばＸＭＬのタグ）のセットである。上述したように、名前空間は、ユニークな名前（ここではタグ）のセットを有する。ボキャブラリは、ＸＭＬ文書を表現するＤＯＭツリーのサブツリーとして現れる。このサブツリーはZoneを含む。特定の例においては、タグセットの境界はZoneによって定義される。Zone２０９は、ZoneFactory２０５と呼ばれるServiceを利用して生成される。上述したように、Zone２０９は、文書を表現するＤＯＭツリーの一部の内部表現である。このような文書の一部へのアクセスを提供するために、論理的な表現が要求される。この論理的表現は、文書が画面上で論理的にどのように表現されるかについてコンピュータに通知する。Canv

as（キャンバス）２１０は、Zoneに対応する論理的なレイアウトを提供するように作用するServiceである。

【００９２】

他方、Pane２１１は、Canvas２１０により提供される論理的なレイアウトに対応する物理的な画面レイアウトである。実際、ユーザは表示画面上で文字や画像によって文書のレンダリングのみを見る。したがって、文書は、画面上に文字や画像を描画するプロセスにより、画面上に描写されなければならない。文書は、Pane２１１により提供される物理的なレイアウトに基づいて、Canvas２１０により画面上に描写される。

【００９３】

Zone２０９に対応するCanvas２１０は、Editlet２０６を使用して生成される。文書のＤＯＭは、Editlet２０６及びCanvas２１０を使用して編集される。元の文書の完全性を維持するために、Editlet２０６及びCanvas２１０は、Zone２０９における１以上のNodeに対応するFacetを使用する。これらのServiceは、Zone及びＤＯＭ内のNodeを直接操作しない。Facetは、Command２０７を利用して操作される。

【００９４】

ユーザは、一般に、画面上のカーソルを移動させたり、コマンドをタイプしたりすることによって、画面と対話する。画面上の論理的なレイアウトを提供するCanvas２１０は、このカーソル操作を受け付ける。Canvas２１０は、対応するアクションをFacetに実行させることができる。この関係により、カーソルサブシステム２０４は、DocumentManager１０８１に対して、ＭＶＣパラダイムのコントローラ（Ｃ）として機能する。Canvas２１０は、イベントを扱うタスクも有する。例えば、Canvas２１０は、マウスクリック、フォーカス移動、及びユーザにより起こされた同様のアクションなどのイベントを扱う。

【００９５】

３．Zone、Facet、Canvas及びPaneの間の関係の概要

文書処理システム内の文書は、少なくとも４つの観点から見ることができる。すなわち、１）文書処理システムにおいて文書の内容及び構造を保持するために用いられるデータ構造、２）文書の保全性に影響を与えずに文書の内容を編集する手段、３）文書の画面上の論理的なレイアウト、４）文書の画面上の物理的なレイアウト、である。Zone、Facet、Canvas及びPaneは、前述の４つの観点に相当する、文書処理システムのコンポーネントをそれぞれ表す。

【００９６】

４．アンドゥサブシステム

上述したように、文書に対するいかなる変更（例えば編集）も取消可能であることが望ましい。例えば、ユーザが編集操作を実行し、次に、その変更の取消を決定したとする。図１２に関連して、アンドゥサブシステム２１２は、文書管理部の取消可能なコンポーネントを実現する。UndoManager（アンドゥマネージャ：アンドゥ管理部）２１２１は、ユーザによって取り消される可能性のある全ての文書に対する操作を保持する。

【００９７】

例えば、ユーザが、文書中の単語を別の単語に置換するコマンドを実行したとする。その後、ユーザは考え直し、元の単語に戻すことを決定したとする。アンドゥサブシステム２１２は、このような操作を支援する。UndoManager２１２１は、このようなUndoableEdit（アンドゥアブルエディット：取消可能な編集）２１２２の操作を保持する。

【００９８】

５．カーソルサブシステム

前述したように、ＭＶＣのコントローラ部分は、カーソルサブシステム２０４を備えてもよい。カーソルサブシステム２０４は、ユーザから入力を受け付ける。これらの入力は、一般にコマンド及び／又は編集操作の性格を有している。したがって、カーソルサブシステム２０４は、DocumentManager１０８１に関連したＭＶＣパラダイムのコントローラ（Ｃ）部分であると考えることができる。

【００９９】

６．ビュー

前述したように、Canvas２１０は、画面上に提示されるべき文書の論理的なレイアウトを表す。ＸＨＴＭＬ文書の例では、Canvas２１０は、文書が画面上でいかに見えるかを論理的に表現したボックスツリー２０８を含んでもよい。このボックスツリー２０８は、DocumentManager１０８１に関連したＭＶＣパラダイムのビュー（Ｖ）部分に含まれよう。

【０１００】

Ｄ．ボキャブラリコネクション

文書処理システムの重要な特徴は、ＸＭＬ文書を、他の表現にマップして取り扱うことが可能で、かつ、マップした先の表現を編集すると、その編集が元のＸＭＬ文書に整合性を保ちつつ反映される環境を提供することにある。

【０１０１】

マークアップ言語により記述された文書、例えばＸＭＬ文書は、文書型定義により定義されたボキャブラリに基づいて作成されている。ボキャブラリは、タグのセットである。ボキャブラリは、任意に定義されてもよいため、無限に多くのボキャブラリが存在しうる。しかしながら、多数の可能なボキャブラリのそれぞれに対して専用の処理／管理環境を提供するのは現実的ではない。ボキャブラリコネクションは、この問題を解決する方法を提供する。

【０１０２】

例えば、文書は２以上のマークアップ言語により記述されてもよい。文書は、例えば、ＸＨＴＭＬ（eXtensible HyperText Markup Language）、ＳＶＧ（Scalable Vector Graphics）、ＭａｔｈＭＬ（Mathematical Markup Language）、その他のマークアップ言語により記述されてもよい。換言すれば、マークアップ言語は、ＸＭＬにおけるボキャブラリやタグセットと同様に見なされてもよい。

【０１０３】

ボキャブラリは、ボキャブラリプラグインを用いて処理される。文書処理システムにおいてプラグインが利用不可能であるボキャブラリにより記述された文書は、プラグインが利用可能である別のボキャブラリの文書にマッピングすることにより表示される。この特徴により、プラグインが用意されていないボキャブラリの文書も適切に表示することができる。

【０１０４】

ボキャブラリコネクションは、定義ファイルを取得し、取得した定義ファイルに基づいて２つの異なるボキャブラリの間でマッピングする能力を含む。あるボキャブラリで記述された文書は、別のボキャブラリにマッピングすることができる。このように、ボキャブラリコネクションは、文書がマッピングされるボキャブラリに対応した表示／編集プラグインにより文書を表示し編集することを可能にする。

【０１０５】

上述したように、各文書は、一般に複数のノードを有するＤＯＭツリーとして文書処理システムにおいて記述される。「定義ファイル」は、それぞれのノードについて、そのノードと他のノードとの対応を記述する。各ノードの要素値及び属性値が編集可能か否かが指定される。ノードの要素値又は属性値を用いた演算式が記述されてもよい。

【０１０６】

マッピングという特徴を利用して、定義ファイルを適用したデスティネーションＤＯＭツリーが生成される。このように、ソースＤＯＭツリーとデスティネーションＤＯＭツリーの関係が構築され保持される。ボキャブラリコネクションは、ソースＤＯＭツリーとデスティネーションＤＯＭツリーの対応を監視する。ユーザから編集指示を受けると、ボキャブラリコネクションは、ソースＤＯＭツリーの関連したノードを変更する。ソースＤＯＭツリーが変更されたことを示す「ミューテーションイベント」が発行され、デスティネーションＤＯＭツリーがそれに応じて変更される。

【０１０７】

ボキャブラリコネクションの使用により、少数のユーザのみに知られていた比較的マイ

ナーなボキャブラリを、別のメジャーなボキャブラリに変換することができる。したがって、少数のユーザによって利用されるマイナーなボキャブラリであっても、文書を適切に表示し、望ましい編集環境を提供することができる。

【０１０８】

このように、文書処理システムの一部であるボキャブラリコネクションサブシステムは、文書の複数の表現を可能にする機能を提供する。

【０１０９】

図１３は、ボキャブラリコネクション（ＶＣ：Vocabulary Connection）サブシステム３００を示す。ＶＣサブシステム３００は、同一の文書の２つの代替表現の整合性を維持する方法を提供する。例えば、２つの表現は、同一文書の、２つの異なるボキャブラリによる表現であってもよい。前述したように、一方はソースＤＯＭツリーであってもよく、他方はデスティネーションＤＯＭツリーであってもよい。

【０１１０】

１．ボキャブラリコネクションサブシステム

ボキャブラリコネクションサブシステム３００の機能は、VocabularyConnection３０１と呼ばれるプラグインを使用して、文書処理システムにおいて実現される。文書が表現されるVocabulary３０５ごとに、対応するプラグインが要求される。例えば、文書の一部がＨＴＭＬで記述され、残りがＳＶＧで記述されている場合、ＨＴＭＬとＳＶＧに対応するボキャブラリプラグインが要求される。

【０１１１】

VocabularyConnectionプラグイン３０１は、適切なVocabulary３０５の文書に対応した、Zone２０９又はPane２１１のための適切なVCCanvas（ボキャブラリコネクションキャンバス）３１０を生成する。VocabularyConnection３０１を用いて、ソースＤＯＭツリー内のZone２０９に対する変更は、変換ルールにより、別のＤＯＭツリー３０６の対応するZoneに伝達される。変換ルールは、ボキャブラリコネクション記述子（Vocabulary Connection Descriptor：ＶＣＤ）の形式で記述される。このようなソースＤＯＭとデスティネーションＤＯＭの間の変換に対応するそれぞれのＶＣＤファイルについて、対応するVCManager（ボキャブラリコネクションマネージャ）３０２が生成される。

【０１１２】

２．Connector

Connector３０４は、ソースＤＯＭツリーのソースノードと、デスティネーションＤＯＭツリーのデスティネーションノードとを接続する。Connector３０４は、ソースＤＯＭツリー中のソースノード、及びソースノードに対応するソース文書に対する修正（変更）を見るために作用する。そして、対応するデスティネーションＤＯＭツリーのノードを修正する。Connector３０４は、デスティネーションＤＯＭツリーを修正することができる唯一のオブジェクトである。例えば、ユーザは、ソース文書、及び対応するソースＤＯＭツリーに対してのみ修正を行うことができる。その後、Connector３０４がデスティネーションＤＯＭツリーに、対応する修正を行う。

【０１１３】

Connector３０４は、ツリー構造を形成するために、論理的にリンクされる。Connector３０４により形成されたツリーは、ConnectorTree（コネクタツリー）と呼ばれる。Connector３０４は、ConnectorFactory（コネクタファクトリ：コネクタ生成部）３０３と呼ばれるServiceを用いて生成される。ConnectorFactory３０３は、ソース文書からConnector３０４を生成し、それらをリンクしてConnectorTreeを形成する。VocabularyConnectionManager３０２は、ConnectorFactory３０３を保持する。

【０１１４】

前述したように、ボキャブラリは名前空間におけるタグのセットである。図示されるように、Vocabulary３０５は、VocabularyConnection３０１によって文書に対して生成される。これは、文書ファイルを解析し、ソースＤＯＭとデスティネーションＤＯＭの間の写像のための適切なVocabularyConnectionManager３０２を生成することにより行われる。

さらに、Connectorを生成するConnectorFactory３０３と、Zone２０９を生成するZoneFactory２０５と、Zone内のノードに対応するCanvasを生成するEditlet２０６との間の適切な関係が作られる。ユーザがシステムから文書を処分又は削除するとき、対応するVocabularyConnectionManager３０２が削除される。

【０１１５】

Vocabulary３０５は、VCCanvas３１０を生成する。さらに、Connector３０４及びデスティネーションＤＯＭツリー３０６が対応して生成される。

【０１１６】

ソースＤＯＭ及びCanvasは、それぞれ、モデル（Ｍ）及びビュー（Ｖ）に対応する。しかしながら、このような表現は、ターゲットのボキャブラリが画面上に描写可能である場合に限って意味がある。描写は、ボキャブラリプラグインにより行われる。ボキャブラリプラグインは、主要なボキャブラリ、例えば、ＸＨＴＭＬ、ＳＶＧ、ＭａｔｈＭＬについて提供される。ボキャブラリプラグインは、ターゲットのボキャブラリに関連して使用される。これらは、ボキャブラリコネクション記述子を用いてボキャブラリ間でマッピングする方法を提供する。

【０１１７】

このようなマッピングは、ターゲットのボキャブラリが、マッピング可能で、画面上に描写される方法が予め定義されたものである場合にのみ意味がある。このようなレンダリング方法は、例えばＸＨＴＭＬなどのように、Ｗ３Ｃなどの組織により定義された標準規格となっている。

【０１１８】

ボキャブラリコネクションが必要であるとき、VCCanvasが使用される。この場合、ソースのビューを直接生成することができないので、ソースのCanvasは生成されない。この場合、VCCanvasが、ConnectorTreeを使用して生成される。このVCCanvasは、イベントの変換のみを扱い、画面上の文書の描写を援助しない。

【０１１９】

３．DestinationZone、Pane、及びCanvas

上述したように、ボキャブラリコネクションサブシステムの目的は、同一の文書の２つの表現を同時に生成し保持することである。第２の表現も、ＤＯＭツリーの形式であり、これはデスティネーションＤＯＭツリーとして既に説明した。第２の表現における文書を見るために、DestinationZone、Canvas及びPaneが必要である。

【０１２０】

VCCanvasが作成されると、対応するDestinationPane３０７が生成される。さらに、関連するDestinationCanvas３０８と、対応するBoxTree３０９が生成される。同様に、VCCanvas３１０も、ソース文書に対するPane２１１及びZone２０９に関連づけられる。

【０１２１】

DestinationCanvas３０８は、第２の表現における文書の論理的なレイアウトを提供する。特に、DestinationCanvas３０８は、デスティネーション表現における文書を描写するために、カーソルや選択のようなユーザインタフェイス機能を提供する。DestinationCanvas３０８に生じたイベントは、Connectorに供給される。DestinationCanvas３０８は、マウスイベント、キーボードイベント、ドラッグアンドドロップイベント、及び文書のデスティネーション（第２）表現のボキャブラリに特有なイベントを、Connector３０４に通知する。

【０１２２】

４．ボキャブラリコネクションコマンドサブシステム

ボキャブラリコネクション（ＶＣ）サブシステム３００の要素として、ボキャブラリコネクション（ＶＣ）コマンドサブシステム３１３がある。ボキャブラリコネクションコマンドサブシステム３１３は、ボキャブラリコネクションサブシステム３００に関連した命令の実行のために使用されるVCCommand（ボキャブラリコネクションコマンド）３１５を生成する。VCCommandは、内蔵のCommandTemplate（コマンドテンプレート）３１８を使用

して、及び／又は、スクリプトサブシステム３１４においてスクリプト言語を使用してスクラッチからコマンドを生成することにより、生成することができる。

【０１２３】

コマンドテンプレートには、例えば、「If」コマンドテンプレート、「When」コマンドテンプレート、「挿入（Insert）」コマンドテンプレートなどがある。これらのテンプレートは、VCCommandを作成するために使用される。

【０１２４】

５．ＸＰａｔｈサブシステム

ＸＰａｔｈサブシステム３１６は、文書処理システムの重要な構成であり、ボキャブラリコネクションの実現を支援する。Connector３０４は、一般にxpath情報を含む。上述したように、ボキャブラリコネクションのタスクの１つは、ソースＤＯＭツリーの変化をデスティネーションＤＯＭツリーに反映させることである。xpath情報は、変更／修正を監視されるべきソースＤＯＭツリーのサブセットを決定するために用いられる１以上のxpath表現を含む。

