(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2007 年4 月19 日 (19.04.2007)

PCT

(10) 国際公開番号
WO 2007/043661 A1

(51) 国際特許分類:
G06F 3/14 (2006.01) *G06F 17/21* (2006.01)

(21) 国際出願番号: PCT/JP2006/320488

(22) 国際出願日: 2006 年10 月13 日 (13.10.2006)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願 2005-301006
2005 年10 月14 日 (14.10.2005) JP

(71) 出願人 (米国を除く全ての指定国について): 株式会社ジャストシステム (JUSTSYSTEMS CORPORATION) [JP/JP]; 〒7710189 徳島県徳島市川内町平石若松１０８番地４ Tokushima (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 松家 勝弘 (MATSUKA, Katsuhiro) [JP/JP]; 〒7710189 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 Tokushima (JP).

(74) 代理人: 森下 賢樹 (MORISHITA, Sakaki); 〒1500021 東京都渋谷区恵比寿西２－１１－１２ Tokyo (JP).

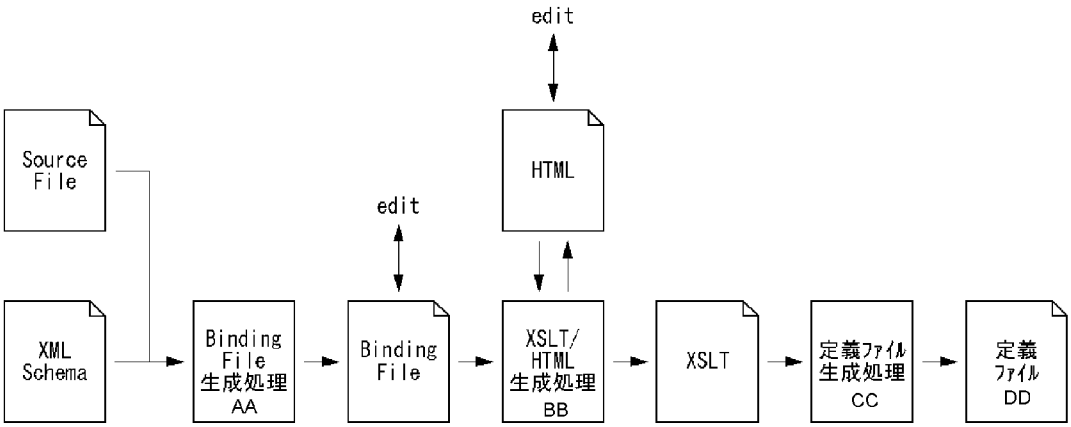
(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY,

[続葉有]

(54) Title: DATA PROCESSING DEVICE

(54) 発明の名称: データ処理装置



AA- BINDING FILE CREATION PROCESSING
BB- XSLT/HTML CREATION PROCESSING
CC- DEFINITION FILE CREATION PROCESSING
DD- DEFINITION FILE

(57) Abstract: To design the editing user interface screen of a structured document file simply. A binding file is created from a schema file for defining the element structure of a source file. The editing display layout of the source file is designed through the binding file, and its result is stored as a layout file. The user can also edit the layout file. An XSLT file is created from the layout file and the binding file, and a definition file for creating the editing user interface screen of the source file is created from the XSLT file.

(57) 要約: 構造化文書ファイルの編集用ユーザインタフェース画面を簡易にデザインする。 ソースファイルの要素構造を定義するスキーマファイルから、バインディ

[続葉有]

WO 2007/043661 A1

KG, KZ, MD, RU, TJ, TM), ヨーロッパ (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), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告書

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

ングファイルを生成する。バインディングファイルを介して、ソースファイルの編集用表示レイアウトをデザインし、その結果がレイアウトファイルとして保存される。ユーザはレイアウトファイルを編集することもできる。レイアウトファイルとバインディングファイルからＸＳＬＴファイルが生成され、ＸＳＬＴファイルからソースファイルの編集用ユーザインタフェース画面を生成するための定義ファイルが生成される。

明 細 書

データ処理装置

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構造化文書ファイルを処理するための技術に関する。

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、データ処理装置である。

この装置は、外部のデータベースに対して、データを要求するためのクエリ(Query)を送信し、クエリによって指定されたデータを構造化文書ファイルとして取得し、その構造化文書ファイルを参照して要素構造を特定することによってスキーマ情報を取得し、構造化文書ファイルの編集用ユーザインタフェース画面を表示するための定

義データをスキーマ情報に基づいて生成する。

[0006] 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

[0007] 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

[0008] [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]本実施例における定義ファイル生成過程を説明するための模式図である。

[図31]本実施例におけるスキーマファイルを示す図である。

[図32]図31のスキーマファイルに対応するソースファイルを示す図である。

[図33]図31のスキーマファイルと図32のソースファイルに基づいて生成される定義ファイルを示す図である。

[図34]バインディングファイルの編集画面を示す図である。

[図35]図34におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。

[図36]図35における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

[図37]バインディングファイルの編集画面の別例を示す図である。

[図38]図37におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。

[図39]図38における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

[図40]バインディングファイルの編集画面の更に別例を示す図である。

[図41]図40におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。

[図42]図41における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

[図43]バインディングファイルの編集画面の更に別例を示す図である。

[図44]図43におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。

[図45]図44における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

[図46]定義ファイル生成過程を更に説明するための模式図である。

[図47]本実施例におけるデータ処理装置とXMLデータベースとの連携方法を説明するための模式図である。

符号の説明

[0009] 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、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。

発明を実施するための最良の形態

[0010] (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、ハードウェ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ェアのみ、ソフトウェアのみ、またはそれらの組み合わ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。

[0011] 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。

[0012] 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。

[0013] 編集ユニット24は、ユーザインタ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。

[0014] 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。

[0015] 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。

[0016] 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)とよ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フェースを提供する。

[0017] 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)とよぶ。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フェースを提供する。

[0018] 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。

[0019] 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、その定義にしたがって、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。

[0020] 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」とよぶ。

[0021] 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。

[0022] 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、それぞれ対応づける。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。

[0023] 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づけられている。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。

[0024] 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。例えば、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。

[0025] 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。

[0026] 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。

[0027] 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。

[0028] 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。

[0029] 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すべての表示部が、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。

[0030] 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。

[0031] 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。

[0032] 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。

[0033] 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、更に、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。更に、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。

[0034] 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フェースを提供することができる。

[0035] 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。

[0036] 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ときに、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。

[0037] 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。

[0038] 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。

[0039] 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ともなって、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。

[0040] 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。

[0041] 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。

[0042] 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。

[0043] 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。

[0044] 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。

[0045] 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。

[0046] 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。

[0047] 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/)

[0048] 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、DOM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。

[0049] 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。

[0050] 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]

[0051] 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。

[0052] 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。

[0053] 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。

[0054] 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。

[0055] 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。

[0056] 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。

[0057] 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。

[0058] 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。

[0059] 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。

[0060] 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。

[0061] 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。

[0062] 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。

[0063] 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。

[0064] 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。

[0065] 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。

[0066] 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。

[0067] 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。

[0068] 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。

[0069] 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。

[0070] 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Core Component（コアコンポーネント）110を含む。

[0071] 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。User Application106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。

[0072] 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。

[0073] 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。

[0074] 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。

[0075] 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。

[0076] 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。

[0077] 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。

[0078] 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。

[0079] 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。

[0080] 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。

[0081] 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。

[0082] 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。

[0083] 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。

[0084] 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。

[0085] 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。

[0086] 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。

[0087] 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。Canvas(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。

[0088] 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。

[0089] 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。

[0090] 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。

[0091] 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。

[0092] 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。

[0093] 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。

[0094] 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。

[0095] 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。

[0096] 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。

[0097] 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。

[0098] 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。

[0099] 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。

[0100] 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。

[0101] 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。

[0102] 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。

[0103] 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。

[0104] 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。

[0105] 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。

[0106] 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。

[0107] 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。

[0108] 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。

[0109] 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。

[0110] 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。

[0111] 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。

[0112] 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。

[0113] 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。

[0114] 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。

[0115] 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。

[0116] 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。

[0117] 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。

[0118] 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。VCCo

mmand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。

[0119] 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。

[0120] 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。

[0121] 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。

[0122] 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。

[0123] 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。

[0124] 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。

[0125] 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。

[0126] 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。

[0127] 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。

[0128] 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。

[0129] 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。

[0130] 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。

[0131] 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)及び図11(c)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。

[0132] 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ServiceProvider402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。

[0133] 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。

[0134] 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。

[0135] 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。

[0136] 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。

[0137] 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。

[0138] 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。

[0139] 図15(b)は、Frame1071、MenuBar1072、及びStatusBar1073の関係を示す。

[0140] 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。

[0141] 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。

[0142] 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。

[0143] 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。

[0144] 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。

[0145] 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。

[0146] 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。

[0147] 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。

[0148] 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。

[0149] 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CommandInvoker1051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。

[0150] 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。

[0151] 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。

[0152] 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。

[0153] 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。

[0154] 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、DOM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。

[0155] 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CanvasFactoryであるEditletとを返す。

[0156] 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。

[0157] 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。

[0158] 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。

[0159] 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。

[0160] 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。

[0161] 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。

[0162] 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。

[0163] 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。

[0164] 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。

[0165] 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。

[0166] 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。

[0167] 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。

[0168] 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。

[0169] 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。

[0170] 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。

[0171] 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。

[0172] 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。

[0173] 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。

[0174] 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。

[0175] 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。

[0176] 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。

[0177] 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。

[0178] 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。

[0179] 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。

[0180] 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。

[0181] 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。

[0182] 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。

[0183] (実施の形態)

本実施例におけるデータ処理装置は、前提技術で説明した文書処理装置20の機能をその一部として含み、前提技術で説明したボキャブラリコネクションにより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示す定義ファイルを簡易に生成すること

ができる。まず、図30において本実施例における定義ファイル生成過程を概観したあと、図12以降において表示態様を中心として説明する。

[0184] 図30は、本実施例における定義ファイル生成過程を説明するための模式図である。

データ処理装置は、編集対象となるXML文書ファイル(以下、「ソースファイル」とよぶ)と、ソースファイルの要素構造を定義するスキーマファイルを取得する。ここでいうスキーマファイルとは、XML－Schema、DTD(Document Type Definition)などの仕様にしたがって記述される。生成物としての定義ファイルは、このソースファイルを編集するために適切な表示レイアウト情報をもつデスティネーションファイルを生成するためのファイルである。デスティネーションファイルは、前提技術で説明したデスティネーションツリーをファイル化したものであるといえる。

[0185] データ処理装置は、スキーマファイルがあるときには、スキーマファイルからバインディングファイル(Binding File)を生成する。バインディングファイルは、デスティネーションファイルにおける表示レイアウトを編集するために使用される。スキーマファイルがなければ、データ処理装置はソースファイル本体から要素とその構造を抽出してバインディングファイルを生成する。この場合、データ処理装置は、ソースファイルのルート要素からツリートラバース(tree traverse)方式によって子要素を抽出することにより、要素とその構造を抽出する。

更に、バインディングファイルによって、ソースファイルの要素に関する規則を再定義可能である。たとえば、ソースファイルからバインディングファイルを生成する場合において、ソースファイル中の要素Aは、子要素Bを4つ持っているとする。このとき、バインディングファイルには、要素Aの持ち得る子要素Bの数は4個までであるという規則が一応記載される。ユーザはバインディングファイルが提供するメソッドを介して、この要素Aと子要素Bに関する規則を再定義できる。たとえば、要素Aが持ち得る子要素Bの数は、1～10までとして定義してもよい。スキーマファイルからバインディングファイルが生成される場合であっても、スキーマファイルにおいて定義されている規則の範囲内において、このような要素に関する規則を再定義できてもよい。

このようにバインディングファイルは、要素に関する規則と、それを定義するための

機能も提供する。

なお、以下においては、スキーマファイルからソースファイルの要素構造を示すスキーマ情報を取得するものとして説明する。

[0186] ユーザは、データ処理装置にてバインディングファイルを編集可能である。バインディングファイルに対して、ユーザはデスティネーションファイルの基本的な表示レイアウトをGUI(Graphical User Interface)により設定できる。こうして、バインディングファイルで定義された表示レイアウト情報をスキーマ情報の各要素に適用することによりレイアウトファイルが生成される。レイアウトファイルは、スキーマファイルに含まれていた各要素の具体的な表示レイアウトを示すHTMLファイルである。なお、レイアウトファイルは、タグによって構造化されるタイプの構造化文書ファイルに限られる必要はなく、表計算アプリケーションやプレゼンテーション用アプリケーションのように表示レイアウト情報を含むファイルであればよい。ユーザはレイアウトファイルそのものを編集することにより、表示レイアウトを更に精緻に編集できる。本実施例においては、バインディングファイルによって、デスティネーションファイルの表示レイアウトの基本設定を行い、レイアウトファイルによってその詳細設定を行う。

[0187] バインディングファイルに示されていた要素とレイアウトファイルの表示領域との対応関係から、ソースファイルとデスティネーションファイル間のデータ変換形式を定めるためのXSLTファイルが生成される。最後に、このXSLTファイルに基づいて、バインディングファイルに対応するソースファイルと、レイアウトファイルに対応するデスティネーションファイルの対応関係を示す定義ファイルが生成される。

以下、ユーザインタフェースを中心としてこれらの処理の流れを説明する。

[0188] 図31は、本実施例におけるスキーマファイルを示す図である。

ここに示すスキーマファイルは、後述の図13に示すソースファイルがしたがうべき要素構造に関する規則を記述している。また、このスキーマファイルは、XML－Schemaという仕様にしたがって記述されている。

同図においては、たとえば、上から3行目において、「customerList」という名前の要素に関し、そのデータ型は「customerListType」であるとして定義されている。「customerListType」というデータ型は、次行において、「sfa」という名前空間にて「listID」、「to

talEstimate」、「totalNumber」、「customer」という4つの子要素を含むとして定義されている。更に、「customer」要素のデータ型は「customerType」であり、その内容も定義されている。また、「customer」要素の個数は0個以上として定義されている。

ソースファイルは、スキーマファイルに示される規則にしたがって記述されなければならない。スキーマファイルはソースファイルに含まれる各要素のデータ型や構造を規定するファイルであるから、ソースファイルそのものよりも要素間の構造に関するルールを理解しやすい。

[0189] 図32は、図31のスキーマファイルに対応するソースファイルを示す図である。

このソースファイルにおいては、「customerList」の子要素として、「listID」、「totalNumber」、「totalEstimate」および「cusutomer」が定義されている。また、これらの要素には値が含まれている。「cusutomer」要素は、3つ含まれている。

[0190] 図33は、図32のスキーマファイルと図31のソースファイルに基づいて生成される定義ファイルを示す図である。

この定義ファイルの一部を示している。ここに示す定義ファイルにおいては、「sfa:customerList/sfa:listID」や「sfa:customerList/sfa:totalNumber」といったソースファイルにおける各要素をXHTML形式のデスティネーションファイルに変換するためのルールが記述されている。本実施例におけるデータ処理装置は、直感的なユーザインタフェースにてこの定義ファイルを簡易に生成できる。

[0191] 図34は、バインディングファイルの編集画面を示す図である。

データ処理装置は、スキーマファイルから生成したバインディングファイルを同図に示す所定形式にて画像表示させる。同図下部の領域(以下、「プロパティ領域」)には、スキーマファイルに示されていた各要素がツリー表示されるとともに、そのデータ型なども編集可能に表示される。プロパティ領域において、要素名に付されたチェックボックスにチェックを入れることによってその子要素を展開表示させることができる。このような態様によれば、スキーマファイルに膨大な数の要素が定義されているときであっても、編集対象となる要素だけに表示対象を絞ることができる。

[0192] データ処理装置は、各要素に対して一意にIDを設定する。たとえば、「listID」という要素には「L1」というIDが設定されている。IDは、要素名の頭文字の「L」と通し番号

の「1」をあわせることにより決定されている。また、データ処理装置は、各要素に対して一意にサンプル値を設定する。「listID」という要素には「2005-G30182」というサンプル値が設定されている。同図中央上部には、これらの要素の表示形式を定義するための領域(以下、「レイアウト領域」)が設けられている。ユーザはレイアウト領域において、各要素のIDを配列することにより、レイアウトファイルに反映させることができる。レイアウト領域とレイアウトファイルの関係については図43以降に関連して詳述する。なお、図33のプロパティ領域中段において、要素「customer」をテーブル形式で表示させるように表示形式が指定されている。この指定が、次の図35に示すレイアウトファイルに反映される。

[0193] 図35は、図34におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。

図33における指定にしたがって、要素「customer」の子要素はテーブル形式で表示されている。たとえば、スキーマファイルの要素「customerList/customer/name」は、レイアウトファイルに示されるテーブルのもっとも左の表示領域に対応している。また、このときの表示形式は、図34のレイアウト領域における設定が反映される。ユーザは、この編集画面においてレイアウトファイルをいわゆるWYSIWYG(What You See Is What You Get)にて編集できる。

[0194] こうして、スキーマファイルに示される各要素の表示レイアウトがレイアウトファイルとして保存される。データ処理装置は、このようなスキーマファイルの要素とレイアウトファイルの要素の対応関係からXSLTファイルを生成し、更に、前提技術で説明した定義ファイルを生成する。

[0195] ユーザは、レイアウトファイルの編集画面において、要素の表示位置をドラッグアンドドロップによって変更できる。すでに定義ファイルが生成された後の編集であれば、データ処理装置は、スキーマファイルの要素とレイアウトファイルの表示位置の対応関係の変化を定義ファイルに反映させなければならない。データ処理装置は、レイアウトファイルにおけるサンプル値とスキーマファイルの要素との対応関係を監視している。このため、レイアウトファイルにおける要素の位置が変更されても、サンプル値の位置に応じて定義ファイルを更新できる。ここでレイアウトファイルに含まれる各表示

要素は、サンプル値によって特定される。そのため、XSLTファイルを生成するときには、レイアウトファイルにおけるサンプル値をキーとして、対応関係が再定義される。

[0196] 図36は、図35における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

ソースファイルから定義ファイルにしたがってデスティネーションファイルが生成される。図35は、このデスティネーションファイルを表示させた画面である。ソースファイルの要素「customer」は3つあったので、図35のテーブル形式にしたがって3つの「customer」要素が表示されている。ユーザは、図36の画面を介してソースファイルのデータを編集できる。この仕組みは、前提技術でボキャブラリコネクションとして説明した仕組みである。

[0197] 図37は、バインディングファイルの編集画面の別例を示す図である。

同図のプロパティ領域中段においては、要素「customer」をリスト形式で表示させるように指定されている点が図34と異なる。

[0198] 図38は、図37におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。

図37における表示形式の指定にしたがって、要素「customer」の子要素はリスト形式で表示されている。このように、バインディングファイルにおける表示形式の指定にしたがって、レイアウトファイルも変化する。

同図の場合、スキーマファイルの要素「customerList/customer/name」は、レイアウトファイルに示される各リストの先頭要素に対応している。データ処理装置は、このようなスキーマファイルの要素とレイアウトファイルの要素の対応関係から前提技術で説明した定義ファイルを生成する。

[0199] 図39は、図38における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

ソースファイルの要素「customer」は3つあったので図37のリスト形式にしたがって3つの要素「customer」の内容がリスト表示されている。ユーザは、図39の画面を介してソースファイルを編集できる。

[0200] 図40は、バインディングファイルの編集画面の更に別例を示す図である。

同図においては、要素「totalNumber」の値を算出するための計算式として「count(N1)」がユーザからの編集操作により設定されている。「N1」はすなわち要素「name」のIDであるから、要素「totalNumber」の値は、ソースファイルにおける要素「name」の数である。また、要素「totalEstimate」の値を算出するための計算式として「sum(E1)」が設定されている。「E1」はすなわち要素「estimate」のIDであるから、要素「totalEstimate」の値は、ソースファイルにおける要素「estimate」の値の合計値である。このように、バインディングファイルの編集画面においては、長い要素名ではなくID値によって各要素をシンプルな入力形式にて取り扱うことができる。

[0201] 図41は、図40におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。図41は図35とかわるところはない。

[0202] 図42は、図41における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

「totalNumber」という項目には「3」、すなわち、ソースファイルにおける要素「name」の数が表示されている。同様に、「totalEstimate」という項目には「8000」、すなわち、ソースファイルにおける要素「estimate」の値を合計値(1000+3500+3500=8000)が表示されている。

[0203] 図43は、バインディングファイルの編集画面の更に別例を示す図である。

ここでは、図31に示したスキーマファイルとは別のスキーマファイルを例にとって説明する。

同図に示すように、ユーザは、レイアウト領域においてIDを配列することにより、レイアウトファイルにおける基本的なレイアウトを設定できる。バインディングファイルが表示されるとき、レイアウト領域には、各要素のIDが初期設定として配列される。このときの配列は、スキーマファイルにおける要素構造を反映した配列であってもよい。たとえば、親子、あるいは、兄弟の関係にある要素同士は、表示位置が近くなるように配列されてもよい。また、「totalNumber」、「Number」、「subtotalNumber」のように、要素名が近いときには、これらの要素同士は、表示位置が近くなるように配列されてもよい。このようにIDの配列を初期設定する、最初から配列を作成するよりも、レイアウト作成の省力化を図ることができる。

[0204] 図44は、図43におけるバインディングファイルの編集結果に基づくレイアウトファイル編集画面を示す図である。

レイアウトファイルにおいては、図43のレイアウト領域の編集内容にしたがって各要素が表示されている。

[0205] 図45は、図44における編集結果に基づいたデスティネーションファイルを表示させたときの画面図である。

[0206] 図46も、定義ファイル生成過程を更に説明するための模式図である。

同図について付言すると、バインディングファイルに対するユーザの編集操作によって、バインディングファイルに補完的な情報が付加される。たとえば、要素に関する規則の定義や再定義などである。このバインディングファイルをもとにして、定義ファイルが生成されるが、定義ファイルの代わりに、XSLTファイルが生成されてもよい。そのほかにも、ソースファイルとデスティネーションファイル間のデータマッピングを実現するためのオブジェクト、たとえば、Java(登録商標)のオブジェクトが生成されてもよい。

[0207] 本実施例においては、XML文書ファイルをソースファイルとして取得するとして説明した。このXML文書ファイルは、外部のデータベースから取得されてもよい。

[0208] 図47は、本実施例におけるデータ処理装置とXMLデータベースとの連携方法を説明するための模式図である。

まず、データ処理装置から外部のXMLデータベース(以下、「XML－DB」とよぶ)に対して、Xクエリ(XQuery)に基づくデータ要求(以下、単に「クエリ」とよぶ)が送信される。Xクエリとは、XMLベースの文書ファイルに対してさまざまな問い合わせを行うために開発された言語である。このXML－DBは、Xクエリに対応可能なデータベースである。XML－DBは、データ処理装置からのクエリに応じて、求められたデータをXML文書ファイル(以下、「リザルトファイル」とよぶ)として返信する。このリザルトファイルの要素とその構造は、もともとのクエリによって規定される。クエリによって求められたデータであって、かつ、XML－DBに存在するデータがリザルトファイルの要素となる。データ処理装置は、XML－DBから送信されたリザルトファイルの要素とその構造を抽出し、図30等に関連して説明した定義ファイル生成処理を実行しても

よい。あるいは、クエリによって規定される要素構造に応じて編集画面を生成するためのスキーマを特定し、リザルトファイルに該当要素が含まれていないときには、そのスキーマから不要項目を削除してもよい。

[0209] 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。

産業上の利用可能性

[0210] 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。

請求の範囲

[1] 所定のタグセットで記述された構造化文書ファイルについて、その要素構造を示すスキーマ情報を取得するスキーマ情報取得部と、

前記構造化文書ファイルの編集用ユーザインタフェース画面を表示するための定義データを前記スキーマ情報に基づいて生成する定義データ生成部と、

外部のデータベースに対して、データを要求するためのクエリ(Query)を送信するデータ要求部と、を備え、

前記スキーマ情報取得部は、前記データベースから前記クエリによって取得対象として指定されたデータを構造化文書ファイルとして取得し、その構造化文書ファイルを参照して要素構造を特定することによって前記スキーマ情報を取得することを特徴とするデータ処理装置。

[2] 前記スキーマ情報取得部は、更に、前記構造化文書ファイルの要素構造に関する規則が定義されたスキーマファイルにより前記スキーマ情報を取得することを特徴とする請求項1に記載のデータ処理装置。

[3] 前記スキーマ情報取得部は、更に、前記構造化文書ファイルを参照してその要素構造を特定することにより前記スキーマ情報を取得することを特徴とする請求項1に記載のデータ処理装置。