【０１２５】

６．ソースＤＯＭツリー、デスティネーションＤＯＭツリー、及びConnectorTreeの概要

ソースＤＯＭツリーは、別のボキャブラリに変換される前のボキャブラリで文書を表現したＤＯＭツリー又はZoneである。ソースＤＯＭツリーのノードは、ソースノードと呼ばれる。

【０１２６】

それに対して、デスティネーションＤＯＭツリーは、ボキャブラリコネクションに関連して前述したように、同一の文書を、マッピングにより変換された後の異なるボキャブラリで表現したＤＯＭツリー又はZoneである。デスティネーションＤＯＭツリーのノードは、デスティネーションノードと呼ばれる。

【０１２７】

ConnectorTreeは、ソースノードとデスティネーションノードの対応を表すConnectorに基づく階層的表現である。Connectorは、ソースノードと、ソース文書になされた修正を監視し、デスティネーションＤＯＭツリーを修正する。Connectorは、デスティネーションＤＯＭツリーを修正することを許された唯一のオブジェクトである。

【０１２８】

Ｅ．文書処理システムにおけるイベントフロー

実用のためには、プログラムはユーザからのコマンドに応答しなければならない。イベントは、プログラム上で実行されたユーザアクションを記述し実行する方法である。多くの高級言語、例えばＪａｖａ（登録商標）は、ユーザアクションを記述するイベントに頼っている。従来、プログラムは、ユーザアクションを理解し、それを自身で実行するために、積極的に情報を集める必要があった。これは、例えば、プログラムが自身を初期化した後、ユーザが画面、キーボード、マウスなどでアクションを起こしたときに適切な処理を講じるために、ユーザのアクションを繰り返し確認するループに入ることを意味する。しかしながら、このプロセスは扱いにくい。さらに、それは、ユーザが何かをするのを待つ間、ＣＰＵサイクルを消費してループするプログラムを必要とする。

【０１２９】

多くの言語が、異なるパラダイムを採用することにより、これらの問題を解決している。そのうちの一つは、現代の全てのウィンドウシステムの基礎となっている、イベントドリブンプログラミングである。このパラダイムでは、全てのユーザアクションは、「イベント」と呼ばれる抽象的な事象の集合に属する。イベントは、十分詳細に、特定のユーザアクションを記述する。プログラムがユーザにより生成されたイベントを積極的に収集するのではなく、監視すべきイベントが生じたときに、システムがプログラムに通知する。この方法によりユーザとの対話を扱うプログラムは「イベントドリブン」であると言われる。

【０１３０】
これは、多くの場合、全てのユーザにより生成されたイベントの基本特性を獲得する「Event（イベント）」クラスを使用して扱われる。
【０１３１】
文書処理システムは、自身のイベント、及びこれらのイベントを扱う方法を定義して使用する。いくつかの型のイベントが使用される。例えば、マウスイベントは、ユーザのマウスアクションから起こるイベントである。マウスを含むユーザアクションは、Canvas２１０によって、マウスイベントに渡される。このように、Canvasは、システムのユーザによる相互作用の最前部にあると言える。必要であれば、最前部にあるCanvasは、そのイベントに関連した内容を子へ渡す。
【０１３２】
それに対して、キーストロークイベントは、Canvas２１０から流れる。キーストロークイベントは、即時的なフォーカスを有する。すなわち、それは、いかなる瞬間でも作業に関連する。Canvas２１０上に入力されたキーストロークイベントは、その親に渡される。キー入力は、文字列挿入を扱うことが可能な、異なるイベントによって処理される。文字列の挿入を扱うイベントは、キーボードを使用して文字が挿入されたときに発生する。他の「イベント」は、例えば、ドラッグイベント、ドロップイベント、マウスイベントと同様に扱われる他のイベントを含む。
【０１３３】
１．ボキャブラリコネクション外のイベントの取り扱い
イベントは、イベントスレッドを用いて渡される。Canvas２１０は、イベントを受け取ると、その状態を変更する。必要であれば、Command１０５２がCanvas２１０によりCommandQueue１０５３にポストされる。
【０１３４】
２．ボキャブラリコネクション内のイベントの取り扱い
VocabularyConnectionプラグイン３０１を用いて、DestinationCanvasの一例であるXHTMLCanvas１１０６は、発生したイベント、例えば、マウスイベント、キーボードイベント、ドラッグアンドドロップイベント、及びボキャブラリに特有のイベントなどを受け取る。これらのイベントは、コネクタ３０４に通知される。より詳細には、図２１（ｂ）に図示されるように、VocabularyConnectionプラグイン３０１内のイベントフローは、SourcePane１１０３、VCCanvas１１０４、DestinationPane１１０５、DestinationCanvasの一例であるDestinationCanvas１１０６、デスティネーションＤＯＭツリー及びConnectorTreeを通過する。
【０１３５】
Ｆ．ProgramInvoker及びProgramInvokerと他の構成との関係
ProgramInvoker１０３及びそれと他の構成との関係は、図１４（ａ）に更に詳細に示される。ProgramInvoker１０３は、文書処理システムを開始するために実行される実行環境中の基本的なプログラムである。図１１（ｂ）に図示されるように、UserApplication１０６、ServiceBroker１０４１、CommandInvoker１０５１、及びResource１０９は、全てProgramInvoker１０３に接続される。前述したように、アプリケーション１０２は、実行環境中で実行されるコンポーネントである。同様に、ServiceBroker１０４１は、システムに様々な機能を加えるプラグインを管理する。他方、CommandInvoker１０５１は、ユーザにより提供される命令を実行して、コマンドを実行するために使用されるクラス及びファンクションを保持する。
【０１３６】
１．プラグイン及びサービス
ServiceBroker１０４１について、図１４（ｂ）を参照して更に詳細に説明する。前述したように、ServiceBroker１０４１は、システムに様々な機能を追加するプラグイン（及び関連するサービス）を管理する。Service１０４２は、文書処理システムに特徴を追加又は変更可能な最も下の層である。「Service」は、ServiceCategory４０１とServiceP

rovider４０２の２つの部分からなる。図１４（ｃ）に図示されるように、１つのServiceCategory４０１は、複数の関連するServiceProvider４０２を持ちうる。それぞれのServiceProviderは、特定のServiceCategoryの一部または全部を実行するように作用する。ServiceCategory４０１は、他方では、Serviceの型を定義する。

【０１３７】

Serviceは、１）文書処理システムに特定の特色を提供する「特色サービス」、２）文書処理システムにより実行されるアプリケーションである「アプリケーションサービス」、３）文書処理システムの全体にわたって必要な特色を提供する「環境サービス」、の３つの型に分類することができる。

【０１３８】

Serviceの例は、図１４（ｄ）に示される。アプリケーションServiceのCategoryにおいては、システムユーティリティが対応するServiceProviderの例である。同様に、Editlet２０６はCategoryであり、HTMLEditlet及びSVGEditletは対応するServiceProviderである。ZoneFactory２０５は、Serviceの別のCategoryであり、対応するServiceProvider（図示せず）を有する。

【０１３９】

プラグインは、文書処理システムに機能性を加えると既に説明したが、いくつかのServiceProvider４０２及びそれらに関連するクラスからなるユニットと見なされてもよい。各プラグインは、宣言ファイルに記述された依存性及びServiceCategory４０１を有する。

【０１４０】

２．ProgramInvokerとアプリケーションとの関係

図１４（ｅ）は、ProgramInvoker１０３とUserApplication１０６との関係についての更なる詳細を示す。必要な文書やデータなどは、ストレージからロードされる。必要なプラグインは、全てServiceBroker１０４１上にロードされる。ServiceBroker１０４１は、全てのプラグインを保持し管理する。プラグインは、システムに物理的に追加することができ、又、その機能はストレージからロードすることができる。プラグインの内容がロードされると、ServiceBroker１０４１は、対応するプラグインを定義する。つづいて、対応するUserApplication１０６が生成され、実行環境１０１にロードされ、ProgramInvoker１０３にアタッチされる。

【０１４１】

Ｇ．アプリケーションサービスと環境との関係

図１５（ａ）は、ProgramInvoker１０３上にロードしたアプリケーションサービスの構成についての更なる詳細を示す。コマンドサブシステム１０５のコンポーネントであるCommandInvoker１０５１は、ProgramInvoker１０３内のCommand１０５２を起動又は実行する。Command１０５２は、文書処理システムにおいて、ＸＭＬなどの文書を処理し、対応するＸＭＬＤＯＭツリーを編集するために用いられる命令である。CommandInvoker１０５１は、Command１０５２を実行するために必要なクラス及びファンクションを保持する。

【０１４２】

ServiceBroker１０４１も、ProgramInvoker１０３内で実行される。UserApplication１０６は、ユーザインタフェイス１０７及びCoreComponent１１０に接続される。CoreComponent１１０は、全てのPaneの間で文書を共有する方法を提供する。CoreComponent１１０は、さらにフォントを提供し、Paneのためのツールキットの役割を果たす。

【０１４３】

図１５（ｂ）は、Frame１０７１、MenuBar１０７２、及びStatusBar１０７３の関係を示す。

【０１４４】

Ｈ．アプリケーションコア

図１６（ａ）は、全ての文書、及び文書の一部及び文書に属するデータを保持するアプリケーションコア１０８についての更なる説明を提供する。CoreComponent１１０は、文

書１０８２を管理するDocumentManager１０８１にアタッチされる。DocumentManager１０８１は、文書処理システムに関連づけられたメモリに格納される全ての文書１０８２の所有者である。

【０１４５】

画面上の文書の表示を容易にするために、DocumentManager１０８１はRootPane１０８４にも接続される。ClipBoard１０８７、SnapShot１０８８、Drag&Drop６０１、及びOverlay６０２の機能も、CoreComponent１１０にアタッチされる。

【０１４６】

SnapShot１０８８は、アプリケーションの状態を元に戻すために使用される。ユーザがSnapShot１０８８を起動したとき、アプリケーションの現状が検知され、格納される。その後、アプリケーションの状態が別の状態に変わるとき、格納された状態の内容は保存される。SnapShot１０８８は、図１６（ｂ）に図示される。動作において、アプリケーションがあるＵＲＬから他へ移動するときに、前に戻る動作及び先に進む動作をシームレスに実行可能とするために、SnapShot１０８８は以前の状態を記憶する。

【０１４７】

Ｉ．DocumentManager内における文書の構成

図１７（ａ）は、DocumentManager１０８１の更なる説明と、DocumentManagerにおいて文書が構成され保持される様子を示す。図１１（ｂ）に示したように、DocumentManager１０８１は、文書１０８２を管理する。図１７（ａ）に示される例において、複数の文書のうちの１つはRootDocument（ルート文書）７０１であり、残りの文書はSubDocument（サブ文書）７０２である。DocumentManager１０８１は、RootDocument７０１に接続され、RootDocument７０１は、全てのSubDocument７０２に接続される。

【０１４８】

図１２及び図１７（ａ）に示すように、DocumentManager１０８１は、全ての文書１０８２を管理するオブジェクトであるDocumentContainer２０３に結合される。DOMService７０３及びIOManager７０４を含むツールキット２０１（例えばＸＭＬツールキット）の一部を形成するツールも、DocumentManager１０８１に供給される。再び図１７（ａ）を参照して、DOMService７０３は、DocumentManager１０８１により管理される文書に基づいたＤＯＭツリーを生成する。各Document７０５は、それがRootDocument７０１であってもSubDocument７０２であっても、対応するDocumentContainer２０３によって管理される。

【０１４９】

図１７（ｂ）は、文書Ａ－Ｅが階層的に配置される様子を示す。文書ＡはRootDocumentである。文書Ｂ－Ｄは、文書ＡのSubDocumentである。文書Ｅは、文書ＤのSubDocumentである。図１７（ｂ）の左側は、これと同じ文書の階層が画面上に表示された例を示す。RootDocumentである文書Ａは、基本フレームとして表示される。文書ＡのSubDocumentである文書Ｂ－Ｄは、基本フレームＡの中のサブフレームとして表示される。文書ＤのSubDocumentである文書Ｅは、サブフレームＤのサブフレームとして画面に表示される。

【０１５０】

再び図１７（ａ）を参照して、UndoManager（アンドゥマネージャ：アンドゥ管理部）７０６及びUndoWrapper（アンドゥラッパー）７０７は、それぞれのDocumentContainer２０３に対して生成される。UndoManager７０６及びUndoWrapper７０７は、取消可能なコマンドを実行するために使用される。この特徴を使用することにより、編集操作を使用して文書に対して実行された変更を取り消すことができる。SubDocumentの変更は、RootDocumentとも密接な関係を有する。アンドゥ操作は、階層内の他の文書に影響する変更を考慮に入れて、例えば、図１７（ｂ）に示されるような連鎖状の階層における全ての文書の間で整合性が維持されることを保証する。

【０１５１】

UndoWrapper７０７は、DocumentContainer２０３内のSubDocumentに関連するアンドゥオブジェクトをラップし、それらをRootDocumentに関連するアンドゥオブジェクトに結合

させる。UndoWrapper７０７は、UndoableEditAcceptor（アンドゥアブルエディットアクセプタ：アンドゥ可能編集受付部）７０９に利用可能なアンドゥオブジェクトの収集を実行する。

【０１５２】

UndoManager７０６及びUndoWrapper７０７は、UndoableEditAcceptor７０９及びUndoableEditSource（アンドゥアブルエディットソース）７０８に接続される。当業者には理解されるように、Document７０５がUndoableEditSource７０８であってもよく、取消可能な編集オブジェクトのソースであってもよい。

【０１５３】

Ｊ．アンドゥコマンド及びアンドゥフレームワーク

図１８（ａ）及び図１８（ｂ）は、アンドゥフレームワーク及びアンドゥコマンドについて更なる詳細を提供する。図１８（ａ）に示されるように、UndoCommand８０１、RedoCommand８０２、及びUndoableEditCommand８０３は、図１１（ｂ）に示したようにCommand Invoker１０５１に積むことができるコマンドであり、順に実行される。UndoableEditCommand８０３は、UndoableEditSource７０８及びUndoableEditAcceptor７０９に更にアタッチされる。「foo」EditCommand８０４及び「bar」EditCommand８０５は、UndoableEditCommandの例である。

【０１５４】

１．UndoableEditCommandの実行

図１８（ｂ）は、UndoableEditCommandの実行を示す。まず、ユーザが編集コマンドを使用してDocument７０５を編集すると仮定する。第１ステップＳ１では、UndoableEditAcceptor７０９が、Document７０５のＤＯＭツリーであるUndoableEditSource７０８にアタッチされる。第２ステップＳ２では、ユーザにより発行されたコマンドに基づいて、Document７０５がＤＯＭのＡＰＩを用いて編集される。第３ステップＳ３では、ミューテーションイベントのリスナーが、変更がなされたことを通知される。すなわち、このステップでは、ＤＯＭツリーの全ての変更を監視するリスナーが編集操作を検知する。第４ステップＳ４では、UndoableEditがUndoManager７０６のオブジェクトとして格納される。第５ステップＳ５では、UndoableEditAcceptor７０９がUndoableEditSource７０８からデタッチされる。UndoableEditSource７０８は、Document７０５自身であってもよい。