[4] 前記スキーマ情報に示される各要素の表示上のレイアウトを示すレイアウトファイルを生成するレイアウトファイル生成部を更に備え、

前記定義データ生成部は、前記レイアウトファイルに基づいて、前記構造化文書ファイルを編集用のデータ形式に変換するための定義ファイルを前記定義データとして生成することを特徴とする請求項1から3のいずれかに記載のデータ処理装置。

[5] 前記スキーマ情報に示される要素の表示上のレイアウトを編集するための編集画面を表示させる編集処理部を更に備え、

前記レイアウトファイル生成部は、前記編集画面に対するユーザの編集操作に応じて前記レイアウトファイルを生成することを特徴とする請求項4に記載のデータ処理装置。

[6] 前記編集処理部は、更に、前記レイアウトファイルを画面表示させ、前記レイアウト

ファイルに対するユーザの編集操作に応じて前記レイアウトファイルを更新することを特徴とする請求項5に記載のデータ処理装置。

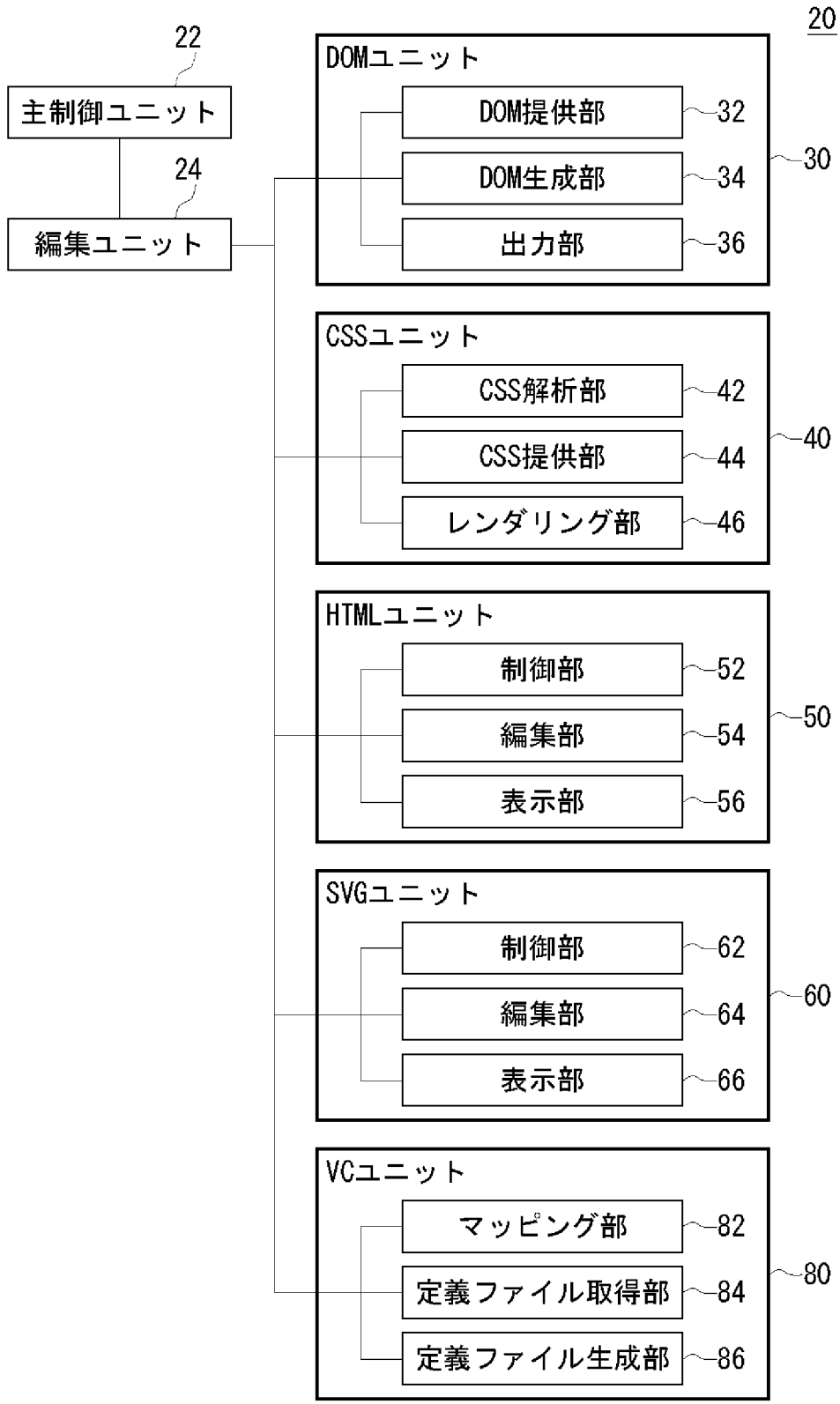
[7] 前記編集処理部は、前記編集画面において、前記スキーマ情報に含まれる各要素の表示上のレイアウトを初期設定して表示させることを特徴とする請求項5または6に記載のデータ処理装置。

[8] 前記編集処理部は、前記スキーマ情報に示される要素構造に応じて、各要素の表示上のレイアウトを初期設定することを特徴とする請求項7に記載のデータ処理装置。

[9] 前記編集処理部は、前記レイアウトファイルの表示形式を選択するための入力インタフェースを前記編集画面に表示させ、

前記レイアウトファイル生成部は、選択された表示形式に応じてレイアウトファイルを生成することを特徴とする請求項5から8のいずれかに記載のデータ処理装置。

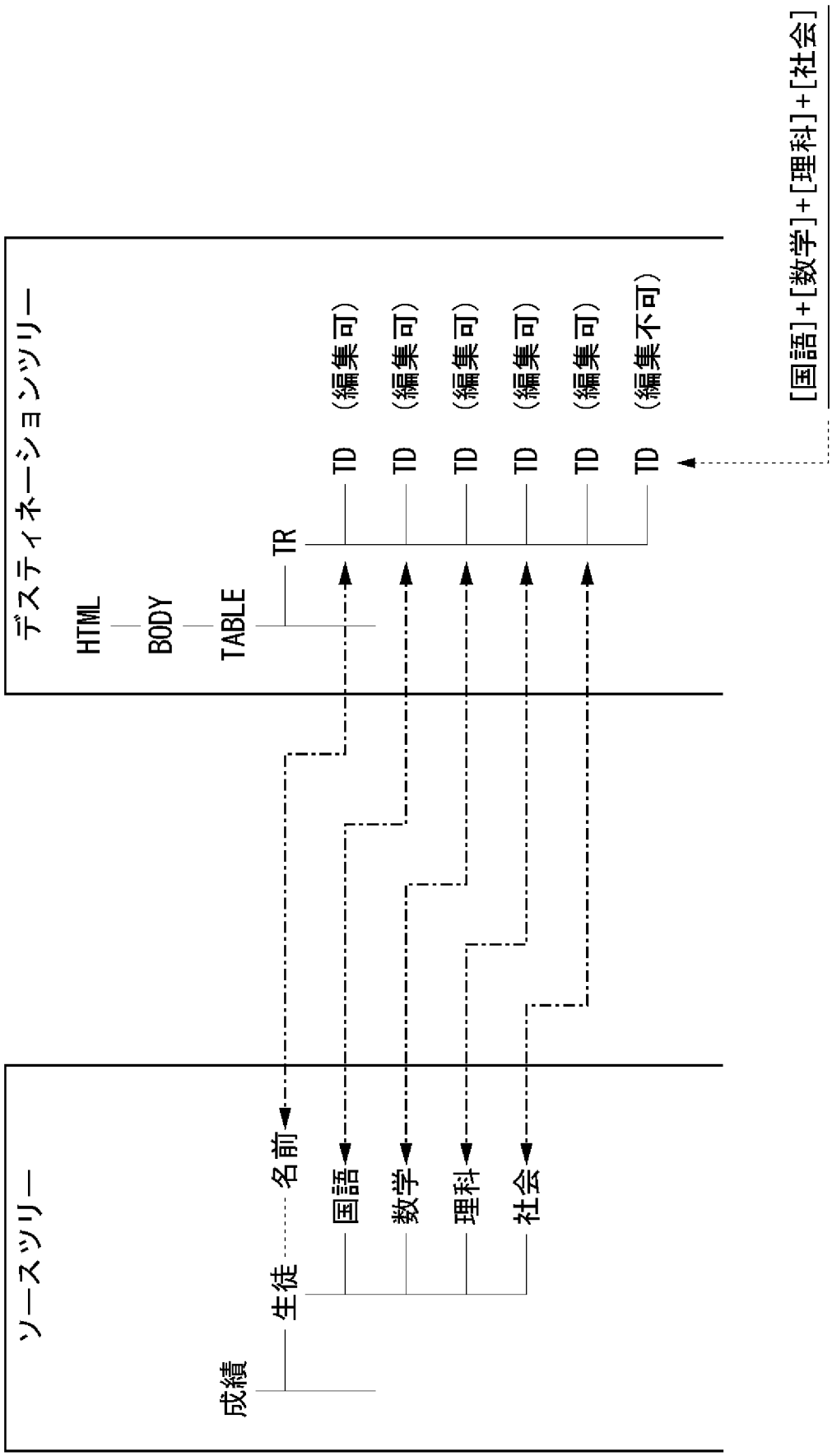
[図1]


20
22
主制御ユニット
24
編集ユニット
DOMユニット
DOM提供部
32
DOM生成部
34
出力部
36
30
CSSユニット
CSS解析部
42
CSS提供部
44
レンダリング部
46
40
HTMLユニット
制御部
52
編集部
54
表示部
56
50
SVGユニット
制御部
62
編集部
64
表示部
66
60
VCユニット
マッピング部
82
定義ファイル取得部
84
定義ファイル生成部
86
80

[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]


ソーススツリー
成績
生徒
名前
国語
数学
理科
社会
デスティネーションツリー
HTML
BODY
TABLE
TR
TD (編集可)
TD (編集可)
TD (編集可)
TD (編集可)
TD (編集可)
TD (編集不可)
[国語]+[数学]+[理科]+[社会]
4

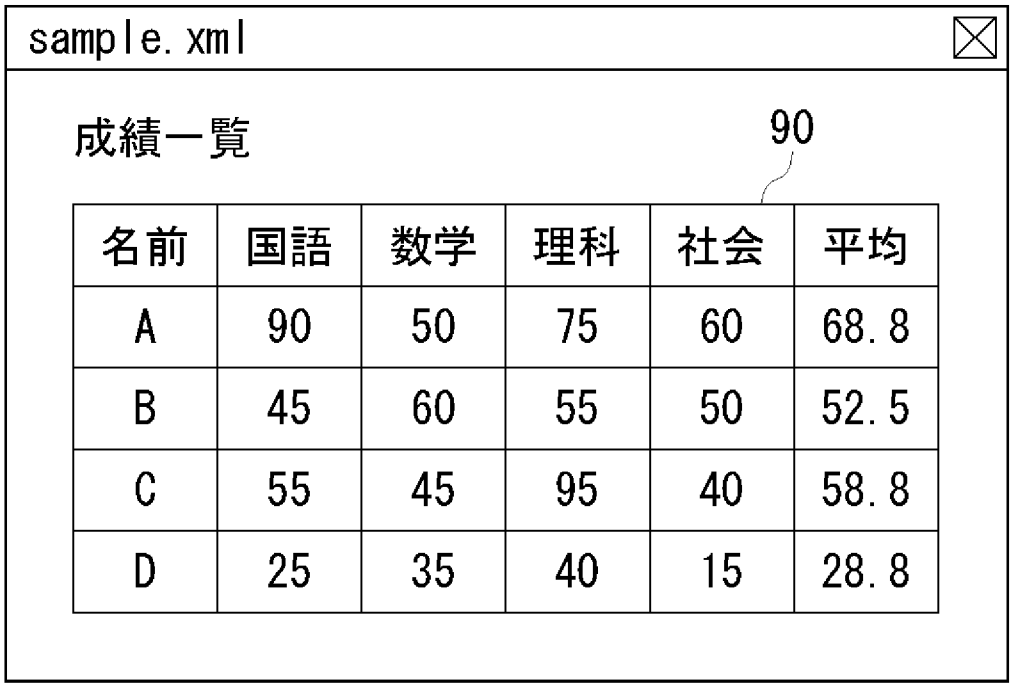
[図4(a)]

```
<?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)]

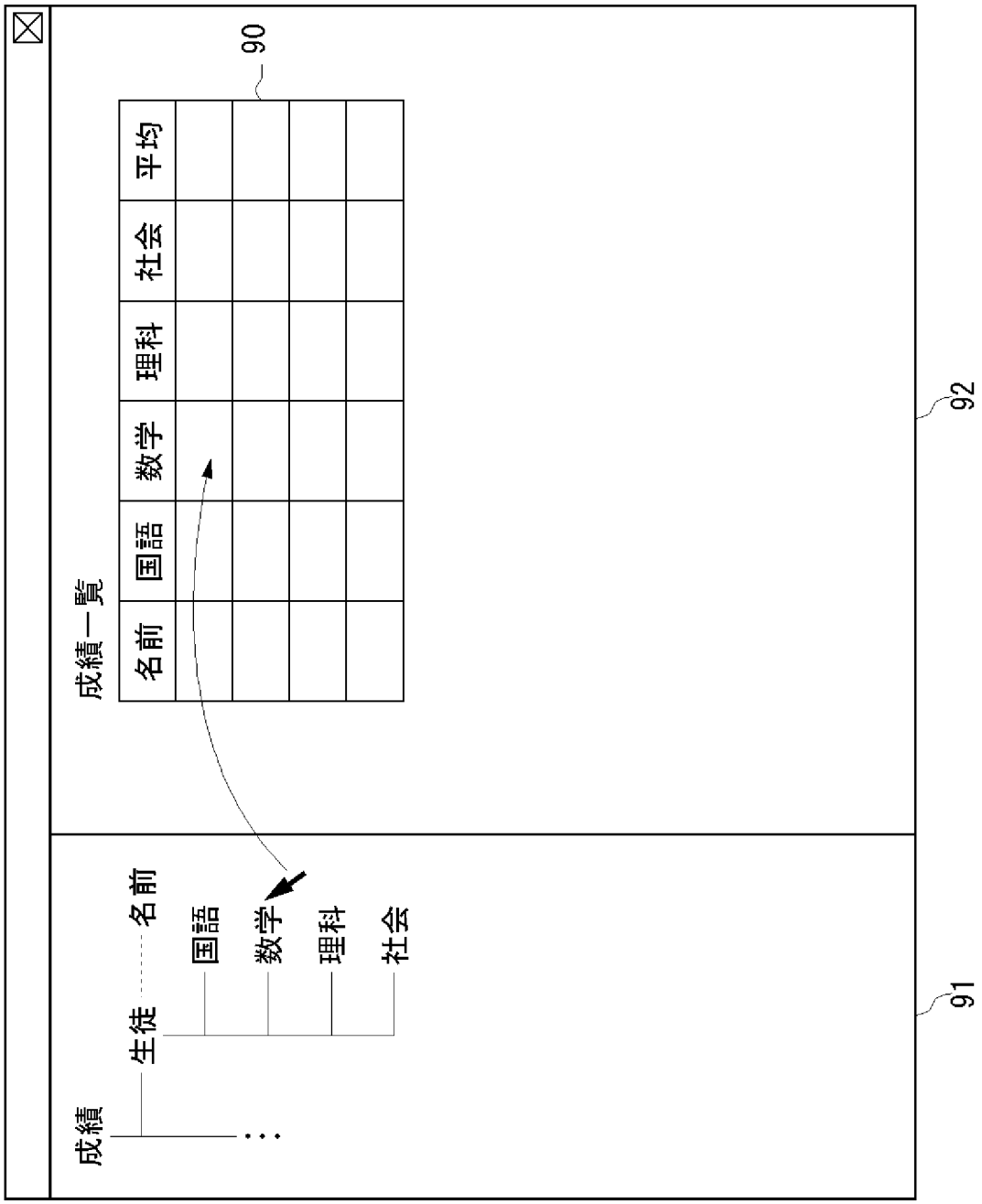
```
        tr{
          border:none;
        }
        .data{
          padding:0.2em 0.5em;
        }
      </style>
    </head>
    <body>
      <h1>成績一覧</h1>
      <table>
        <tr><th><div class="data">名前</div></th>
        <th></th>
        <th><div class="data">国語</div></th>
        <th><div 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]

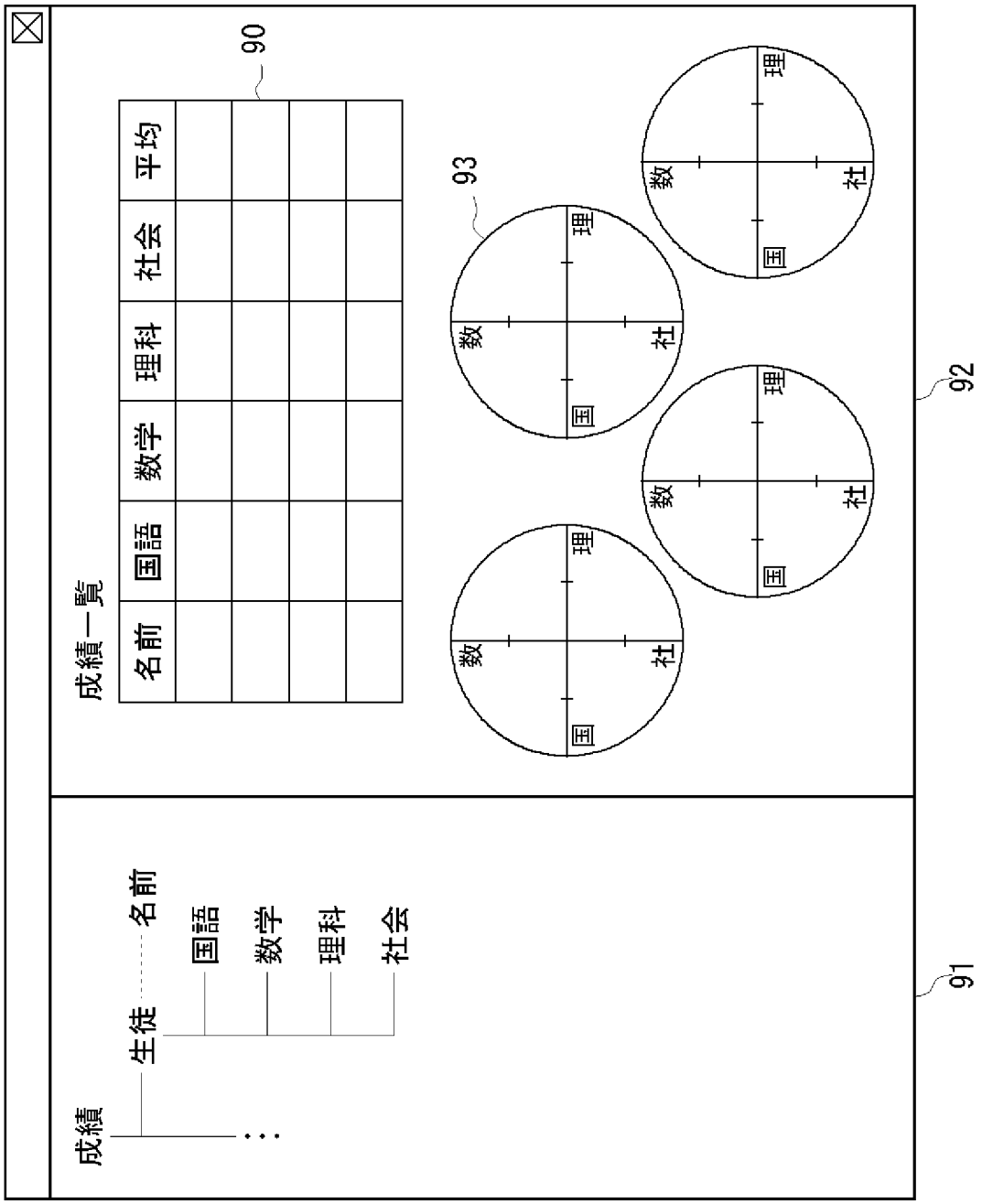


名前	国語	数学	理科	社会	平均
A	90	50	75	60	68.8
B	45	60	55	50	52.5
C	55	45	95	40	58.8
D	25	35	40	15	28.8

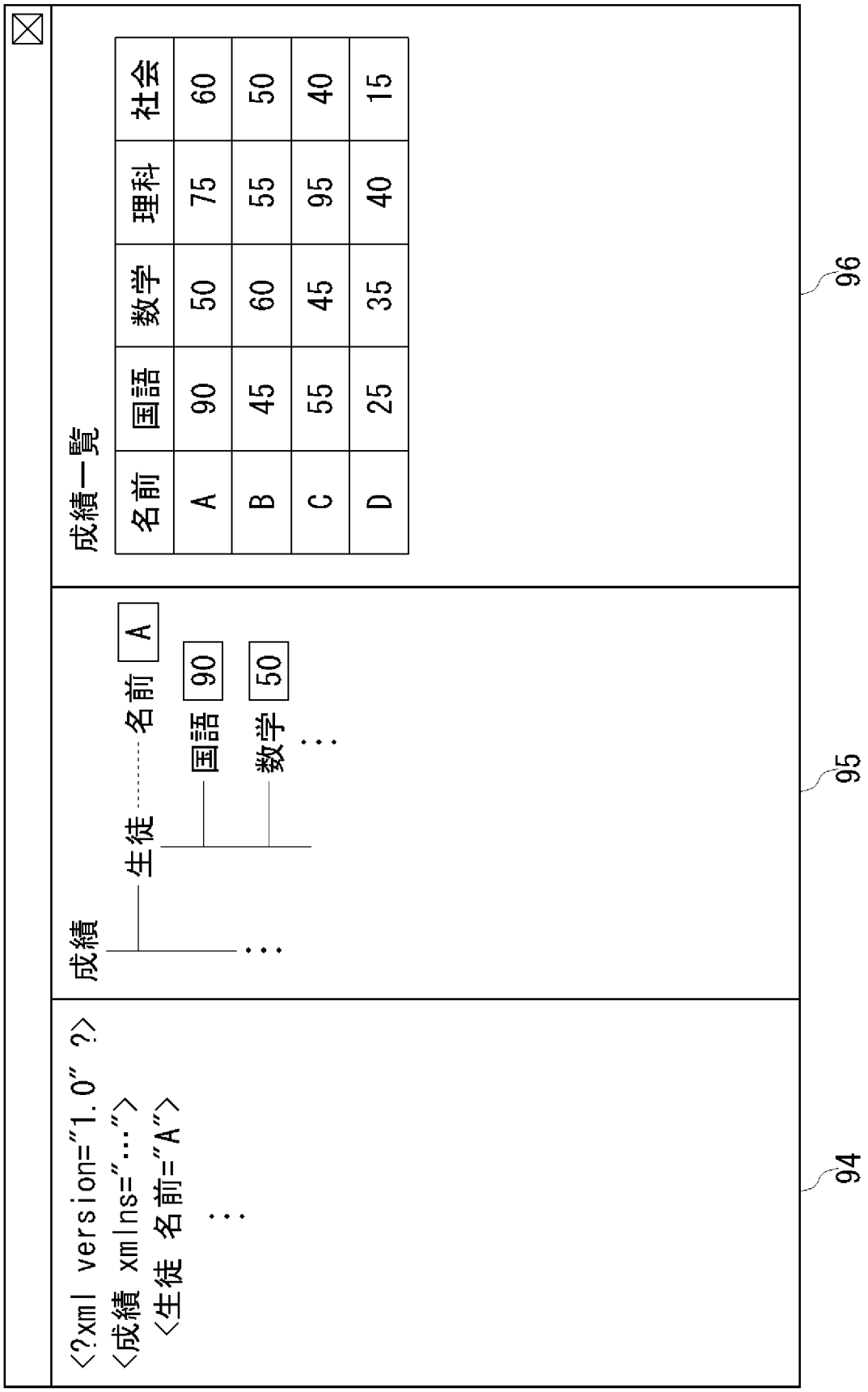
[図6]



[図7]