【０１５５】

Ｋ．システムへの文書のロードに関する手順

上記のサブセクションでは、システムの様々なコンポーネント及びサブコンポーネントについて説明した。以下、これらのコンポーネントの使用に関する方法論について説明する。図１９（ａ）は、文書処理システムに文書がロードされる様子の概要を示す。それぞれのステップは、図２４－２８において、特定の例に関連して詳述される。

【０１５６】

簡単には、文書処理システムは、文書に含まれるデータからなるバイナリデータストリームからＤＯＭを生成する。ApexNode（エイペックスノード：頂点ノード）が、注目対象でありZoneに属する文書の一部のために生成される。つづいて、対応するPaneが同定される。同定されたPaneは、ApexNode及び物理的な画面表面からZone及びCanvasを生成する。Zoneは、次に、それぞれのノードにFacetを生成し、それらに必要とされる情報を提供する。Canvasは、ＤＯＭツリーから、ノードをレンダリングするためのデータ構造を生成する。

【０１５７】

より詳細には、文書はストレージ９０１からロードされる。文書のＤＯＭツリー９０２が生成される。文書を保持するための、対応するDocumentContainer９０３が生成される。DocumentContainer９０３は、DocumentManager９０４にアタッチされる。ＤＯＭツリーは、ルートノードと、ときには複数のセカンダリノードを含む。

【０１５８】

一般に、このような文書は、テキスト及びグラフィクスの双方を含む。したがって、Ｄ

ＯＭツリーは、例えば、ＸＨＴＭＬサブツリーだけでなくＳＶＧサブツリーを有してもよい。ＸＨＴＭＬサブツリーは、ＸＨＴＭＬのApexNode９０５を有する。同様に、ＳＶＧサブツリーは、ＳＶＧのApexNode９０６を有する。

【０１５９】

ステップ１では、ApexNode９０６が、画面の論理的なレイアウトであるPane９０７にアタッチされる。ステップ２では、Pane９０７は、PaneOwner（ペインオーナー：ペインの所有者）９０８であるCoreComponentに、ApexNode９０６のためのZoneFactoryを要求する。ステップ３では、PaneOwner９０８は、ZoneFactoryと、ApexNode９０６のためのCanvas FactoryであるEditletとを返す。

【０１６０】

ステップ４では、Pane９０７がZone９０９を生成する。Zone９０９はPane９０７にアタッチされる。ステップ５では、Zone９０９がそれぞれのノードに対してFacetを生成し、対応するノードにアタッチする。ステップ６では、Pane９０７がCanvas９１０を生成する。Canvas９１０はPane９０７にアタッチされる。Canvas９１０には様々なCommandが含まれる。ステップ７では、Canvas９１０が文書を画面にレンダリングするためのデータ構造を構築する。ＸＨＴＭＬの場合、これはボックスツリー構造を含む。

【０１６１】

１．ZoneのＭＶＣ

図１９（ｂ）は、ＭＶＣパラダイムを用いてZoneの構成の概要を示す。この場合、Zone及びFacetは文書に関連した入力であるから、モデル（Ｍ）はZone及びFacetを含む。Canvasと、文書を画面にレンダリングするためのデータ構造体は、ユーザが画面上に見る出力であるから、ビュー（Ｖ）はCanvas及びデータ構造体に対応する。Commandは、文書とその様々な関係に対して制御操作を実行するので、コントロール（Ｃ）はCanvasに含まれるCommandを含む。

【０１６２】

Ｌ．文書の表現

図２０を用いて、文書及びその様々な表現の例について以下に説明する。この例で使用される文書は、テキストと画像の双方を含む。テキストは、ＸＨＴＭＬを用いて表され、画像は、ＳＶＧを用いて表される。図２０は、文書のコンポーネント及び対応するオブジェクトの関係のＭＶＣ表現を詳細に示す。この例において、Document１００１は、Document１００１を保持するDocumentContainer１００２にアタッチされる。文書はＤＯＭツリー１００３により表現される。ＤＯＭツリーは、ApexNode１００４を含む。

【０１６３】

ApexNodeは、黒丸で表される。頂点でないノードは、白丸で表される。ノードを編集するために用いられるFacetは、三角形で表され、対応するノードにアタッチされる。文書がテキストと画像を有するので、この文書のＤＯＭツリーは、ＸＨＴＭＬ部分とＳＶＧ部分を含む。ApexNode１００４は、ＸＨＴＭＬサブツリーの最上のノードである。これは、文書のＸＨＴＭＬ部分の物理的な表現のための最上PaneであるXHTMLPane１００５にアタッチされる。ApexNode１００４は、文書のＤＯＭツリーの一部であるXHTMLZone１００６にもアタッチされる。

【０１６４】

Node１００４に対応するFacetも、XHTMLZone１００６にアタッチされる。XHTMLZone１００６は、XHTMLPane１００５にアタッチされる。XHTMLEditletは、文書の論理的な表現であるXHTMLCanvas１００７を生成する。XHTMLCanvas１００７は、XHTMLPane１００５にアタッチされる。XHTMLCanvas１００７は、Document１００１のＸＨＴＭＬコンポーネントのためのBoxTree１００９を生成する。文書のＸＨＴＭＬ部分を保持し描画するために必要な様々なCommand１００８も、XHTMLCanvas１００７に追加される。

【０１６５】

同様に、文書のＳＶＧサブツリーのApexNode１０１０は、文書のＳＶＧコンポーネントを表現するDocument１００１のＤＯＭツリーの一部であるSVGZone１０１１にアタッチさ

れる。ApexNode１０１０は、文書のＳＶＧ部分の物理的な表現の最上のPaneであるSVGPane１０１３にアタッチされる。文書のＳＶＧ部分の論理的な表現を表すSVGCanvas１０１２は、SVGEditletにより生成され、SVGPane１０１３にアタッチされる。画面上に文書のＳＶＧ部分をレンダリングするためのデータ構造及びコマンドは、SVGCanvasにアタッチされる。例えば、このデータ構造は、図示されるように、円、線、長方形などを含んでもよい。

【０１６６】

図２０に関連して説明された文書例の表現の一部について、図２１（ａ）に関連して、前述したＭＶＣパラダイムを用いて更に説明する。図２１（ａ）は、文書１００１のＸＨＴＭＬコンポーネントにおけるＭＶの関係を簡略化して示す。モデルは、Document１００１のＸＨＴＭＬコンポーネントのためのXHTMLZone１１０１である。XHTMLZoneのツリーには、いくつかのNode及びそれらに対応するFacetが含まれる。対応するXHTMLZone及びPaneは、ＭＶＣパラダイムのモデル（Ｍ）部分の一部である。ＭＶＣパラダイムのビュー（Ｖ）部分は、Document１００１のＸＨＴＭＬコンポーネントの、対応するXHTMLCanvas１１０２及びBoxTreeである。文書のＸＨＴＭＬ部分は、Canvasと、それに含まれるCommandを使用して画面に描写される。キーボードやマウス入力などのイベントは、図示されるように、逆方向へ進む。

【０１６７】

SourcePaneは、更なる機能、すなわち、ＤＯＭの保有者としての役割を有する。図２１（ｂ）は、図２１（ａ）に示したDocument１００１のコンポーネントに対するボキャブラリコネクションを提供する。ＤＯＭホルダーとして機能するSourcePane１１０３は、文書のソースＤＯＭツリーを含む。ConnectorTreeは、ConnectorFactoryにより生成され、デスティネーションＤＯＭの保有者としても機能するDestinationPane１１０５を生成する。DestinationPane１１０５は、XHTMLDestinationCanvas１１０６としてボックスツリーの形式でレイアウトされる。

【０１６８】

Ｍ．プラグインサブシステム、ボキャブラリコネクション、及びコネクタの関係

図２２（ａ）－（ｃ）は、それぞれ、プラグインサブシステム、ボキャブラリコネクション、及びConnectorに関連する更なる詳細を示す。プラグインサブシステムは、文書処理システムに機能を追加又は交換するために用いられる。プラグインサブシステムは、ServiceBroker１０４１を含む。ServiceBroker１０４１にアタッチされるZoneFactoryService１２０１は、文書の一部に対するZoneを生成する。EditletService１２０２も、ServiceBroker１０４１にアタッチされる。EditletService１２０２は、Zone中のNodeに対応するCanvasを生成する。

【０１６９】

ZoneFactoryの例は、XHTMLZone及びSVGZoneをそれぞれ生成するXHTMLZoneFactory１２１１及びSVGZoneFactory１２１２である。文書例に関連して前述したように、文書のテキストコンポーネントは、XHTMLZoneを生成することにより表現されてもよいし、画像はSVGZoneを用いて表現されてもよい。EditletServiceの例は、XHTMLEditlet１２２１及びSVGEditlet１２２２を含む。

【０１７０】

図２２（ｂ）は、ボキャブラリコネクションに関連する更なる詳細を示す。ボキャブラリコネクションは、前述したように、文書処理システムの重要な特徴であり、２つの異なる方法で文書の整合のとれた表現及び表示を可能とする。ConnectorFactory３０３を保持するVCManager３０２は、ボキャブラリコネクションサブシステムの一部である。ConnectorFactory３０３は、文書のConnector３０４を生成する。前述したように、Connectorは、ソースＤＯＭ中のノードを監視し、２つの表現の間の整合性を維持するために、デスティネーションＤＯＭ中のノードを修正する。

【０１７１】

Template３１７は、いくつかのノードの変換ルールを表す。ボキャブラリコネクション

記述子（ＶＣＤ）ファイルは、特定のパス又はルールを満たす要素又は要素の集合を他の要素に変換するいくつかのルールを表すTemplateのリストである。Template３１７及びCommandTemplate３１８は、全てVCManager３０２にアタッチされる。VCManagerは、ＶＣＤファイル中の全てのセクションを管理するオブジェクトである。１つのＶＣＤファイルに対して、１つのVCManagerオブジェクトが生成される。

【０１７２】

図２２（ｃ）は、Connectorに関連する更なる詳細を提供する。ConnectorFactory３０３は、ソース文書からConnectorを生成する。ConnectorFactory３０３は、Vocabulary、Template、及びElementTemplateにアタッチされ、それぞれ、VocabularyConnector、TemplateConnector、ElementConnectorを生成ずる。

【０１７３】

VCManager３０２は、ConnectorFactory３０３を保持する。Vocabularyを生成するために、対応するＶＣＤファイルが読み込まれる。こうして、ConnectorFactory３０３が生成される。このConnectorFactory３０３は、Zoneを生成するZoneFactory及びCanvasを生成するEditletに関連する。

【０１７４】

つづいて、ターゲットボキャブラリのEditletServiceが、VCCanvasを生成する。VCCanvasも、ソースＤＯＭツリー又はZoneにおけるApexNodeのConnectorを生成する。必要に応じて、子のConnectorが再帰的に生成される。ConnectorTreeは、ＶＣＤファイル中のテンプレートの集合により生成される。

【０１７５】

テンプレートは、マークアップ言語の要素を他の要素に変換するためのルールの集合である。例えば、各テンプレートは、ソースＤＯＭツリー又はZoneにマッチされる。適切にマッチした場合には、頂点Connectorが生成される。例えば、テンプレート「A/*/D」は、間にどんなノードがあるかに関係なく、ノードＡで始まりノードＤで終わる全ての枝に合致する。同様に、「//B」は、ルートからの全ての「B」ノードに一致する。

【０１７６】

Ｎ．ConnectorTreeに関係するＶＣＤファイルの例

特定の文書と関係する処理を説明する例を続ける。ドキュメントタイトルのある「MySampleXML」というタイトルの文書が文書処理システムにロードされる。図２３は、「MySampleXML」ファイルのための、VCManager及びConnectorFactoryTreeを用いたＶＣＤスクリプトの例を示す。スクリプトファイル中のボキャブラリセクション、テンプレートセクションと、VCManagerにおける対応するコンポーネントが示される。タグ「vcd:vocabulary」において、属性「match」は「sample:root」、「label」は「MySampleXML」、「call-template」は「sample template」となっている。

【０１７７】

この例では、Vocabularyは、「MySampleXML」のVCManagerにおいて「sample:root」として頂点要素を含む。対応するＵＩラベルは、「MySampleXML」である。テンプレートセクションにおいて、タグは「vcd:template」であり、名前は「sample:template」である。

【０１７８】

Ｏ．ファイルがシステムにロードされる方法の詳細な例

図２４－２８は、文書「MySampleXML」のロードについての詳細な記述を示す。図２４（ａ）に示されるステップ１では、文書がストレージ１４０５からロードされる。DOMServiceは、ＤＯＭツリー及びDocumentManager１４０６と対応するDocumentContainer１４０１を生成する。DocumentContainer１４０１は、DocumentManager１４０６にアタッチされる。文書は、ＸＨＴＭＬ及びMySampleXMLのサブツリーを含む。ＸＨＴＭＬのApexNode１４０３は、タグ「xhtml:html」が付されたＸＨＴＭＬの最上のノードである。「MySampleXML」のApexNode１４０４は、タグ「sample:root」が付された「MySampleXML」の最上ノードである。

【０１７９】
図２４（ｂ）に示されるステップ２では、RootPaneが文書のXHTMLZone、Facet、及びCanvasを生成する。Pane１４０７、XHTMLZone１４０８、XHTMLCanvas１４０９、及びBoxTree１４１０が、ApexNode１４０３に対応して生成される。
【０１８０】
図２４（ｃ）に示されるステップ３では、XHTMLZoneが知らないタグ「sample:root」を発見し、XHTMLCanvasの領域からSubPaneを生成する。
【０１８１】
図２５に示されるステップ４では、SubPaneが「sample:root」を扱うことができ、適切なZoneを生成可能なZoneFactoryを得る。このZoneFactoryは、ZoneFactoryを実行可能なVocabulary内にある。それは、「MySampleXML」のVocabularySectionの内容を含む。
【０１８２】
図２６に示されるステップ５では、「MySampleXML」に対応するVocabularyがDefaultZone１６０１を生成する。対応するEditletが生成され、対応するCanvasを生成するためにSubPane１５０１が提供される。Editletは、VCCanvasを生成する。そして、それはTemplateSectionを呼ぶ。ConnectorFactoryTreeも含まれている。ConnectorFactoryTreeは、ConnectorTreeとなる全てのConnectorを生成する。
【０１８３】
図２７に示されるステップ６では、各ConnectorがデスティネーションＤＯＭオブジェクトを生成する。コネクタのうちのいくつかはxpath情報を含んでいる。xpath情報は、変更／修正を監視する必要のあるソースＤＯＭツリーの部分集合を決定するために使用される１以上のxpath表現を含む。
【０１８４】
図２８に示されるステップ７では、ボキャブラリは、ソースＤＯＭのペインからデスティネーションＤＯＭツリーのDestinationPaneを作成する。これは、SourcePaneに基づいてなされる。デスティネーションツリーのApexNodeは、DestinationPane及び対応するZoneにアタッチされる。DestinationPaneは、DestinationCanvasを生成し、文書をデスティネーションのフォーマットでレンダリングするためのデータ構造及びコマンドを構築する、自身のEditletを提供される。
【０１８５】
図２９（ａ）は、対応するソースノードを持たず、デスティネーションツリーにのみ存在するノード上でイベントが発生したときのフローを示す。マウスイベント、キーボードイベントなど、Canvasが取得したイベントは、デスティネーションツリーを通過して、ElementTemplateConnectorに伝達される。ElementTemplateConnectorは対応するソースノードを持たないので、伝達されたイベントはソースノードに対する編集操作ではない。ElementTemplateConnectorは、伝達されたイベントがCommandTemplateに記述されたコマンドに合致すれば、それに対応するActionを実行する。合致するコマンドがなければ、ElementTemplateConnectorは、伝達されたイベントを無視する。
【０１８６】
図２９（ｂ）は、TextOfConnectorによりソースノードに対応づけられているデスティネーションツリーのノード上でイベントが発生したときのフローを示す。TextOfConnectorは、ソースＤＯＭツリーのＸＰａｔｈで指定されたノードからテキストノードを取得して、デスティネーションＤＯＭツリーのノードにマッピングする。マウスイベント、キーボードイベントなど、Canvasが取得したイベントは、デスティネーションツリーを通過して、TextOfConnectorに伝達される。TextOfConnectorは、伝達されたイベントを、対応するソースノードの編集コマンドにマッピングし、Queue１０５３に積む。編集コマンドは、Facetを介して実行されるＤＯＭのＡＰＩコールの集合である。キューに積まれたコマンドが実行されると、ソースノードが編集される。ソースノードが編集されると、ミューテーションイベントが発行され、リスナーとして登録されたTextOfConnectorにソースノードの変更が通知される。TextOfConnectorは、ソースノードの変更を、対応するデステ