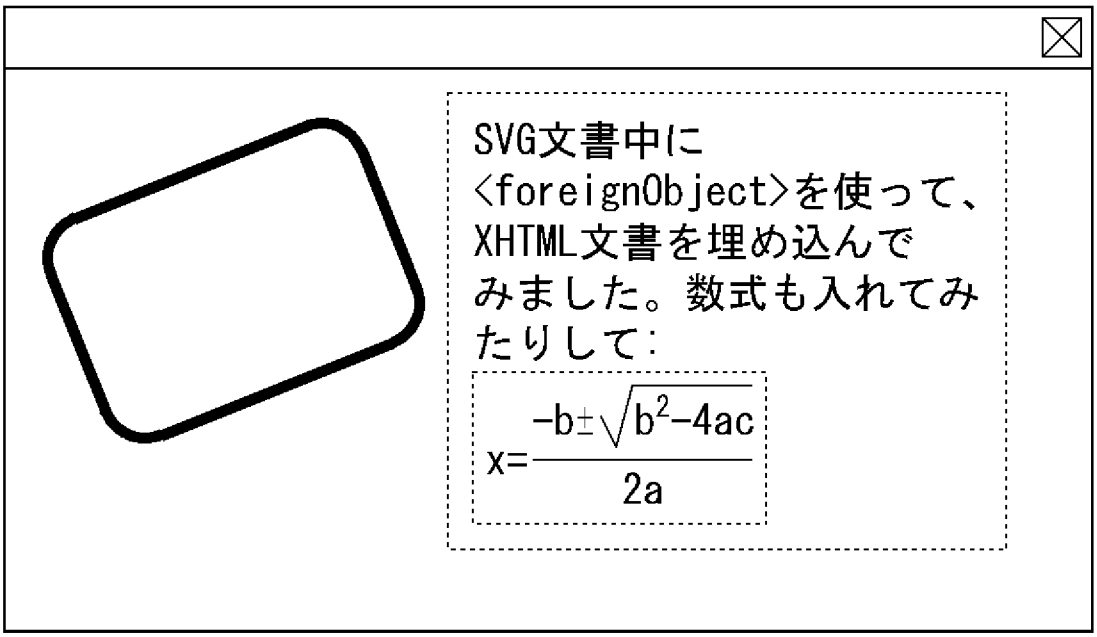
[図8]


<?xml version="1.0" ?>
<成績 xmlns="...">
<生徒 名前="A">
...
94
成績
生徒
名前 A
国語 90
数学 50
...
95
成績一覧
名前 国語 数学 理科 社会
A 90 50 75 60
B 45 60 55 50
C 55 45 95 40
D 25 35 40 15
96