ィネーションノードに反映させるように、デスティネーションツリーを再構築する。このとき、TextOfConnectorを含むテンプレートに、「for each」や「for loop」などの制御文が含まれている場合、ConnectorFactoryがこの制御文を再評価し、TextOfConnectorを再構築した後、デスティネーションツリーが再構築される。

【０１８７】

（実施の形態）

実施の形態では、文書にアノーテーションをつける技術を提案する。すなわち、元データのある範囲又はある領域に対して、新たな情報を付加する、又は関連づける技術を提案する。

【０１８８】

（第１の実施の形態）

第１の実施の形態では、アノーテーションをつけるための特殊なタグを用意し、そのアノーテーションタグを用いて文書にアノーテーションをつける。

【０１８９】

図３０は、本実施の形態の文書処理装置の構成を示す。本実施の形態の文書処理装置１００は、図１に示した前提技術の文書処理装置２０の構成に加えて、アノーテーションユニット７０及び文書取得部７８を備える。アノーテーションユニット７０は、文書に含まれるアノテーションを検出するアノーテーション検出部７２、アノーテーションを表示するアノーテーション表示部７４、ユーザからアノーテーションの付加要求を受け付けて文書にアノーテーションを付加するアノーテーション付加部７６を含む。

【０１９０】

アノーテーションをつけるためのアノーテーションタグは、アノーテーションタグに対応していない他のアプリケーションから無視されるタグであってもよい。すなわち、文書処理装置１００など、アノーテーションタグに対応しているアプリケーションで文書を開くときには、アノーテーションタグであると解釈されつつ、対応していない他のアプリケーションで開くときにもエラーとならないように、非対応のアプリケーションから無視されるタグを定義する。例えば、コメントやＰＩなどの、ＸＭＬ文書に用いることができ、かつ、処理系から無視されるタグを利用してもよい。また、ボキャブラリにおいて意味を有しないことが定義された特別なタグを利用してもよい。例えば、ＸＨＴＭＬの＜span＞タグを用いてアノーテーションをつけてもよい。また、現状ではＸＭＬ文書に用いることはできないが、アノーテーション用のタグであると解釈する処理系では正しく扱えるようなタグを利用してもよい。

【０１９１】

例えば、以下のようなタグ名をアノーテーションタグとして採用してもよい。

・「＊」（アスタリスク） 例：<* attr="value">文字列</*>
・空（ヌル） 例：< attr="value">文字列</ >
・「：」（コロン） 例：<: attr="value">文字列</:>

【０１９２】

取得部７８は、文書取得部と定義ファイル取得部の機能を有しており、文書を取得して文書処理装置２０に送るとともに、文書に対応づけられた定義ファイルを取得して文書処理装置２０に送る。アノーテーション検出部７２は、文書に含まれる上述のアノーテーションタグを検出する。アノーテーション検出部７２が、ＸＭＬ文書中のアノーテーションタグを検出した場合、アノーテーション表示部７４は、そのタグで囲まれたテキストを、他のテキストと識別可能に表示する。例えば、下線を引く、太字にする、表示色を変える、矩形で囲むなどして強調表示してもよいし、タグ名を周囲に表示してもよい。文書処理装置２０に、アノーテーションタグを処理するための専用のプラグインなどの機能ブロック、すなわちアノーテーションユニット７０を設けてもよいし、後述するように、定義ファイルにアノーテーションタグの処理方法を規定するテンプレートを記述し、ＶＣユニット８０によりアノーテーションタグを処理してもよい。後者の場合、ＶＣユニット８０がアノーテーションユニット７０の機能を実現する。すなわち、ＶＣユニット８０は、アノ

ーテーションタグを検出すると、定義ファイルに記述されたテンプレートにしたがって、アノーテーション要素を表示する。

【０１９３】

アノーテーション付加部７６は、ＸＭＬ文書の編集中に、ユーザからアノーテーションの付加を指示されると、ソースツリーにアノーテーションタグのノードを追加する。文書が保存されるとき、アノーテーションを付加されたテキストの前後にアノーテーションタグが付加される。アノーテーション付加部７６は、ユーザからアノーテーションの表示形式の指示を受け付けてもよい。例えば、ユーザが選択した文字列を、太字で表示するのか、斜体で表示するのか、反転表示するのか、などの表示形式の指示を受け付け、受け付けた表示形式にしたがってアノーテーションを表示してもよい。

【０１９４】

定義ファイルに、アノーテーションを付加するためのコマンドを用意してもよい。アノーテーション付加コマンドは、ソースツリーにアノーテーションノードを追加する処理を実行してもよい。この場合、定義ファイルには、アノーテーションタグのテンプレートが用意されており、アノーテーションタグは、例えば、ＸＨＴＭＬの＜b＞タグなどにマッピングされて強調表示されてもよい。

【０１９５】

図３１は、文書処理装置１００により処理される文書の例を示す。文書７５ａには、アノーテーションタグ７７ａが含まれている。図３２は、図３１に示した文書７５ａを文書処理装置１００により表示した画面の例を示す。アノーテーション検出部７２が文書中のアノーテーションタグ７７ａを検出すると、アノーテーション表示部７４がアノーテーションタグ７７ａで囲まれたアノーテーション７８ａを強調表示する。画面７９ａにおいて、ユーザが、例えば、テキスト「付加」を選択して、アノーテーションの付加を要求するコマンドを発行すると、アノーテーション付加部７６は、ソースＤＯＭツリーに、テキスト「付加」を要素値とするアノーテーション要素を新たに追加する。アノーテーション表示部７４は、文書のＤＯＭが変更されたことを通知するミューテーションイベントを受けて、アノーテーション要素を強調表示すべく表示を更新する。

【０１９６】

以上のような構成及び動作により、文書にアノーテーションを付加することを可能とし、文書を処理する際の利便性を向上させることができる。

【０１９７】

（第２の実施の形態）

第２の実施の形態では、アノーテーションの開始と終了を示すアンカーを用意し、そのアノーテーションアンカーを用いて文書にアノーテーションをつける。本実施の形態の文書処理装置の構成は、図３０に示した第１の実施の形態の文書処理装置１００の構成と同様である。

【０１９８】

アノーテーションアンカーは、アノーテーションの開始を示す空タグと、終了を示す空タグであってもよい。例えば、<annotation:start xmlns:annotation="http://xmlns.xfytec.com/annotation" />というタグをアノーテーションの開始位置に付加し、<annotation:end xmlns:annotation="http://xmlns.xfytec.com/annotation" />というタグをアノーテーションの終了位置に付加してもよい。この場合、アノーテーションが入れ子になっていても、ＸＭＬ文書の妥当性は維持される。<annotation:start />タグと<annotation:end />タグの対応は、ｉｄなどを属性で指定することで指定してもよい。例えば、<annotation:start id="1" />から開始されるアノーテーションは、<annotation:end id="1" />で終了する。第１の実施の形態で説明したタグを、アンカーとして利用してもよい。

【０１９９】

アノーテーション検出部７２は、文書に含まれる上述のアノーテーションアンカーを検出する。アノーテーション検出部７２が、ＸＭＬ文書中のアノーテーションアンカーを検出した場合、アノーテーション表示部７４は、その開始アンカーと終了アンカーで囲まれ

たテキストを、他のテキストと識別可能に表示する。例えば、下線を引く、太字にする、表示色を変える、矩形で囲むなどして強調表示してもよいし、タグ名を周囲に表示してもよい。

【０２００】

アノーテーション付加部７６は、ＸＭＬ文書の編集中に、ユーザからアノーテーションの付加を指示されると、ソースツリーに対して、アノーテーションの開始位置に開始アンカーのノードを、終了位置に終了アンカーのノードを追加する。文書が保存されるとき、アンカーが挿入された位置にアノーテーションアンカーが付加される。

【０２０１】

定義ファイルに、アノーテーションを付加するためのコマンドを用意してもよい。アノーテーション付加コマンドは、ソースツリーにアノーテーションの開始アンカーのノードと終了アンカーのノードを追加する処理を実行してもよい。この場合、定義ファイルには、アノーテーションアンカーのテンプレートが用意されており、そのテンプレートにしたがって、開始アンカーと終了アンカーに囲まれた範囲を強調表示してもよい。すなわち、この場合、ＶＣユニット８０がアノーテーションユニット７０の機能を実現する。

【０２０２】

図３３は、文書処理装置１００により処理される文書の例を示す。文書７５ｂには、アノーテーションアンカー７７ｂが含まれている。アノーテーション検出部７２が文書中のアノーテーションアンカー７７ｂを検出すると、アノーテーション表示部７４がアノーテーションアンカー７７ｂで囲まれたアノーテーションを強調表示する。表示される画面は図３２に示した画面７９ａと同様である。画面７９ａにおいて、ユーザが、例えば、テキスト「付加」を選択して、アノーテーションの付加を要求するコマンドを発行すると、アノーテーション付加部７６は、ソースＤＯＭツリーのテキスト「付加」の前後にアノーテーションアンカーを新たに追加する。アノーテーション表示部７４は、文書のＤＯＭが変更されたことを通知するミューテーションイベントを受けて、アノーテーションアンカーに囲まれたアノーテーションを強調表示すべく表示を更新する。

【０２０３】

（第３の実施の形態）

第３の実施の形態では、アノーテーションをつけるためのタグセットを処理するための処理系を用意し、そのタグセットに定義されたタグを用いて文書にアノーテーションをつける。

【０２０４】

本実施の形態では、アノーテーションをつけるためのタグセットを予め用意しておく。文書処理装置１００は、それぞれ相異なるタグセットの要素を処理する複数の処理系、例えばＨＴＭＬユニット５０、ＳＶＧユニット６０を備えており、それらの他に、アノーテーションをつけるために定義されたタグセットの要素を処理する処理系を備える。アノーテーション用タグセットは、第１及び第２の実施の形態で説明したタグを含んでもよい。

【０２０５】

アノーテーション用タグセットの処理系は、ＨＴＭＬユニット５０、ＳＶＧユニット６０などと同様に、ハードコードプラグインとして設けられてもよい。この場合の文書処理装置１００の構成は、図３０に示した第１の実施の形態の文書処理装置１００の構成と同様である。アノーテーションユニット７０は、ＨＴＭＬユニット５０やＳＶＧユニット６０などの他のボキャブラリプラグインと同列のボキャブラリプラグインとして機能してもよいし、他のボキャブラリプラグインとは独立に動作する特殊なプラグインとして機能してもよい。前者の場合、前提技術の説明で詳述したように、文書を記述するタグセットを処理する処理系が、Zoneを生成し、子孫のノードに向かって順にFacetを付加していくときに、アノーテーション用タグセットのノードが検出された場合、アノーテーションユニット７０がディスパッチされ、処理が委譲される。この意味では、上位のZoneがアノーテーション検出部７２の機能を果たすと言える。アノーテーション表示部７４は、アノーテーション用タグセットの要素を他のテキストと識別可能に表示する。アノーテーション付

加部７６は、ＸＭＬ文書の編集中に、ユーザからアノーテーションの付加を指示されると、ソースツリーの該当部分にアノーテーション用タグセットのノードを追加する。後者の場合、他のボキャブラリプラグインが自身の担当する要素を処理した後に、アノーテーション検出部７２がアノーテーション用タグセットのノードを検出して、アノーテーション表示部７４に表示させてもよい。

【０２０６】

アノーテーション用タグセットの処理系は、ＶＣユニット８０であってもよい。この場合の文書処理装置１００は、図３４に示すように、ＶＣユニット８０が更に定義ファイル合成部８８を含む。その他の構成は、図１に示した前提技術の文書処理装置２０と同様である。この例では、文書を記述するタグセット用の定義ファイルの他に、アノーテーション用タグセット用の定義ファイルを用意する。アノーテーション用定義ファイルには、アノーテーションを付加又は削除するコマンドや、アノーテーションの表示方法を規定するテンプレートなどが記述される。

【０２０７】

ユーザが文書にアノーテーションをつける場合、定義ファイル合成部８８は、その文書に対応づけられた定義ファイルに、アノーテーション用定義ファイルを組み込んで、文書に含まれるタグセットと、アノーテーション用タグセットの双方を処理可能な定義ファイルを生成する。すなわち、文書に含まれるタグセットとアノーテーション用タグセットを含む複合ボキャブラリが仮想的に生成され、その複合ボキャブラリを処理可能な複合定義ファイルが生成される。ＶＣユニット８０は、複合定義ファイルを用いて、文書にアノーテーション用タグをつけることが可能となる。文書にアノーテーションをつけることを想定して、定義ファイルに、アノーテーション用定義ファイルをインクルードする命令を予め記述しておいてもよいし、ユーザがアノーテーション用定義ファイルを追加するよう指示したときに、定義ファイル合成部８８が、アノーテーション用定義ファイルを読み込んで定義ファイルに組み込んでもよい。

【０２０８】

図３５は、文書処理装置１００により処理される文書の例を示す。文書７５ｃには、アノーテーションタグセットの要素７７ｃが含まれている。アノーテーション検出部７２が文書中のアノーテーション要素７７ｃを検出すると、アノーテーション表示部７４がアノーテーションを強調表示する。表示される画面は図３２に示した画面７９ａと同様である。画面７９ａにおいて、ユーザが、例えば、テキスト「付加」を選択して、アノーテーションの付加を要求するコマンドを発行すると、アノーテーション付加部７６は、ソースＤＯＭツリーのテキスト「付加」を要素値とする新たなアノーテーションノードを追加する。アノーテーション表示部７４は、文書のＤＯＭが変更されたことを通知するミューテーションイベントを受けて、アノーテーション要素の要素値を強調表示すべく表示を更新する。

【０２０９】

（第４の実施の形態）

第４の実施の形態では、文書ファイルとは別にアノーテーション用のファイルを用意し、そのアノーテーションファイルにＸＰａｔｈ式などを用いて、アノーテーションの位置を記録する。本実施の形態の文書処理装置の構成は、図３０に示した第１の実施の形態の文書処理装置１００の構成と同様である。

【０２１０】

アノーテーション用ファイルには、アノーテーションの位置が、要素値全体を指定して記録されてもよいし、要素値の一部を指定して記録されてもよい。後者の場合、現状のＸＰａｔｈの規格に則った式では要素値の内部の位置を表現できないので、例えば、ＸＰａｔｈ式の後に、要素値中の位置を示す情報を付加してもよい。要素中の位置は、例えば、要素値の先頭からの文字数、バイト数により表現されてもよいし、表示したときの座標などで表現されてもよい。また、ＸＰｏｉｎｔｅｒにより表現されてもよい。これらの情報は、文書に対応づけられた定義ファイルに記録されてもよい。

【０２１１】
アノーテーション検出部７２は、アノーテーションファイル取得部として機能し、文書に対応づけられたアノーテーション用ファイルを読み込んで、アノーテーションの位置を検出する。アノーテーション検出部７２が、アノーテーション用ファイルに記述されたアノーテーション位置を検出した場合、アノーテーション表示部７４は、記録された範囲を、他のテキストと識別可能に表示する。例えば、下線を引く、太字にする、表示色を変える、矩形で囲むなどして強調表示してもよいし、タグ名を周囲に表示してもよい。アノーテーション用ファイルに、アノーテーションの表示形式を記録しておいてもよい。この場合、アノーテーション表示部７４は、指定された表示形式にしたがってアノーテーションを表示する。

【０２１２】
アノーテーション付加部７６は、受付部及び記録部として機能し、ＸＭＬ文書の編集中に、ユーザからアノーテーションの付加を指示されると、アノーテーションを付加すべき範囲を取得し、その開始位置と終了位置を示す情報をアノーテーション用ファイルに記録する。文書が編集されて、アノーテーションの位置が変わった場合は、アノーテーション付加部７６は、それに追従してアノーテーションの開始位置と終了位置をずらしてアノーテーション用ファイルに記録する。

【０２１３】
本実施の形態では、文書にアノーテーションを記録するのではなく、文書とは異なるファイルに外部からアノーテーションを付加するので、文書自身を編集できないような場合にもアノーテーションをつけることができる。例えば、ウェブページにユーザがアノーテーションをつけたい場合にも適用できる。例えば、ウェブページの一部に下線を引くなどして強調表示したい場合に、その強調表示する部分をアノーテーション用ファイルに記録しておいてもよい。また、ウェブページの余白にメモをとり、そのメモをアノーテーション用ファイルに記録しておいてもよい。これにより、次に同じウェブページを見る際には、注釈やメモなどを記録したアノーテーション用ファイルが適用され、強調表示やメモを再現することができる。

【０２１４】
この例の場合、ウェブページが改変されると、ユーザのアノーテーション用ファイルに記録された位置がずれてしまう可能性がある。このような文書の改変に追従できるようにするために、文書が改変されたときに、その改変内容を文書に記録しておいてもよい。また、文書のバージョン番号を文書に記録し、アノーテーション用ファイルにも、アノーテーションをつけたときの文書のバージョン番号を記録しておく。これにより、文書が改変されていたとしても、文書とアノーテーション用ファイルを開いたときに、文書の改編履歴に追従してアノーテーションの位置をずらし、適切にアノーテーションを付加することができる。

【０２１５】
図３６は、文書処理装置１００により処理される文書の例を示し、図３７は、図３６に示した文書に対応づけられたアノーテーション用ファイルの例を示す。図３７に示したアノーテーション用ファイル７７ｄには、図３６に示した文書７５ｄにつけるアノーテーションの位置が記述されている。アノーテーション検出部７２がアノーテーション用ファイル７７ｄのアノーテーション位置を検出すると、アノーテーション表示部７４がその位置のアノーテーションを強調表示する。表示される画面は図３２に示した画面７９ａと同様である。

【０２１６】
（第５の実施の形態）
第５の実施の形態では、文書を表示する画面に重畳して表示される、アノーテーションを表示するためのレイヤーを別に用意し、そのアノーテーションレイヤーにアノーテーションを表示する。本実施の形態の文書処理装置の構成は、図３０に示した第１の実施の形態の文書処理装置１００の構成と同様である。

【０２１７】
アノーテーションレイヤーに表示される内容は、文書内に記録されてもよいし、文書とは別のファイル、例えば定義ファイルに記録されてもよい。アノーテーション検出部７２は、アノーテーションレイヤーに表示される内容が記述されたファイルを取得し、アノーテーション表示部７４は、第２描画部として機能し、アノーテーションレイヤー用ファイルに記述された内容をアノーテーションレイヤーにレイアウトする。また、アノーテーション表示部７４は、表示部として機能し、文書の処理を担当するプラグインユニットの表示部５６、６６などの第１描画部が描画した文書のレイヤーに、アノーテーションレイヤーを重畳して表示させる。アノーテーション付加部７６は、ＸＭＬ文書の編集中に、ユーザからアノーテーションの付加を指示されると、ユーザから指定された表示内容をアノーテーションレイヤー用ファイルに記録する。これにより、ユーザが、文書の表示画面に透明な画面を重ね、その上にアノーテーションを書き込むことができるような機能を提供することができる。アノーテーションレイヤーに表示される内容は、ＳＶＧを用いて記録されてもよいし、その他の形式で図形データとして記録されてもよいし、任意のデータ形式で記録されてもよい。文書が編集された場合、アノーテーション付加部７６は、その編集に追随してアノーテーションをずらして記録してもよい。また、文書のみが改変された場合、第４の実施の形態で説明した技術と同様に、文書の改編に追従してアノーテーションをずらしてもよい。

【０２１８】
図３８（ａ）（ｂ）（ｃ）は、それぞれ、文書が表示されたレイヤー、アノーテーションレイヤー、それらが重畳された表示画面の例を示す。図３８（ａ）は、図３６に示した７５ｄを表示した第１レイヤー７５ｅを示す。図３８（ｂ）は、アノーテーション表示部７４が、アノーテーションレイヤー用ファイルに記述された内容をレイアウトして生成した第２レイヤー７７ｅを示す。アノーテーション表示部７４は、さらに、第２レイヤー７７ｅを第１レイヤー７５ｅに重畳させて表示する。こうして表示された画面７９ｅを図３８（ｃ）に示す。

【０２１９】
（第６の実施の形態）
第６の実施の形態では、アノーテーションを文書に記録せずに、編集時にのみアノーテーションを表示する。本実施の形態の文書処理装置の構成は、図３０に示した第１の実施の形態の文書処理装置１００の構成と同様である。ただし、アノーテーションは編集時に表示されるだけで文書には記録されず、文書を開いたときにアノーテーションを検出する必要はないので、アノーテーション検出部７２は設けなくてもよい。

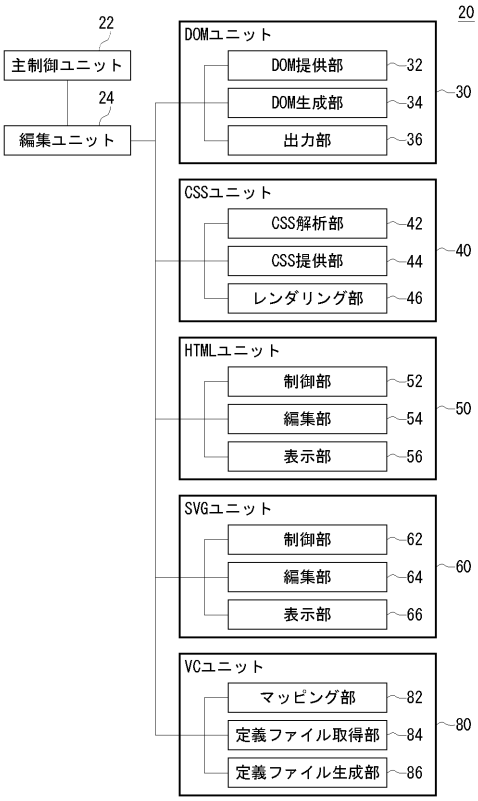
【０２２０】
アノーテーション付加部７６は、受付部として機能し、ユーザがアノーテーションの付加を指示したときに、指示された範囲を、アノーテーションを示す所定のデータ構造に変換する。アノーテーション表示部７４は、第２表示部として機能し、アノーテーションを示すデータを指定された方法で他のテキストと識別可能に表示する。文書の処理を担当するプラグインユニットの表示部５６、６６などが第１表示部として機能する。アノーテーションを示すデータ構造は、第１から第５の実施の形態で説明したいずれの方式を採用してもよい。例えば、アノーテーション用タグでマークアップしてもよいし、アノーテーションアンカーで囲んでもよいし、別のレイヤーにアノーテーションを描画して重畳させてもよい。

【０２２１】
文書が定義ファイルを用いて表示されている場合は、アノーテーション付加部７６は、デスティネーションツリーのみを変更してアノーテーションをつけてもよい。例えば、指定された範囲をＸＨＴＭＬの＜u＞タグにマッピングするようデスティネーションツリーを変更してもよい。この場合、ソースツリーは変更しないので、文書を保存したときにアノーテーションは残らない。これにより、文書を汚さずに、編集時にのみアノーテーションを表示することができる。

【０２２２】
アノーテーション付加部７６は、ソースツリーを変更してアノーテーションをつけてもよいが、この場合、文書を保存する際に、アノーテーションを除去してから保存する。この場合、アノーテーション付加部７６は、除去部として機能する。すなわち、アノーテーション付加部７６がアノーテーションをつけるために変更した箇所を全て元の状態に戻してから文書を保存する。
【０２２３】
（第７の実施の形態）
第７の実施の形態では、ウェブページなど複数のユーザが閲覧する文書に対してユーザがアノーテーションをつけるための技術を提案する。
【０２２４】
ウェブページなどの文書に対してユーザがつけたアノーテーションを記録するための定義ファイルを用意する。この定義ファイルは、第１から第５の実施の形態で説明したいずれの定義ファイルであってもよい。すなわち、アノーテーションタグ、アノーテーションアンカー、アノーテーション用タグセットなどのテンプレートやコマンドを含む定義ファイルであってもよいし、アノーテーション用レイヤーを描画するための定義ファイルであってもよいし、定義ファイルをアノーテーション用ファイルとして用いてもよい。本実施の形態の文書処理装置の構成は、図３０に示した第１の実施の形態の文書処理装置１００の構成と同様である。
【０２２５】
図３９は、ユーザが個別にアノーテーションをつける場合の構成例を示す。文書処理装置１００は、インターネット２９を介してウェブサーバ２５にアクセスし、文書２６と、文書２６を表示するための定義ファイル（ＶＣＤ）２７を取得して、ＶＣユニット８０により文書２６にＶＣＤ２７を適用して表示する。アノーテーションユニット７０が文書２６に対してアノーテーションをつけると、アノーテーションを記述したアノーテーションＶＣＤ２８が文書処理装置１００に保持される。文書処理装置１００は、文書２６を開くときに、まず、本来対応づけられていたＶＣＤ２７を適用した後、ユーザが用意したアノーテーションＶＣＤ２８を適用して表示する。または、文書処理装置１００は、本来対応づけられていたＶＣＤ２７に、アノーテーションＶＣＤ２８をインクルードして、新たな定義ファイルを生成し、その定義ファイルを文書２６に適用してもよい。
【０２２６】
図４０は、複数のユーザがアノーテーションを共有する場合の構成例を示す。この場合、アノーテーションＶＣＤ２８もウェブサーバ２５に保持され、文書２６がユーザの文書処理装置１００に送信されるときに、アノーテーションＶＣＤ２８も送信される。ユーザがアノーテーションユニット７０により文書２６にアノーテーションをつけると、アノーテーションＶＣＤ２８にアノーテーションが記録される。変更されたアノーテーションＶＣＤ２８が保存されると、ウェブサーバ２５のアノーテーションＶＣＤ２８が更新される。これにより、複数のユーザが同一の文書２６にアノーテーションをつけることができるので、複数のユーザが文書に落書きすることができるようなサービスを提供することができる。
【０２２７】
以上、本発明を実施の形態をもとに説明した。この実施の形態は例示であり、それらの各構成要素や各処理プロセスの組合せにいろいろな変形例が可能なこと、またそうした変形例も本発明の範囲にあることは当業者に理解されるところである。
【０２２８】
実施の形態では、ＸＭＬ文書を処理する例について説明したが、本実施の形態の文書処理装置１００は、他のマークアップ言語、例えば、ＳＧＭＬ、ＨＴＭＬなどで記述された文書も同様に処理可能である。
【産業上の利用可能性】
【０２２９】

本発明は、マークアップ言語により構造化されたデータを処理する装置に利用することができる。

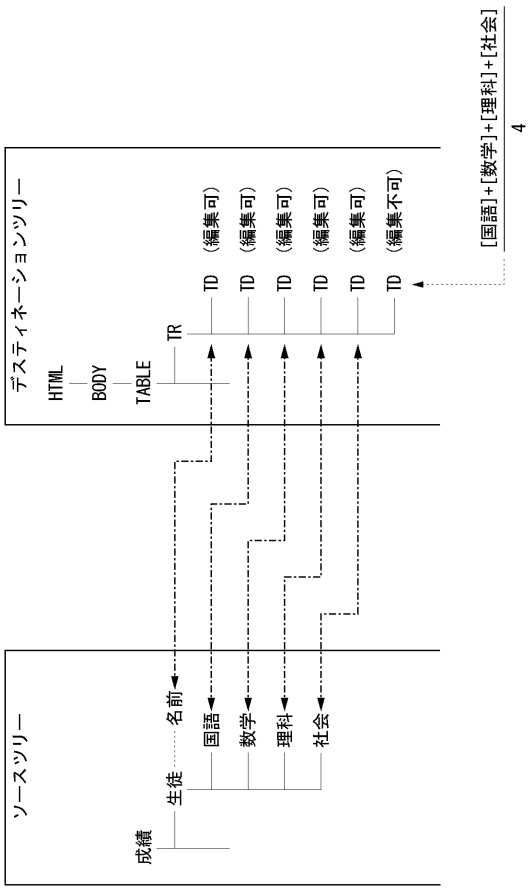
【図１】



【図２】

```
<?xml version="1.0" ?>

<?com.xfytec.vocabulary-connection href="records.vcd" ?>
<成績 xmlns="http://xmlns.xfytec.com/sample/records">
  <生徒 名前="A">
    <国語>90</国語>
    <数学>50</数学>
    <理科>75</理科>
    <社会>60</社会>
  </生徒>
  <生徒 名前="B">
    <国語>45</国語>
    <数学>60</数学>
    <理科>55</理科>
    <社会>50</社会>
  </生徒>
  <生徒 名前="C">
    <国語>55</国語>
    <数学>45</数学>
    <理科>95</理科>
    <社会>40</社会>
  </生徒>
  <生徒 名前="D">
    <国語>25</国語>
    <数学>35</数学>
    <理科>40</理科>
    <社会>15</社会>
  </生徒>
</成績>
```

【図３】



【図４（ａ）】

```
<?xml version="1.0"?>

<vc:vcd xmlns:vc="http://xmlns.xfytec.com/vcd"
        xmlns:src="http://xmlns.xfytec.com/sample/records"
        xmlns="http://www.w3.org/1999/xhtml"
        version="1.0">

<!-- Commands -->
<vc:command name="生徒の追加">
  <vc:insert-fragment
      target="ancestor-or-self::src:生徒"
      position="after">
    <src:生徒/>
  </vc:insert-fragment>
</vc:command>
<vc:command name="生徒の削除">
  <vc:delete-fragment target="ancestor-or-self::src:生徒" />
</vc:command>

<!-- Templates -->
<vc:vc-template match="src:成績" name="成績表" >

  <vc:ui command="生徒の追加">
    <vc:mount-point>
      /MenuBar/成績表/生徒の追加
    </vc:mount-point>
  </vc:ui>
  <vc:ui command="生徒の削除">
    <vc:mount-point>
      /MenuBar/成績表/生徒の削除
    </vc:mount-point>
  </vc:ui>

  <html>
    <head>
      <title>成績表</title>
      <style>
        td,th {
          text-align:center;
          border-right:solid black 1px;
          border-bottom:solid black 1px;
          border-top:none 0px;
          border-left:none 0px;
        }
        table{
          border-top:solid black 2px;
          border-left:solid black 2px;
          border-right:solid black 1px;
          border-bottom:solid black 1px;
          border-spacing:0px;
        }
```