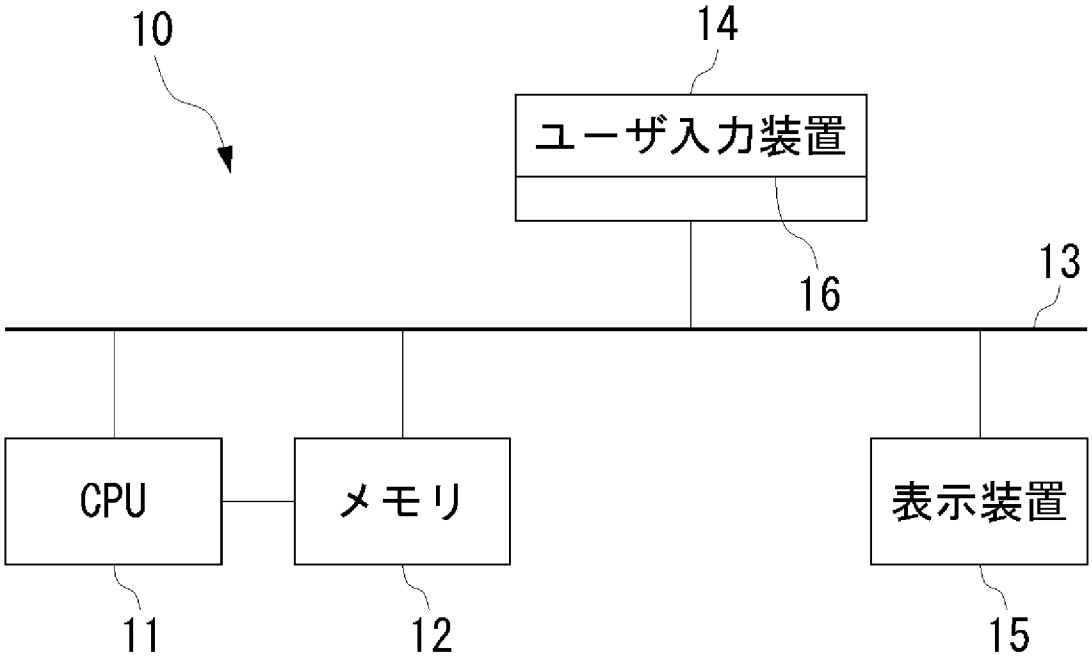
[図9]

```
<?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]


SVG文書中に
<foreignObject>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:
$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$

[図11(a)]


10
14
ユーザ入力装置
16
13
CPU
メモリ
表示装置
11
12
15

[図11(b)]

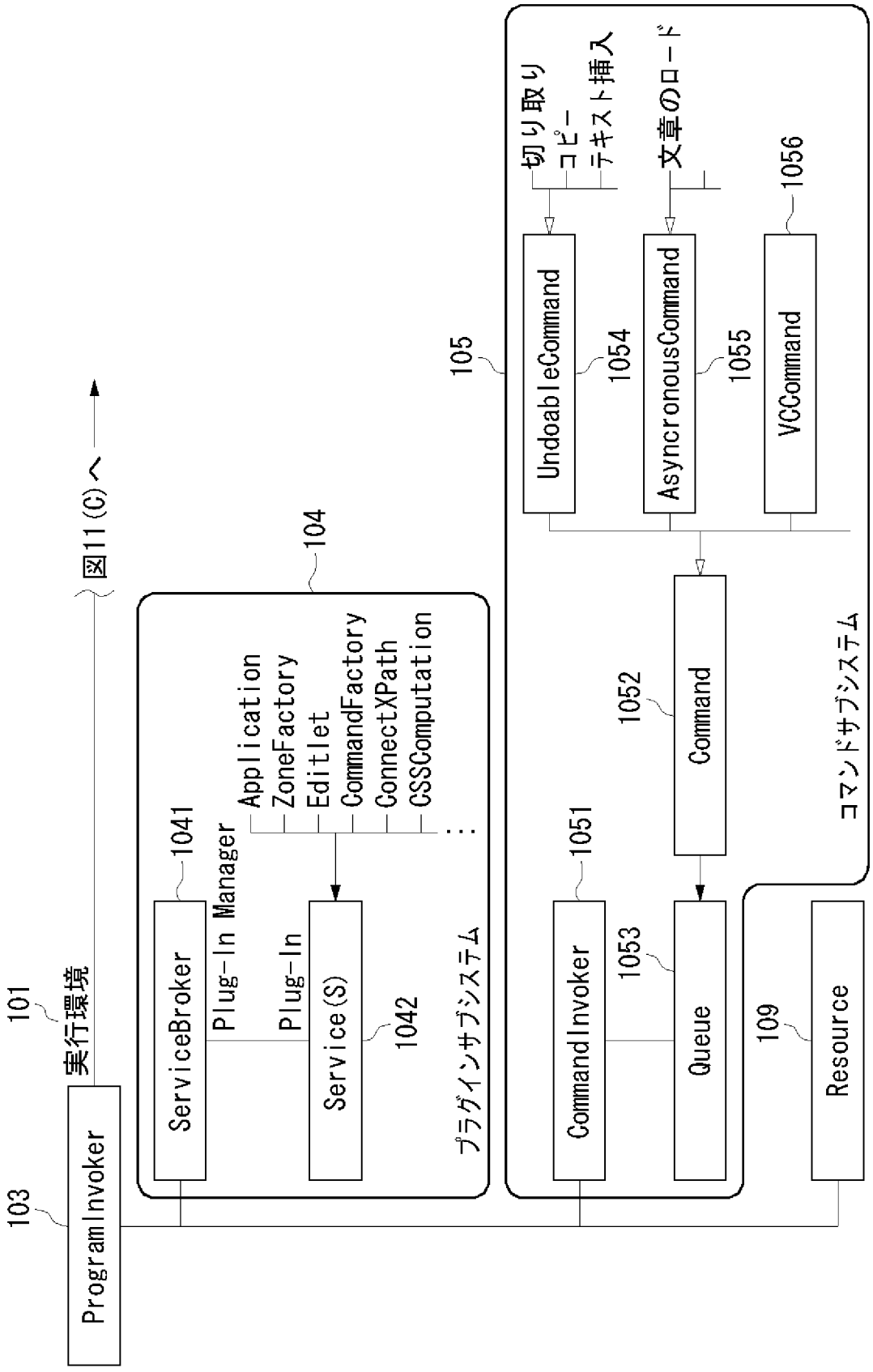


図11(C)へ
101
実行環境
103
ProgramInvoker
104
ServiceBroker
1041
Plug-In Manager
Plug-In
Service(S)
1042
プラグインサブシステム
Application
ZoneFactory
EditIet
CommandFactory
ConnectXPath
CSSComputation
105
UndoableCommand
1054
AsyncronousCommand
1055
VCCommand
1056
切り取り
コピー
テキスト挿入
文章のロード
Command
1052
CommandInvoker
1051
Queue
1053
Resource
109
コマンドサブシステム

[図11(c)]

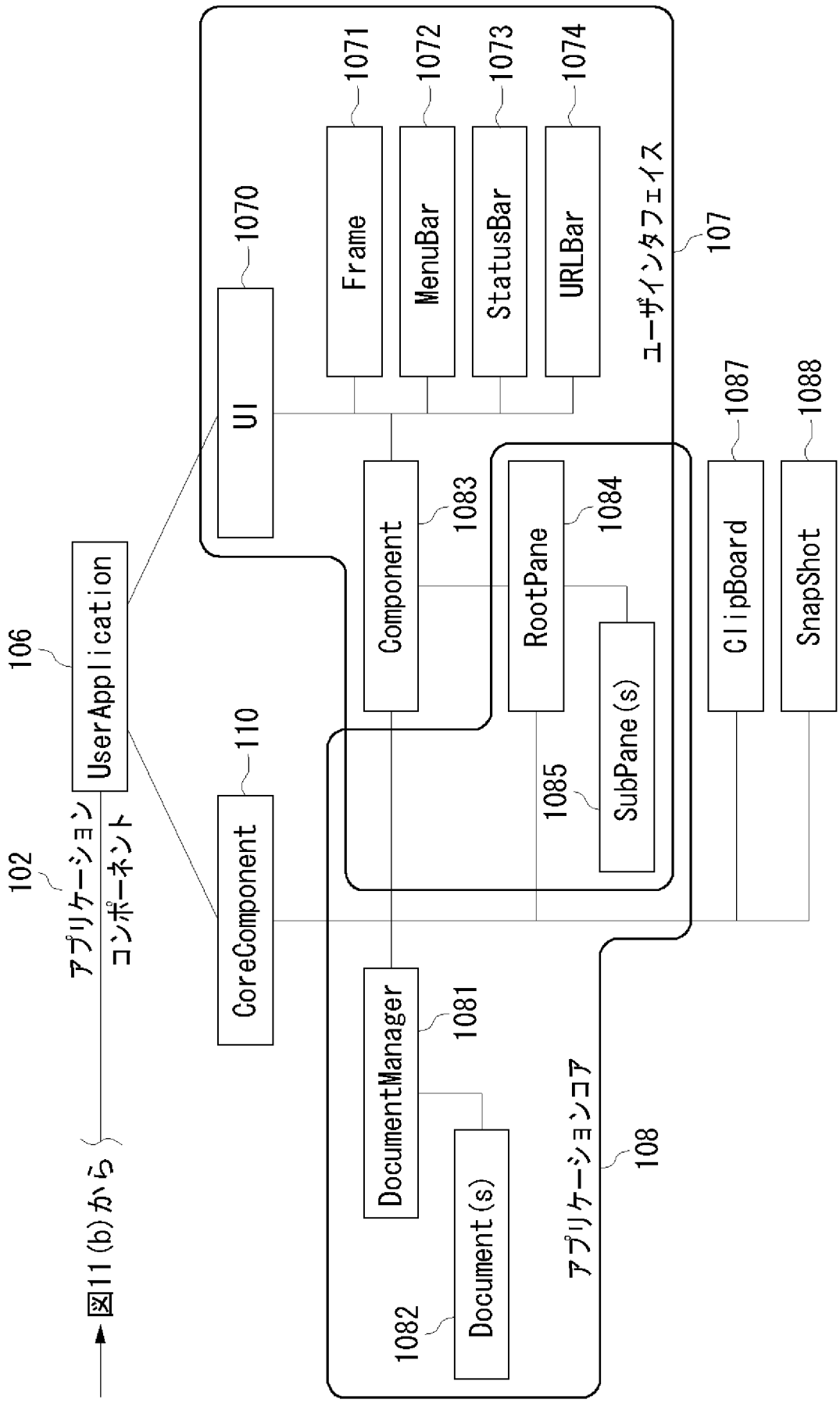
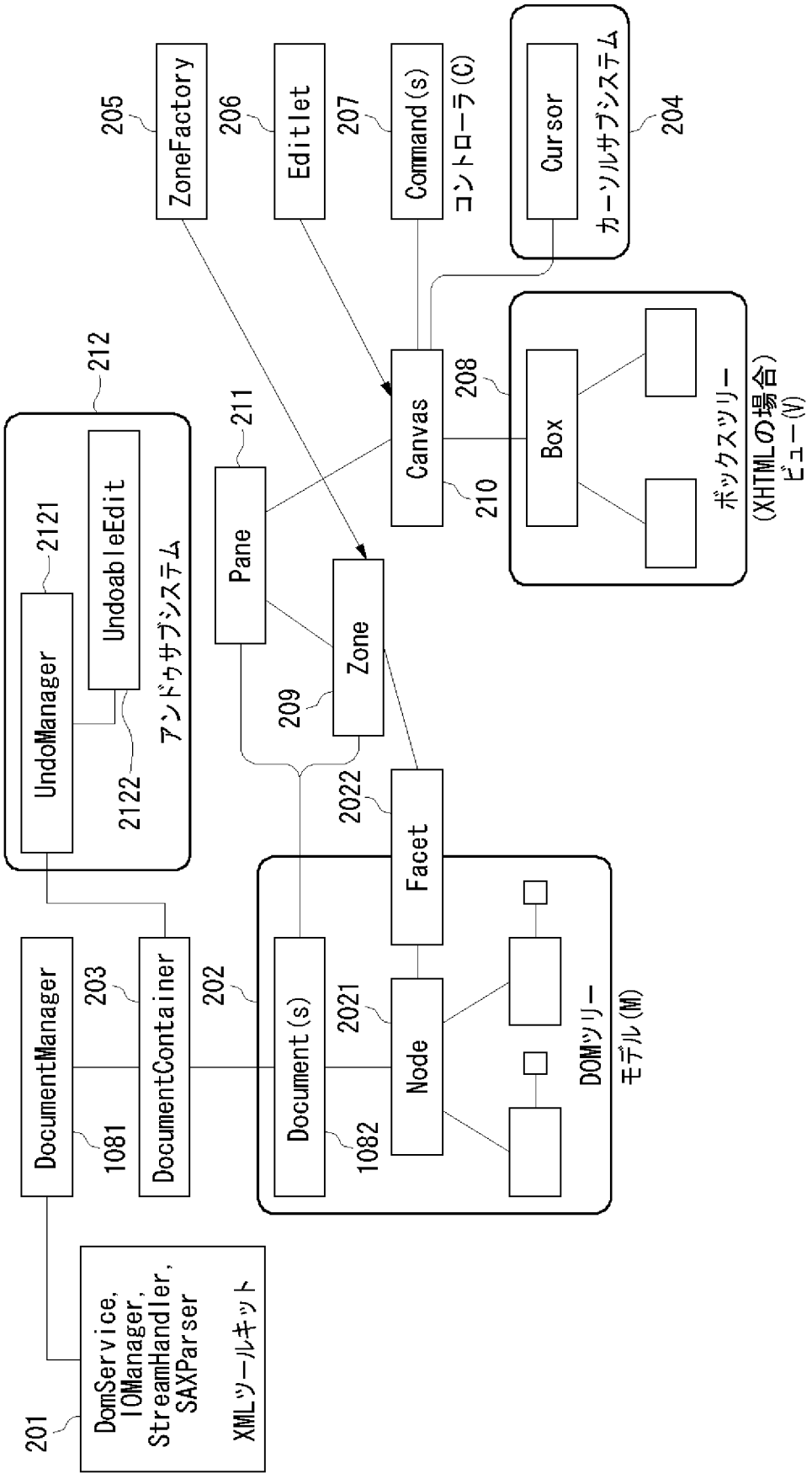
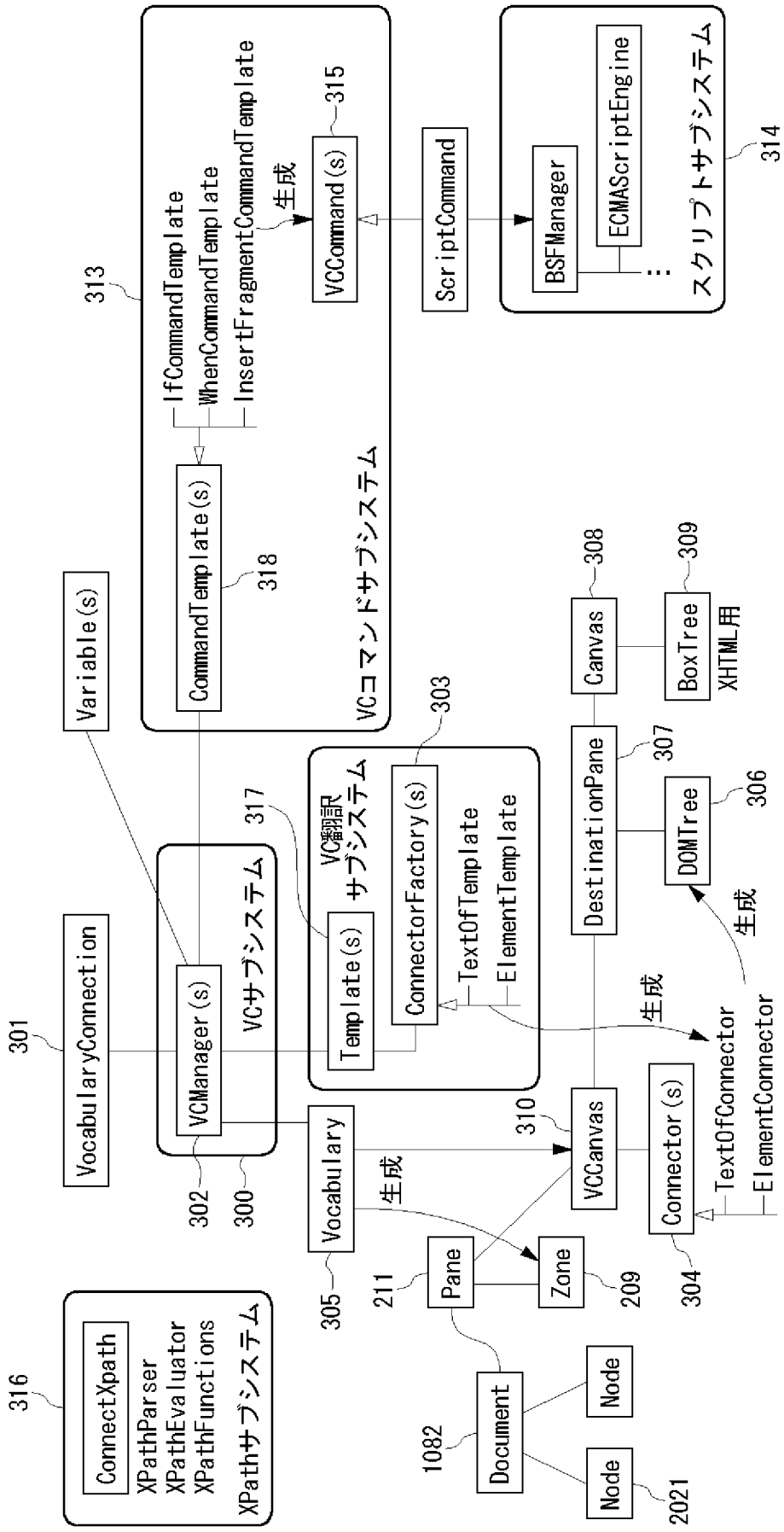


図11(b)から
アプリケーションコンポーネント
102
UserApplication
106
CoreComponent
110
UI
1070
Frame
1071
MenuBar
1072
StatusBar
1073
URLBar
1074
ユーザインタフェース
107
Component
1083
RootPane
1084
SubPane(s)
1085
ClipBoard
1087
SnapShot
1088
DocumentManager
1081
Document(s)
1082
アプリケーションコア
108

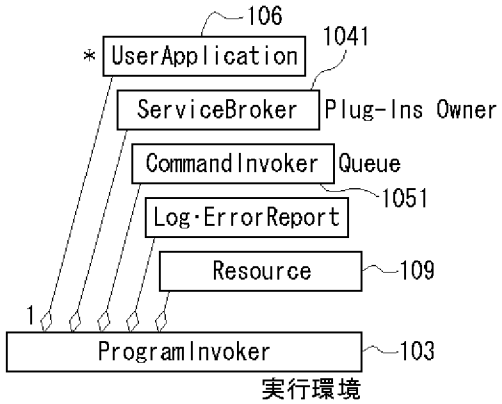
[図12]


201
DomService,
IOManager,
StreamHandler,
SAXParser
XMLツールキット
DocumentManager
1081
DocumentContainer
203
202
Document(s)
1082
Node
2021
Facet
2022
DOMツリー
モデル(M)
Zone
209
Pane
211
Canvas
208
210
Box
ボックスツリー
(XHTMLの場合)
ビュー(V)
UndoManager
2121
UndoableEdit
2122
アンドゥサブシステム
212
ZoneFactory
205
EditLet
206
Command(s)
207
コントローラ(C)
Cursor
204
カーソルサブシステム

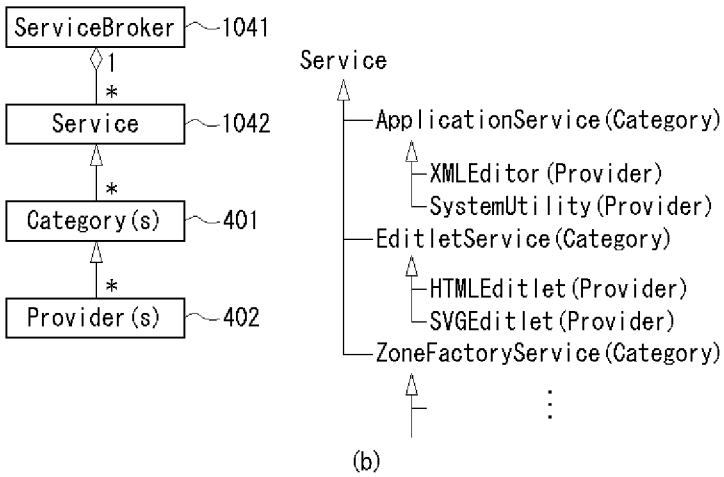
[図13]


313
IfCommandTemplate
WhenCommandTemplate