【図４（ｂ）】

```
        tr{
          border:none;
        }
        .data{
          padding:0.2em 0.5em;
        }
      </style>
    </head>
    <body>
      <h1>成績一覧</h1>
      <table>
        <tr><th><div class="data">名前</div></th>
        <th></th>
        <th><div class="data">国語</div></th>
        <th><div class="data">数学</div></th>
        <th><div class="data">理科</div></th>
        <th><div class="data">社会</div></th>
        <th></th>
        <th><div class="data">平均</div></th></tr>
        <vc:apply-templates select="src:生徒" />
      </table>
    </body>
  </html>
</vc:vc-template>

<vc:template match="src:生徒">
  <tr>
    <td><div class="data">
      <vc:text-of select="@名前" fallback="名無し"/>
    </div></td>
    <td></td>
    <td><div class="data">
      <vc:text-of select="src:国語" fallback="0" type="vc:integer" />
    </div></td>
    <td><div class="data">
      <vc:text-of select="src:数学" fallback="0" type="vc:integer" />
    </div></td>
    <td><div class="data">
      <vc:text-of select="src:理科" fallback="0" type="vc:integer" />
    </div></td>
    <td><div class="data">
      <vc:text-of select="src:社会" fallback="0" type="vc:integer" />
    </div></td>
    <td></td>
    <td><div class="data">
      <vc:value-of
        select="(src:国語 + src:数学 + src:理科 + src:社会) div 4" />
    </div></td>
  </tr>
</vc:template>

</vc:vcd>
```

【図５】

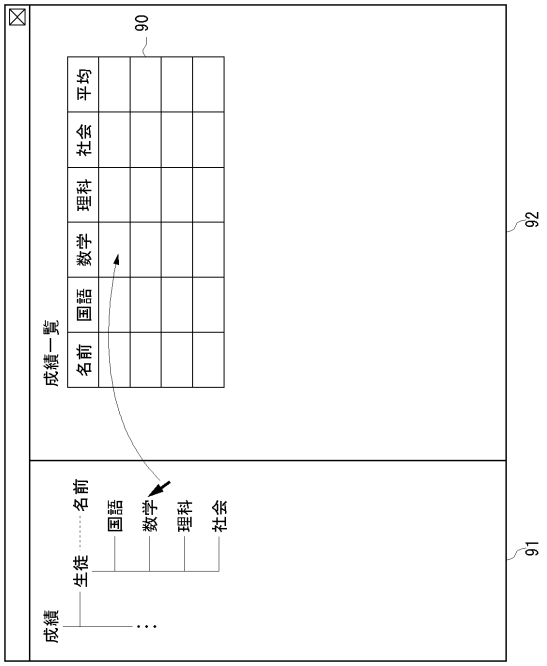
sample.xml

成績一覧

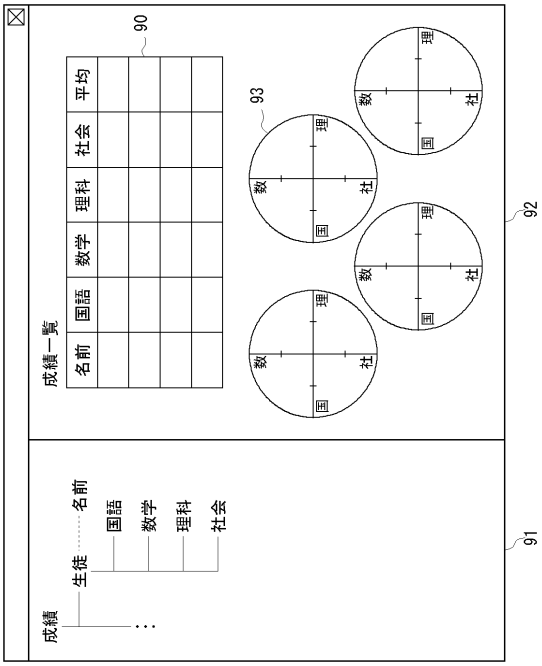
90

名前	国語	数学	理科	社会	平均
A	90	50	75	60	68.8
B	45	60	55	50	52.5
C	55	45	95	40	58.8
D	25	35	40	15	28.8

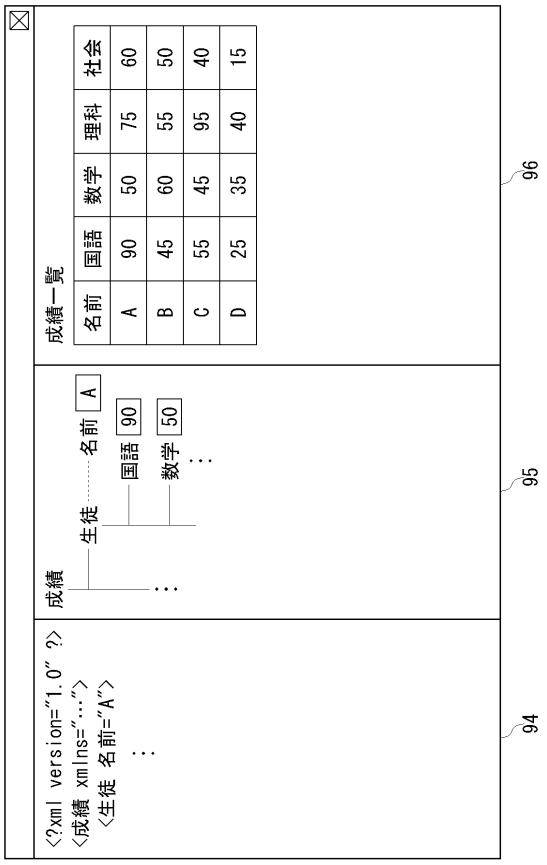
【図６】



【図７】



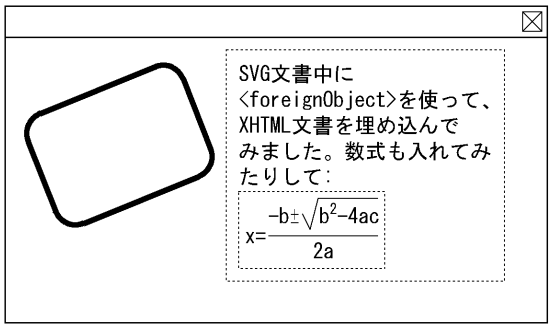
【図８】



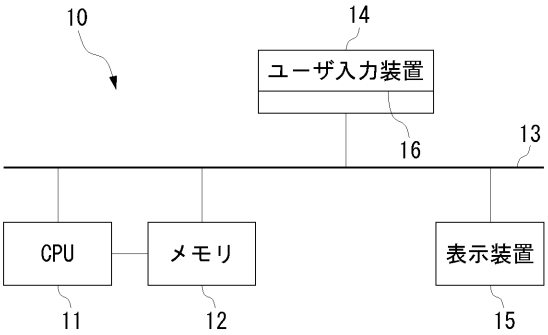
【図９】

```
<?xml version="1.0" ?>

<svg xmlns="http://www.w3.org/2000/svg"
     width="400" height="200"
     viewBox="0 0 400 200"
>
  <rect x="-15" y="65" width="150" height="100" rx="20"
        transform="rotate(-20)"
        style="fill:none; stroke:purple; stroke-width:10"
  />
  <foreignObject x="190" y="10" width="200" height="200">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head><title /></head>
      <body bgcolor="#FFFFCC" text="darkgreen">
        <div style="font-size:12pt">
          SVG文書中に&lt;foreignObject&gt;を使って、
          XHTML文書を埋め込んでみました。
          数式も入れてみたりして:
          <div>
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <mi>x</mi>
              <mo>=</mo>
              <mfrac>
                <mrow>
                  <mo>−</mo>
                  <mi>b</mi>
                  <mo>±</mo>
                  <msqrt>
                    <mrow>
                      <msup>
                        <mi>b</mi>
                        <mn>2</mn>
                      </msup>
                      <mo>−</mo>
                      <mn>4</mn>
                      <mi>a</mi>
                      <mi>c</mi>
                    </mrow>
                  </msqrt>
                </mrow>
                <mrow>
                  <mn>2</mn>
                  <mi>a</mi>
                </mrow>
              </mfrac>
            </math>
          </div><!-- math -->
        </div>
      </body>
    </html>
  </foreignObject>
</svg>
```

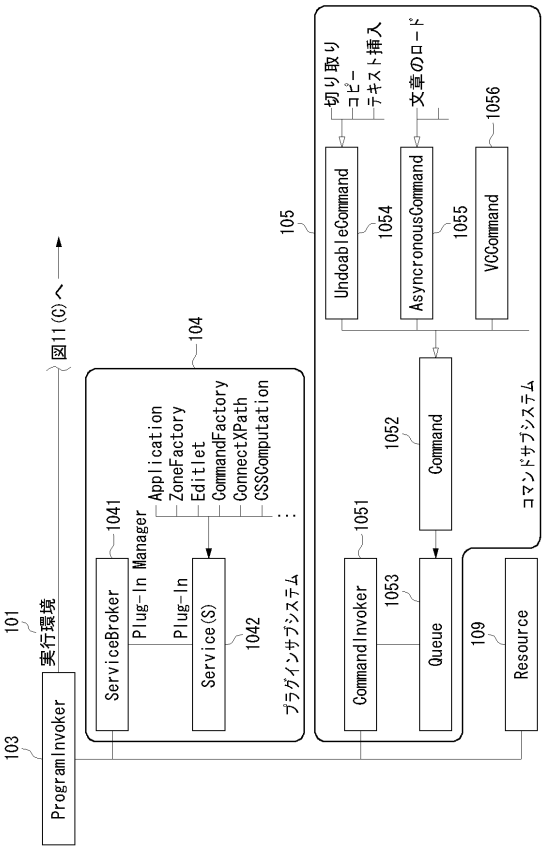
【図１０】



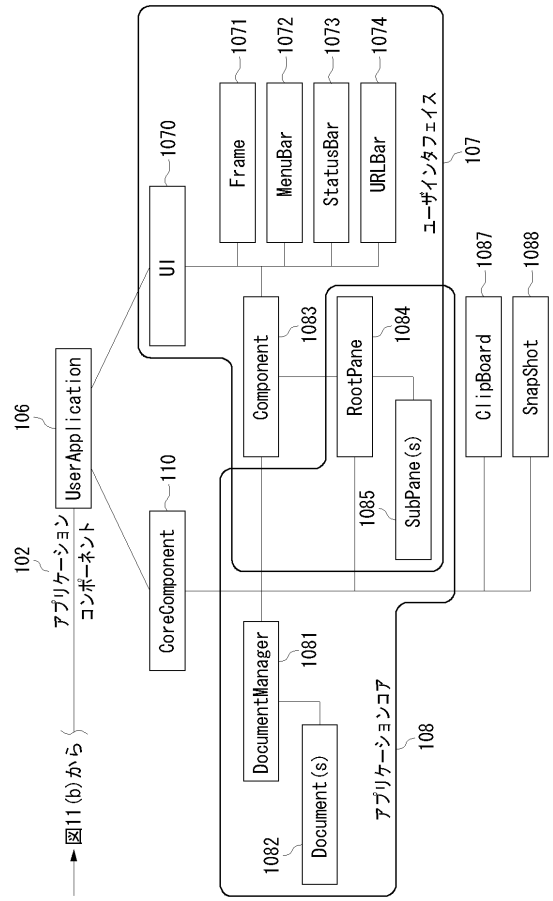
【図１１（ａ）】



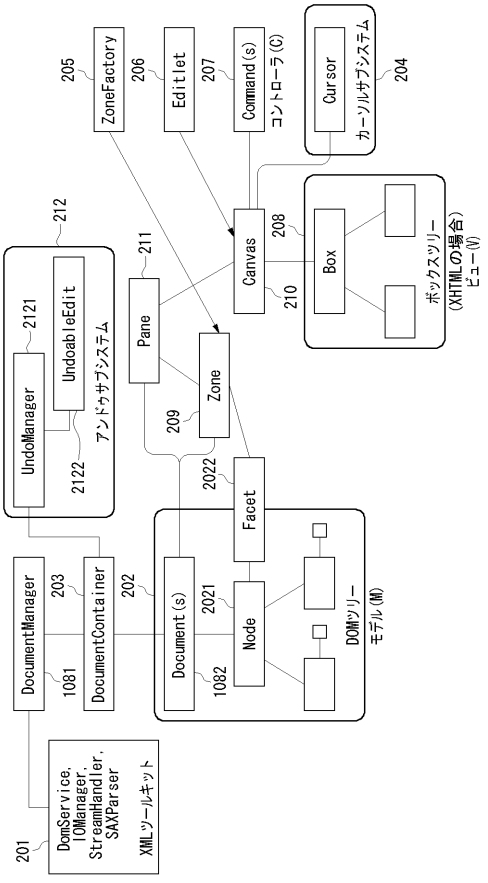
【図１１（ｂ）】



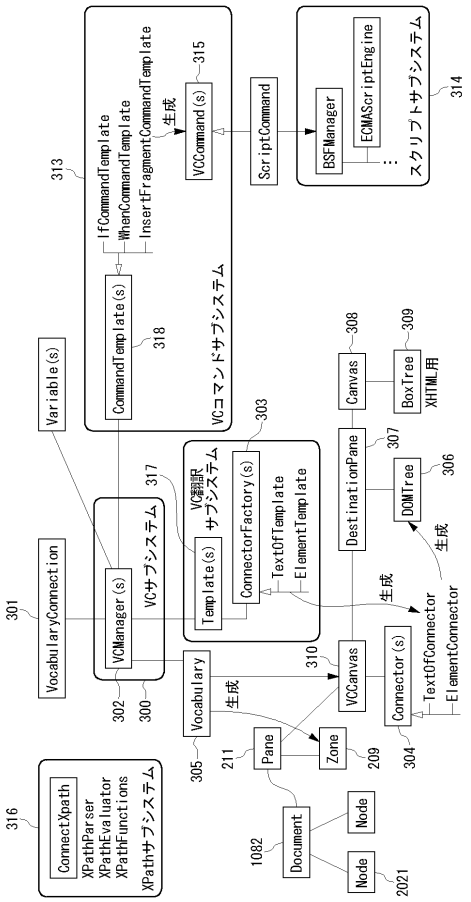
【図１１（ｃ）】



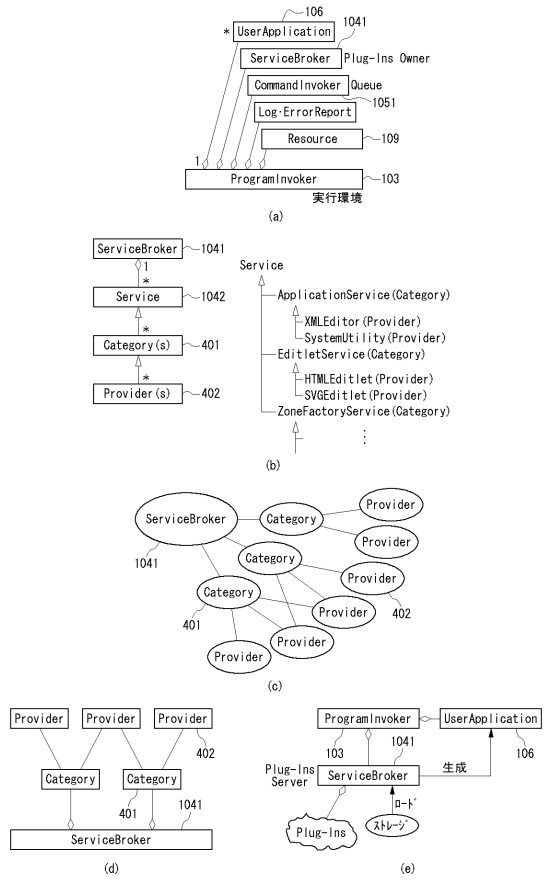
【図１２】



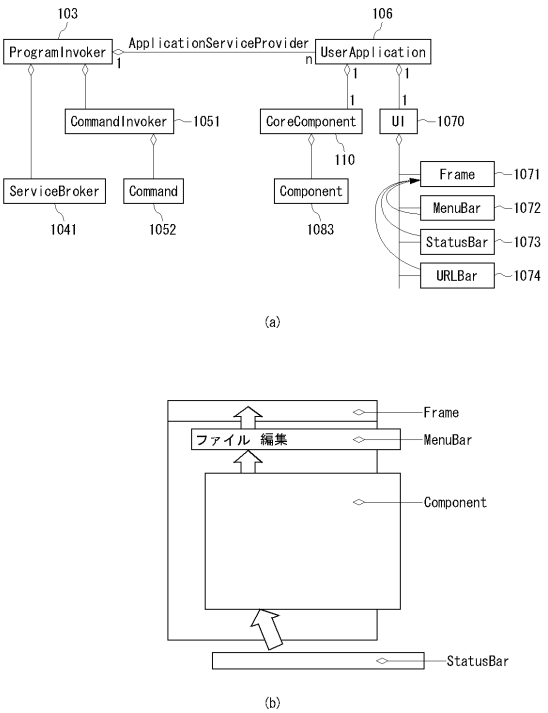
【図１３】



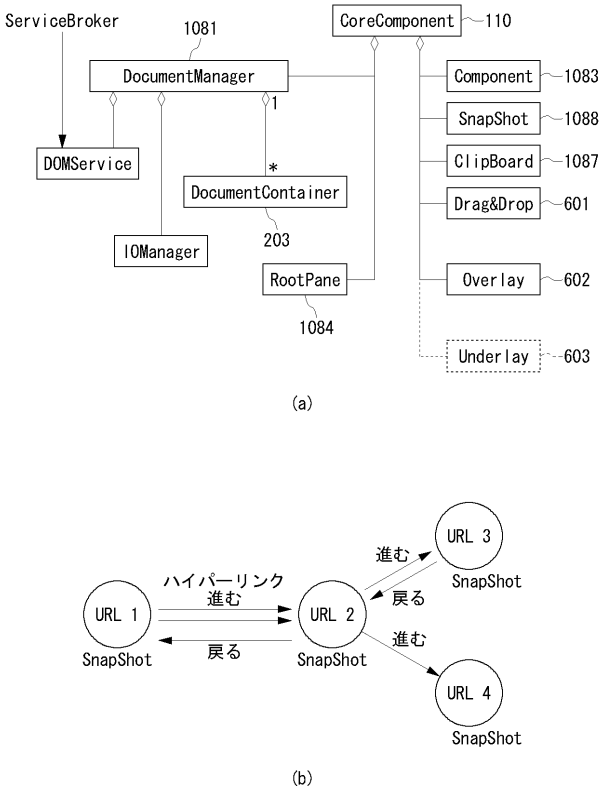
【図１４】



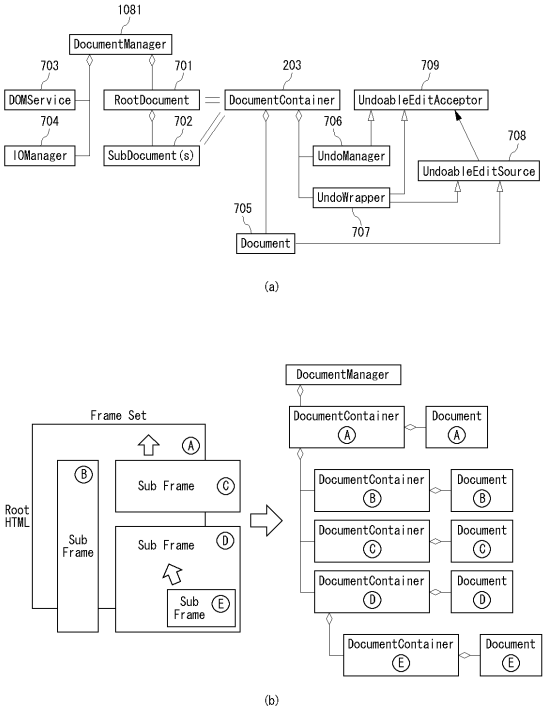
【図１５】



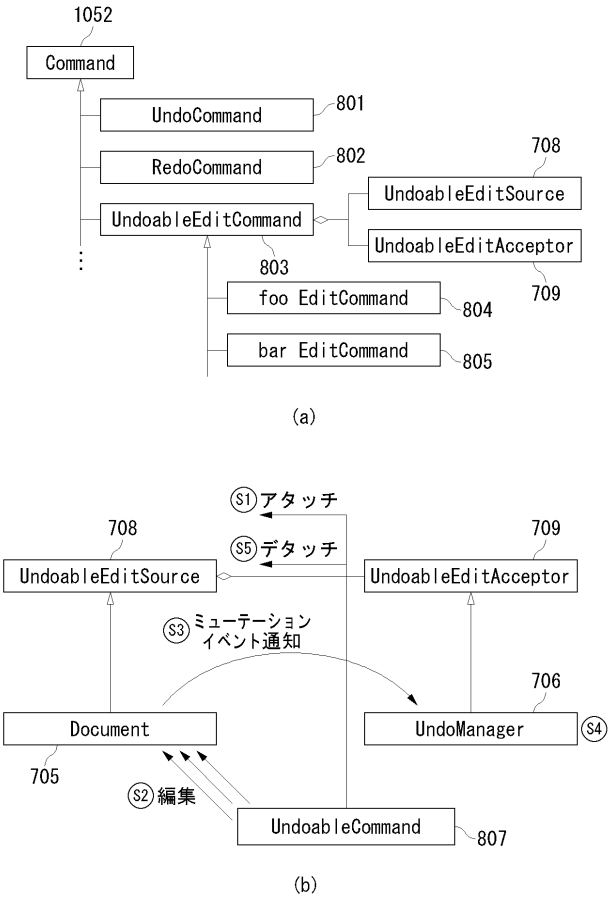
【図１６】



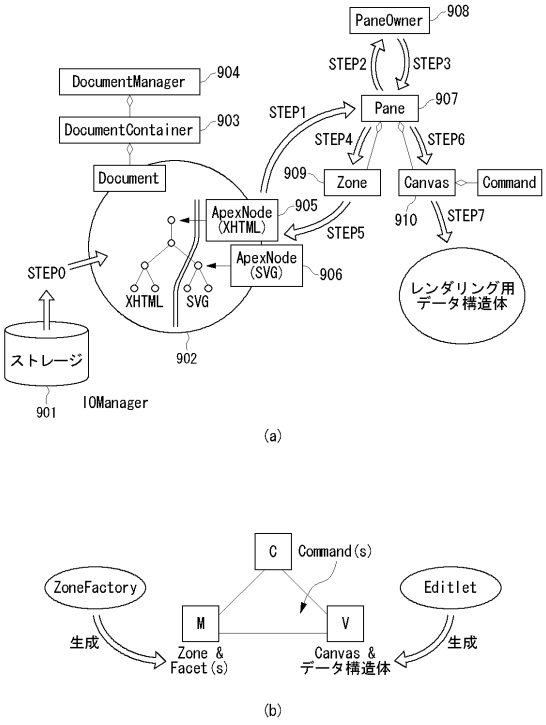
【図１７】



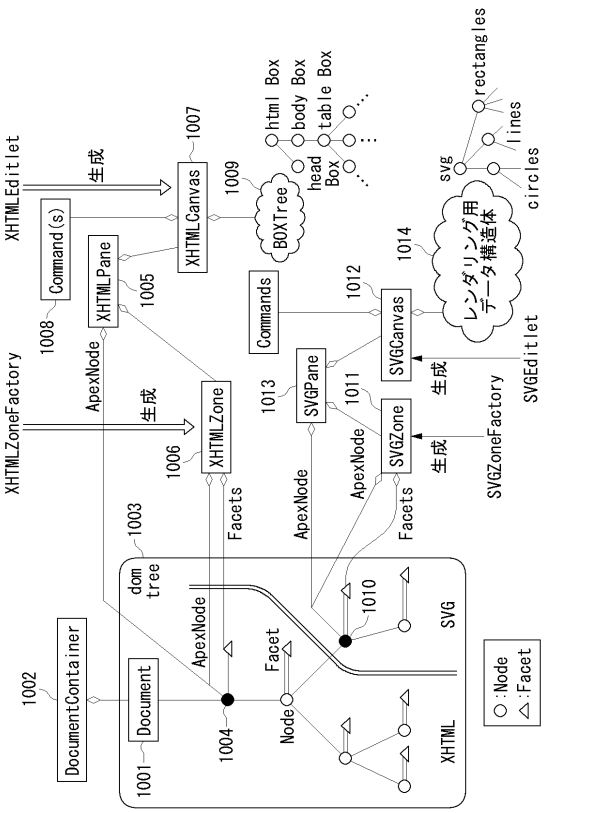
【図１８】



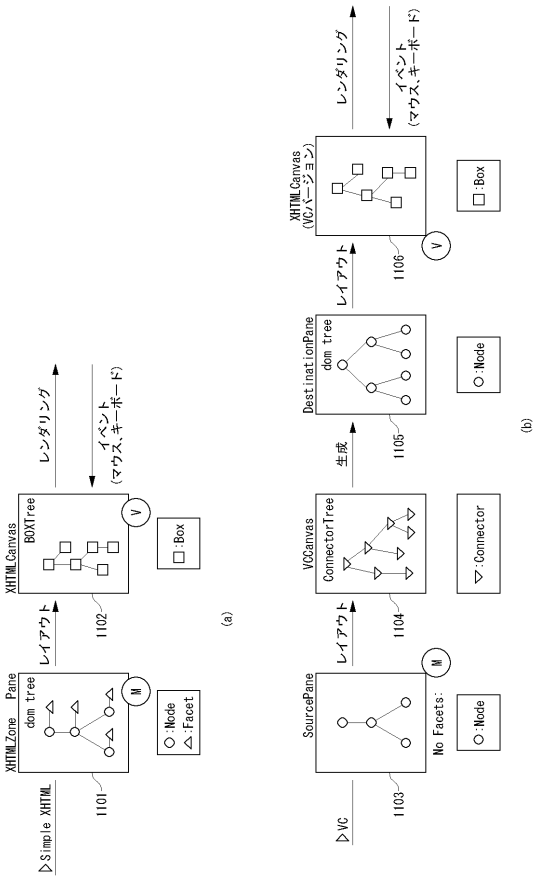
【図１９】



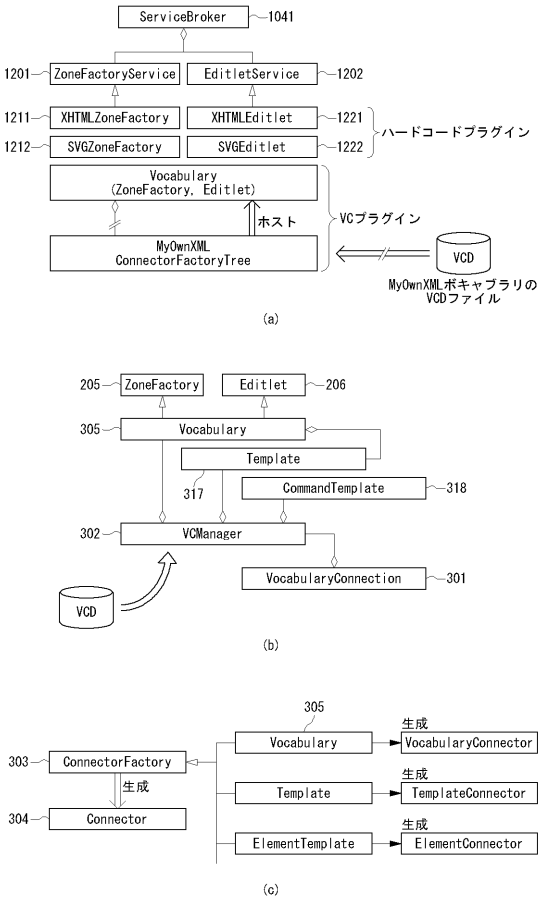
【図２０】



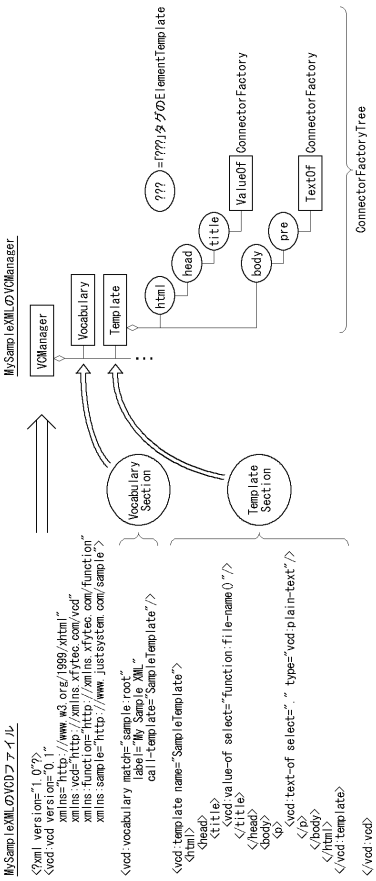
【図２１】



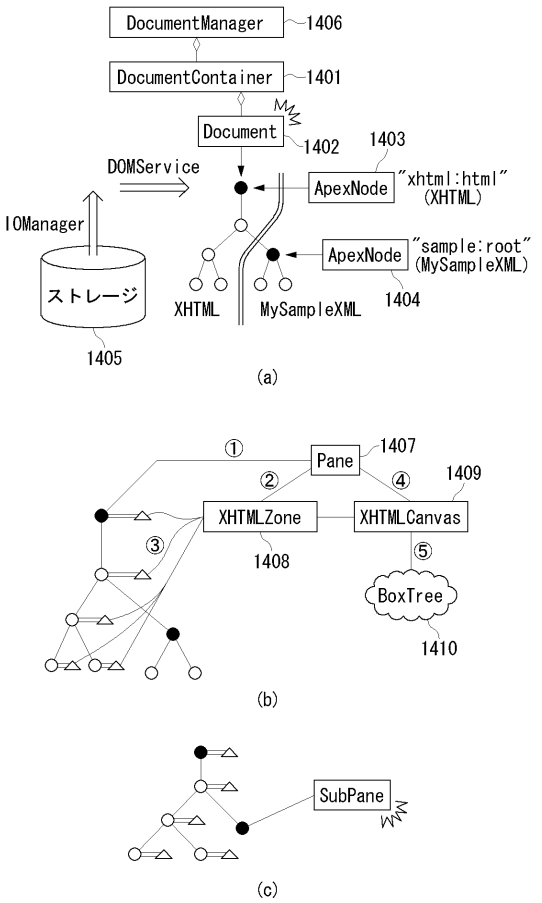
【図２２】



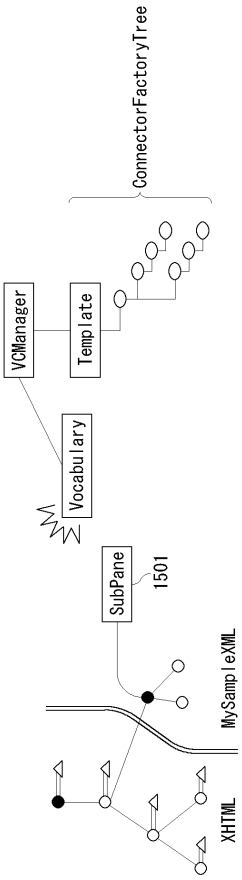
【図２３】



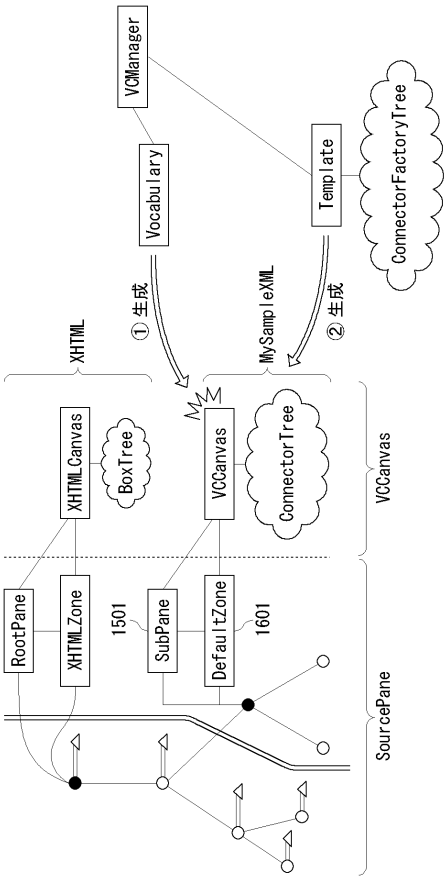
【図２４】



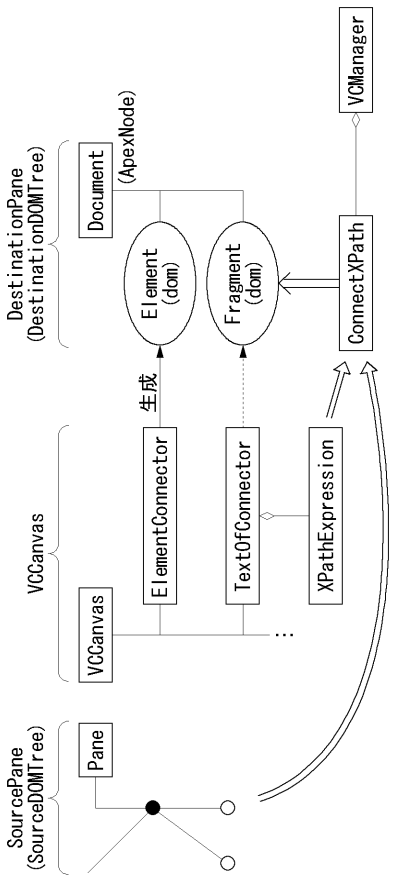
【図２５】



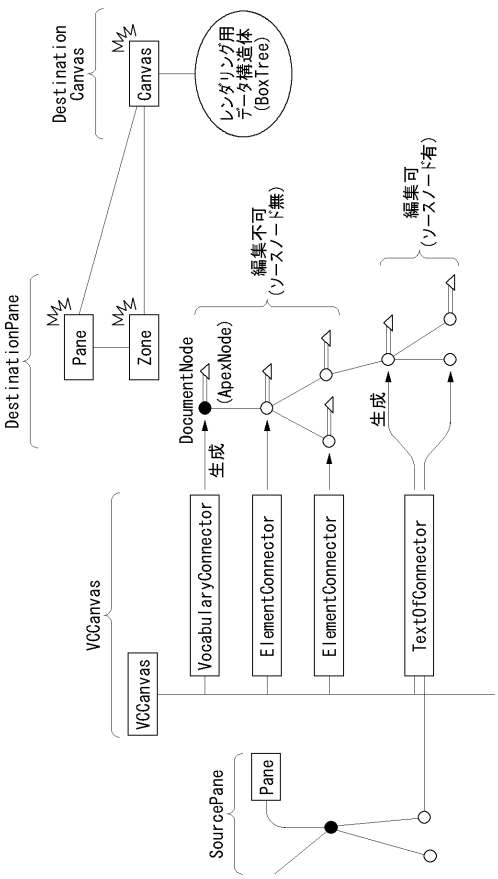
【図２６】



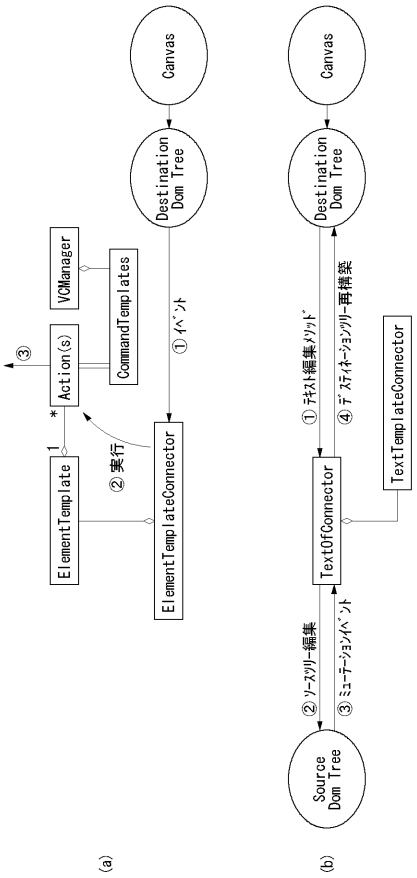
【図２７】



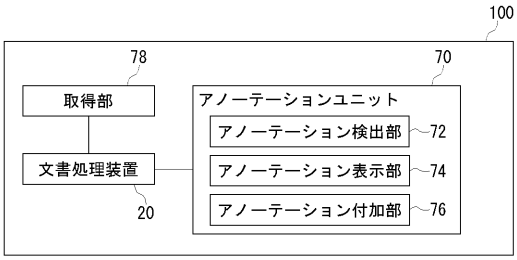
【図２８】



【図２９】



【図３０】

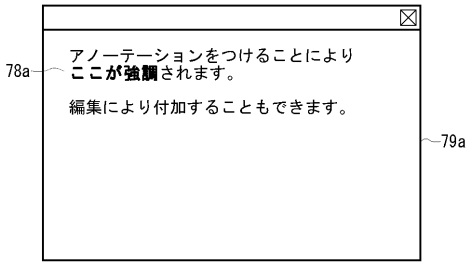


【図３１】

75a

```
<?xml version="1.0" ?>

<?com.xfytec vocabulary-connection href="sample.vcd" ?>
<sample:sample
        xmlns:sample="http://xmlns.xfytec.com/sample">
    <sample:item>
        アノーテーションをつけることにより
77a  <* type="bold">ここが強調</*>
        されます。
    </sample:item>
    <sample:item>
        編集により付加することもできます。
    </sample:item>
</sample:sample>
```

【図３２】

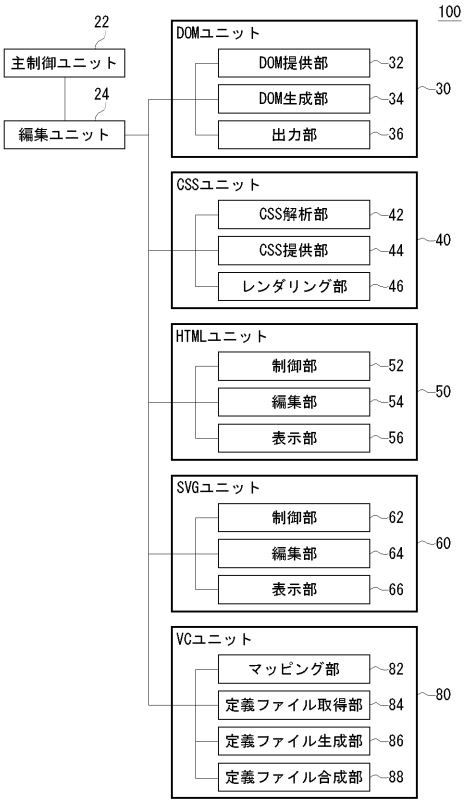


【図３３】

75b

```
<?xml version="1.0" ?>

<?com.xfytec vocabulary-connection href="sample.vcd" ?>
<sample:sample
    xmlns:sample="http://xmlns.xfytec.com/sample"
    xmlns:annotation="http://xmlns.xfytec.com/annotation">
    <sample:item>
        アノーテーションをつけることにより
77b     <annotation:start id="1" />
        ここが強調
        <annotation:end id="1" />
        されます。
    </sample:item>
    <sample:item>
        編集により付加することもできます。
    </sample:item>
</sample:sample>
```

【図３４】



【図３５】

75c

```
<?xml version="1.0" ?>

<?com.xfytec vocabulary-connection href="sample.vcd" ?>
<sample:sample
    xmlns:sample="http://xmlns.xfytec.com/sample"
    xmlns:annotation="http://xmlns.xfytec.com/annotation">
    <sample:item>
        アノーテーションをつけることにより
        <annotation:annotation>
77c     ここが強調
        </annotation:annotation>
        されます。
    </sample:item>
    <sample:item>
        編集により付加することもできます。
    </sample:item>
</sample:sample>
```

【図３６】

75d

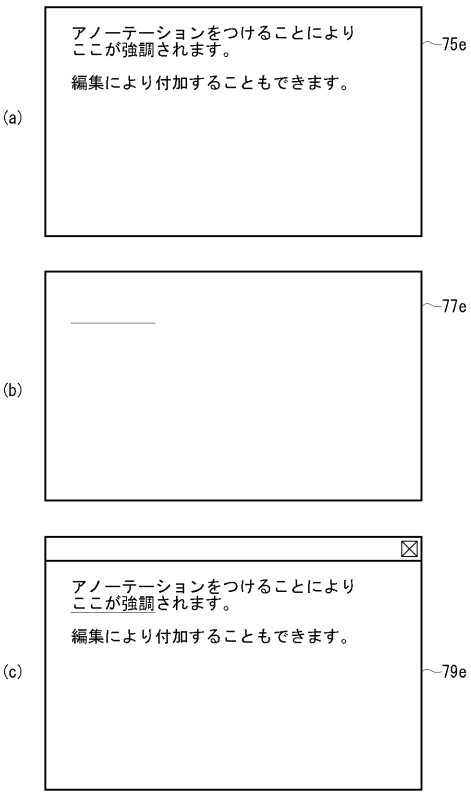
```
<?xml version="1.0" ?>

<?com.xfytec vocabulary-connection href="sample.vcd" ?>
<sample:sample
    xmlns:sample="http://xmlns.xfytec.com/sample">
    <sample:item>
        アノーテーションをつけることによりここが強調されます。
    </sample:item>
    <sample:item>
        編集により付加することもできます。
    </sample:item>
</sample:sample>
```

【図３７】

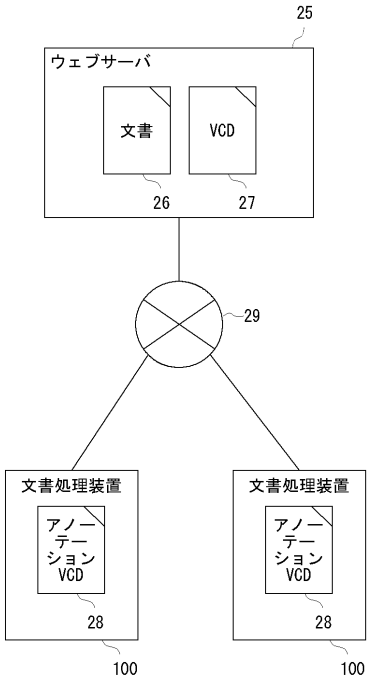
77d

```