InsertFragmentCommandTemplate
生成
315
VCCommand(s)
ScriptCommand
BSFManager
ECMAScriptEngine
...
スクリプトサブシステム
314
CommandTemplate(s)
318
VCコマンドサブシステム
Variable(s)
303
VC翻訳
サブシステム
ConnectorFactory(s)
TextOfTemplate
ElementTemplate
Canvas
308
BoxTree
309
XHTML用
DestinationPane
307
DOMTree
306
生成
317
VCサブシステム
Template(s)
VCManager(s)
301
VocabularyConnection
生成
310
VCCanvas
Connector(s)
TextOfConnector
ElementConnector
302
300
Vocabulary
305
211
Pane
Zone
209
304
316
ConnectXpath
XPathParser
XPathEvaluator
XPathFunctions
XPathサブシステム
1082
Document
Node
Node
2021

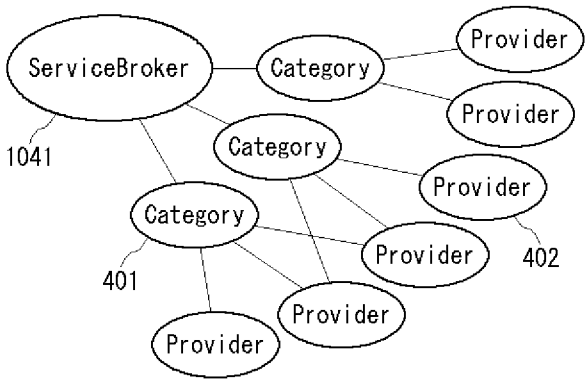
[図14]


106
1041
* UserApplication
ServiceBroker
Plug-Ins Owner
CommandInvoker
Queue
1051
Log・ErrorReport
Resource
109
1
ProgramInvoker
103
実行環境

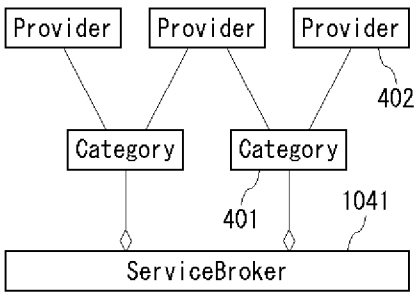
(a)


ServiceBroker
1041
1
*
Service
1042
*
Category(s)
401
*
Provider(s)
402
Service
ApplicationService(Category)
XMLEditor(Provider)
SystemUtility(Provider)
EditletService(Category)
HTMLEditlet(Provider)
SVGEditlet(Provider)
ZoneFactoryService(Category)

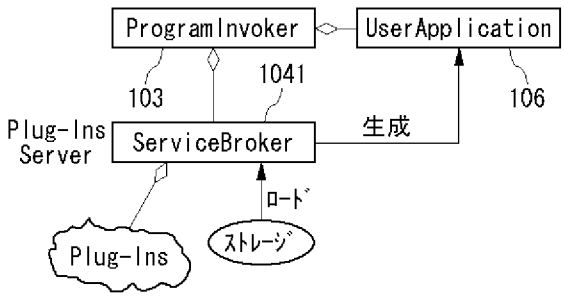
(b)


ServiceBroker
1041
Category
Provider
Provider
Category
Provider
Category
Provider
402
401
Provider
Provider

(c)

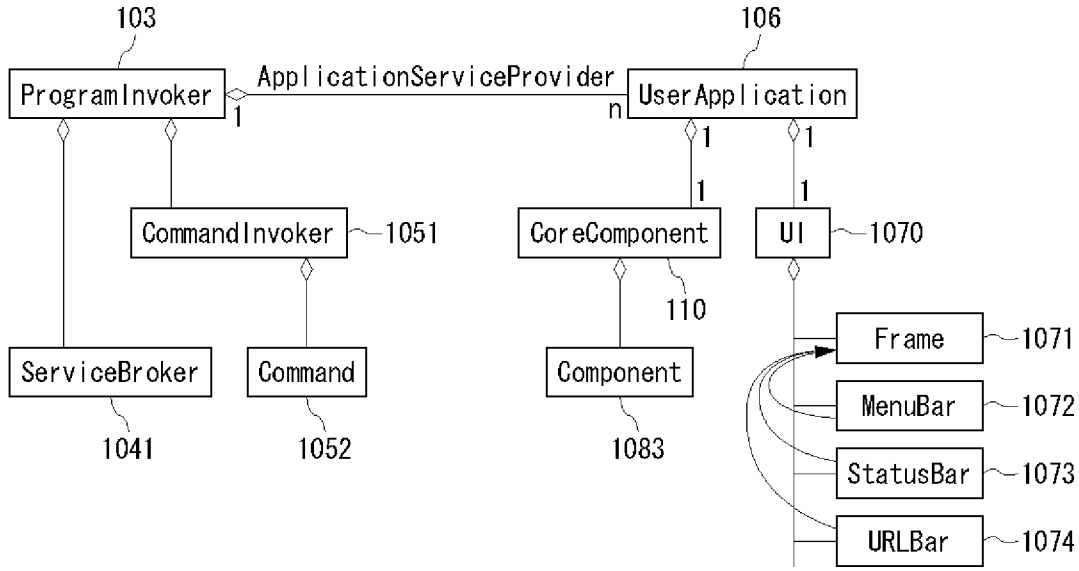

Provider
Provider
Provider
402
Category
Category
401
1041
ServiceBroker

(d)

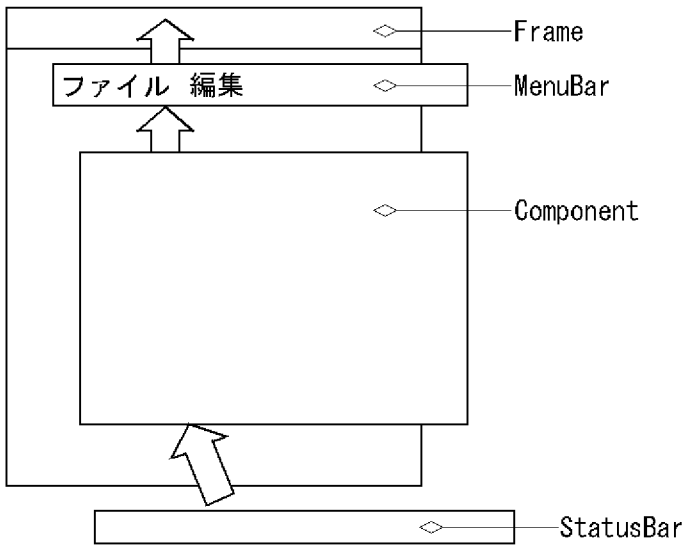

ProgramInvoker
UserApplication
103
1041
106
Plug-Ins
Server
ServiceBroker
生成
ロード
Plug-Ins
ストレージ

(e)

[図15]

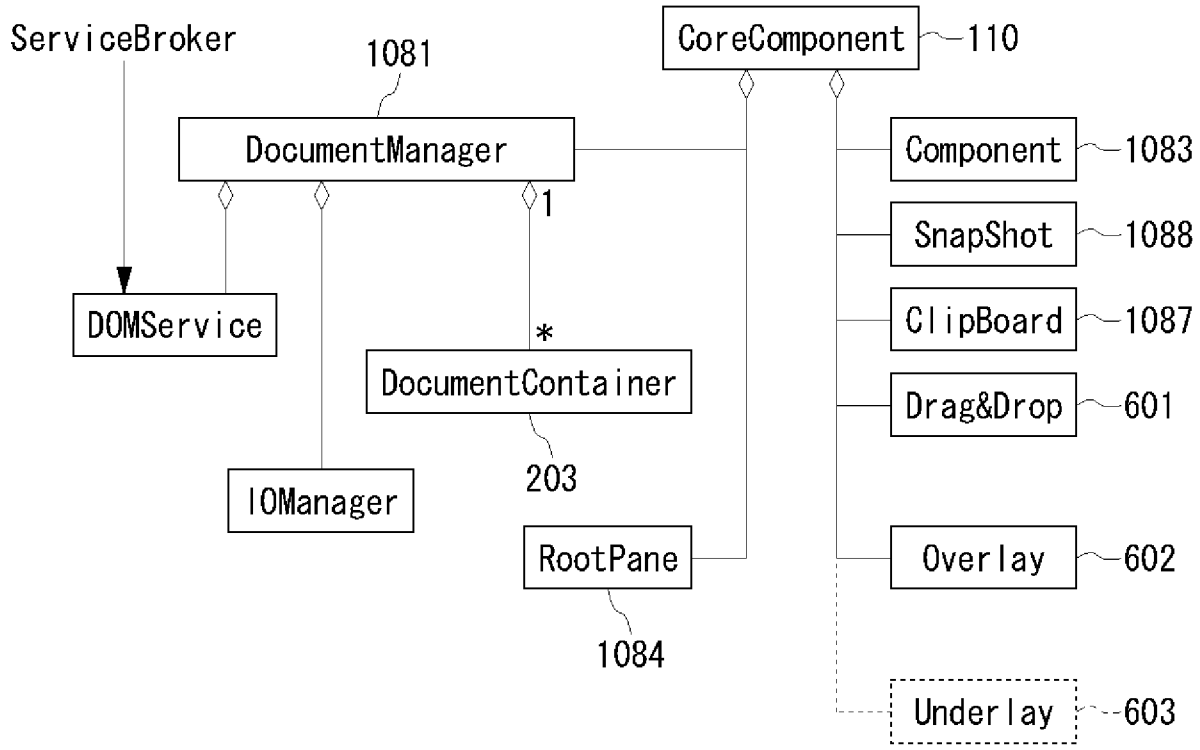

103
106
ProgramInvoker
ApplicationServiceProvider
UserApplication