<?xml version="1.0" ?>


<annotation:annotation
    xmlns:annotation="http://xmlns.xfytec.com/annotation"
    xmlns:sample="http://xmlns.xfytec.com/sample">

  <annotation:item id="1">
        <annotation:start
            node="/sample:sample/sample:item[1]"
            position="18" />
        <annotation:end
            node="/sample:sample/sample:item[1]"
            position="22" />
  </annotation:item>


</annotation:annotation>
```

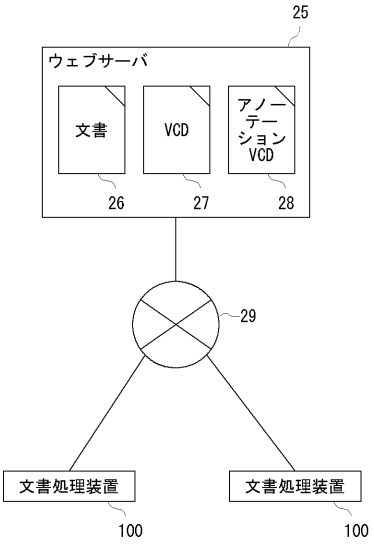
【図３８】



【図３９】



【図４０】


25
ウェブサーバ
文書
VCD
アノーテーションVCD
26
27
28
29
文書処理装置
文書処理装置
100
100

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2005/020028

A. CLASSIFICATION OF SUBJECT MATTER
G06F17/21(2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F17/21(2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2005
Kokai Jitsuyo Shinan Koho 1971-2005 Toroku Jitsuyo Shinan Koho 1994-2005

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JP 2002-207727 A (Fujitsu Ltd.), 26 July, 2002 (26.07.02), Full text; all drawings & EP 1205859 A2 & US 2002-059343 A1	1-7

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 December, 2005 (12.12.05)	27 December, 2005 (27.12.05)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (April 2005)

国際調査報告

国際出願番号 PCT/JP2005/020028

A. 発明の属する分野の分類（国際特許分類（IPC））

Int.Cl. *G06F17/21* (2006.01)

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））

Int.Cl. *G06F17/21* (2006.01)

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	1922－1996年
日本国公開実用新案公報	1971－2005年
日本国実用新案登録公報	1996－2005年
日本国登録実用新案公報	1994－2005年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献のカテゴリー＊	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
X	JP 2002-207727 A（富士通株式会社）2002.07.26, 全文全図 & EP 1205859 A2 & US 2002-059343 A1	1-7

☐ C欄の続きにも文献が列挙されている。

☐ パテントファミリーに関する別紙を参照。

＊ 引用文献のカテゴリー

「A」特に関連のある文献ではなく、一般的技術水準を示すもの

「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの

「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）

「O」口頭による開示、使用、展示等に言及する文献

「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献

「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの

「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの

「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの

「&」同一パテントファミリー文献

国際調査を完了した日 12. 12. 2005	国際調査報告の発送日 27. 12. 2005
国際調査機関の名称及びあて先 日本国特許庁（ISA/JP） 郵便番号100－8915 東京都千代田区霞が関三丁目4番3号	特許庁審査官（権限のある職員） 5M 4233 長 由紀子 電話番号 03－3581－1101 内線 3599

様式PCT/ISA/210（第2ページ）（2005年4月）

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IS,IT,LT,LU,LV,MC,NL,PL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KM,KN,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,LY,MA,MD,MG,MK,MN,MW,MX,MZ,NA,NG,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,SM,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

（注）この公表は、国際事務局（ＷＩＰＯ）により国際公開された公報を基に作成したものである。なおこの公表に係る日本語特許出願（日本語実用新案登録出願）の国際公開の効果は、特許法第１８４条の１０第１項(実用新案法第４８条の１３第２項）により生ずるものであり、本掲載とは関係ありません。