1
n
1
1
1
1
CommandInvoker
1051
CoreComponent
UI
1070
110
Frame
1071
ServiceBroker
Command
Component
MenuBar
1072
StatusBar
1073
URLBar
1074
1041
1052
1083

(a)

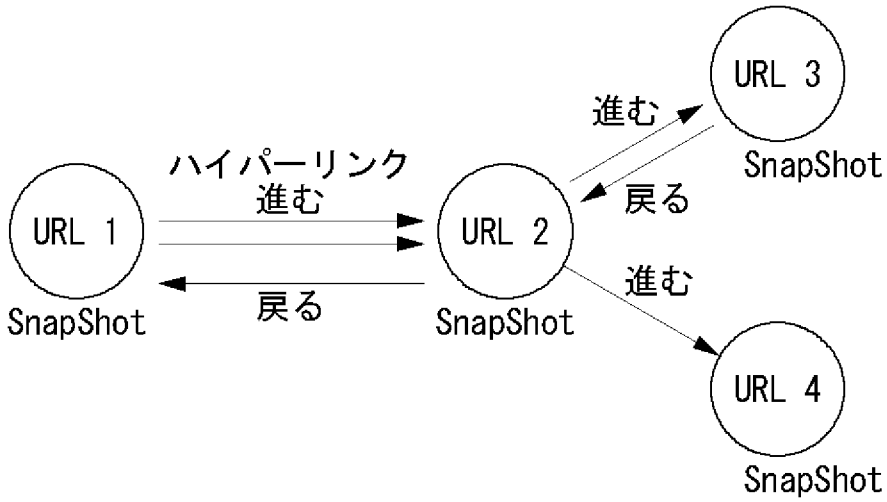

Frame
ファイル 編集
MenuBar
Component
StatusBar

(b)

[図16]

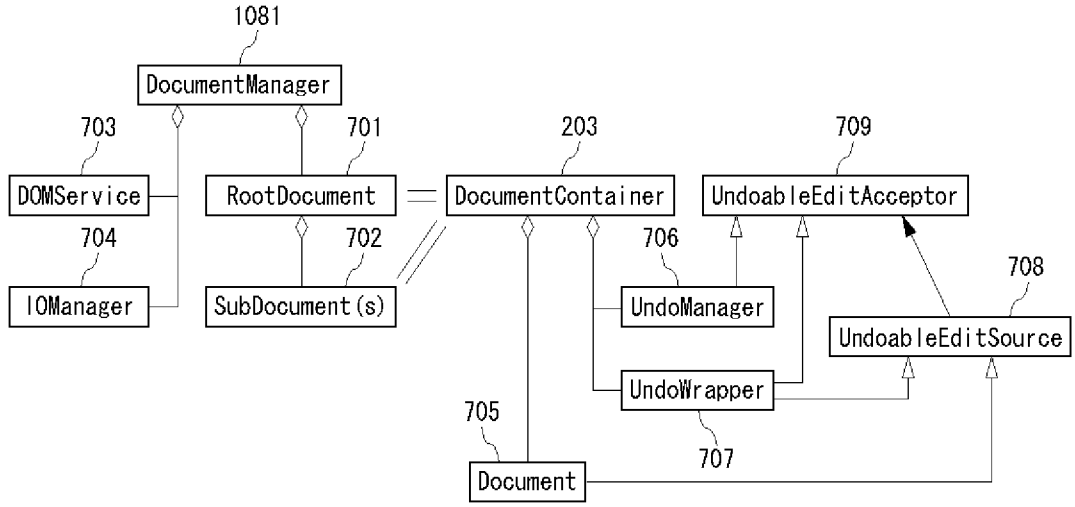

ServiceBroker
1081
DocumentManager
CoreComponent
110
Component
1083
SnapShot
1088
ClipBoard
1087
Drag&Drop
601
Overlay
602
Underlay
603
1
*
DOMService
DocumentContainer
203
IOManager
RootPane
1084

(a)

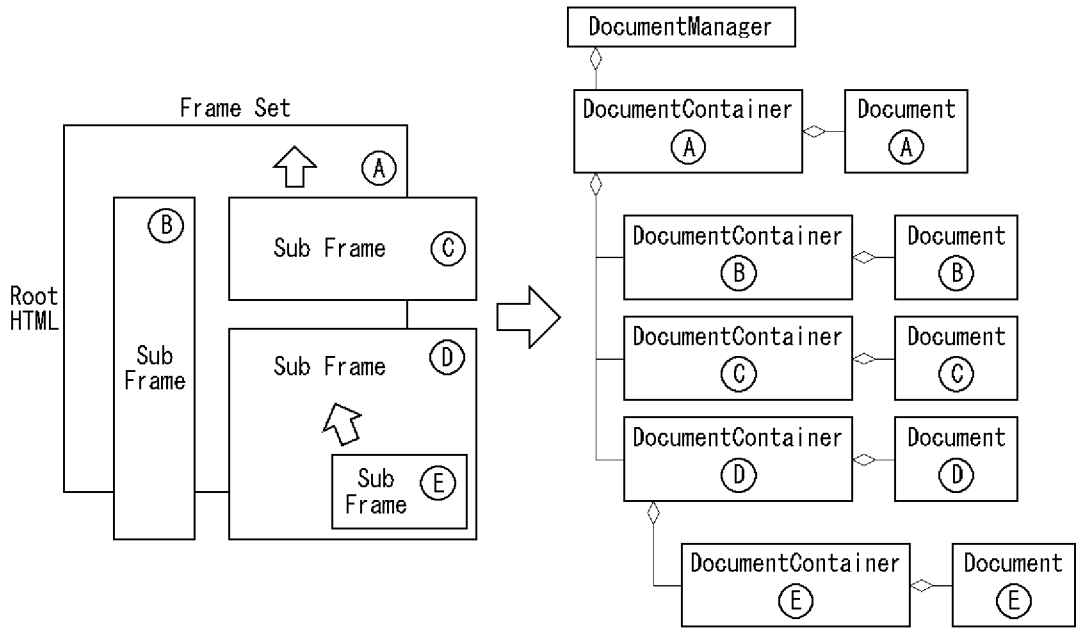

URL 1
SnapShot
ハイパーリンク
進む
戻る
URL 2
SnapShot
進む
戻る
URL 3
SnapShot
進む
URL 4
SnapShot

(b)

[図17]

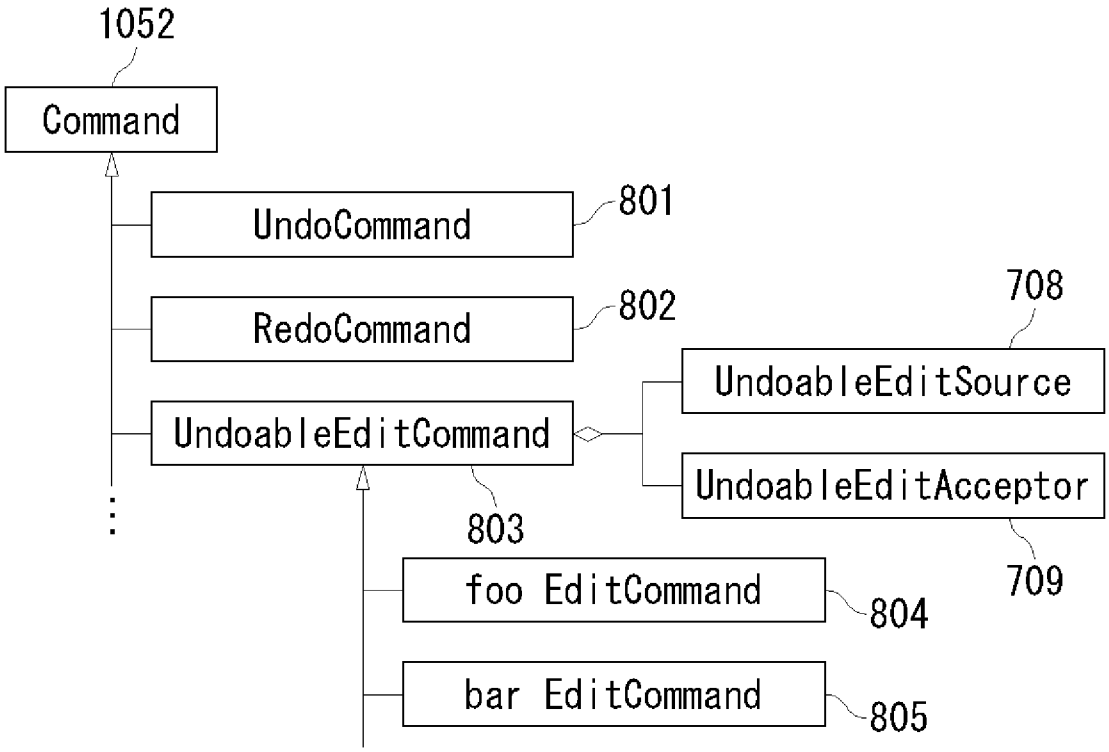

1081
DocumentManager
703
701
203
709
DOMService
RootDocument
DocumentContainer
UndoableEditAcceptor
704
702
706
708
IOManager
SubDocument(s)
UndoManager
UndoableEditSource
UndoWrapper
705
707
Document

(a)

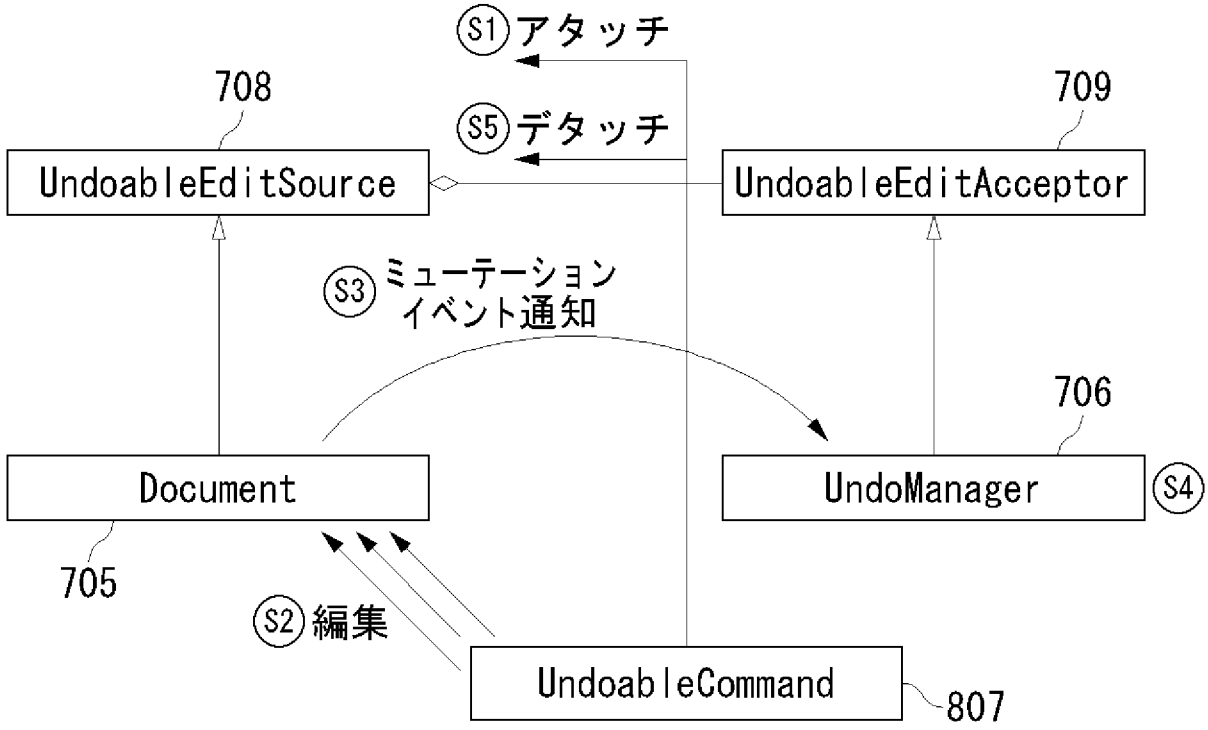

Frame Set
A
B
Sub Frame
C
Root HTML
Sub Frame
Sub Frame
D
Sub Frame
E
DocumentManager
DocumentContainer A
Document A
DocumentContainer B
Document B
DocumentContainer C
Document C
DocumentContainer D
Document D
DocumentContainer E
Document E

(b)

[図18]

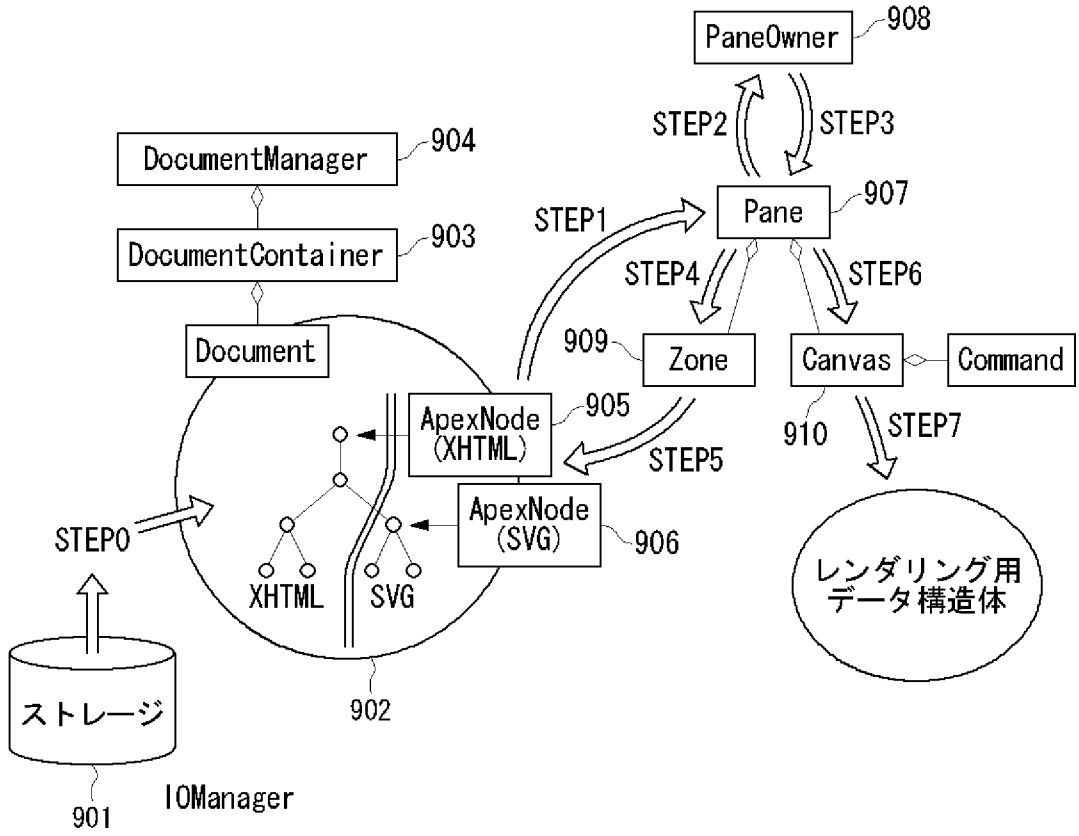

1052
Command
UndoCommand
801
RedoCommand
802
708
UndoableEditSource
UndoableEditCommand
UndoableEditAcceptor
803
709
foo EditCommand
804
bar EditCommand
805

(a)

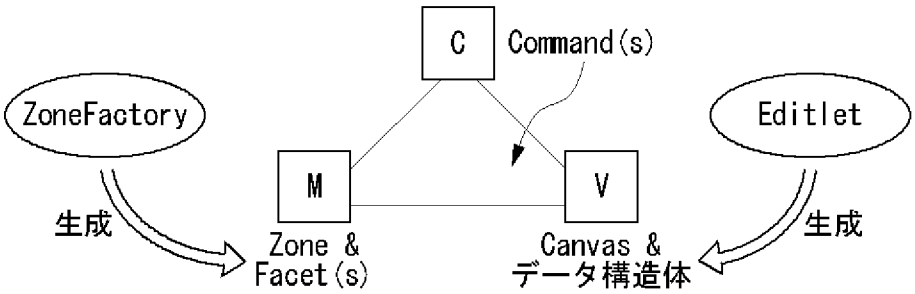

S1 アタッチ
708
709
S5 デタッチ
UndoableEditSource
UndoableEditAcceptor
S3 ミューテーション
イベント通知
706
Document
UndoManager
S4
705
S2 編集
UndoableCommand
807

(b)

[図19]

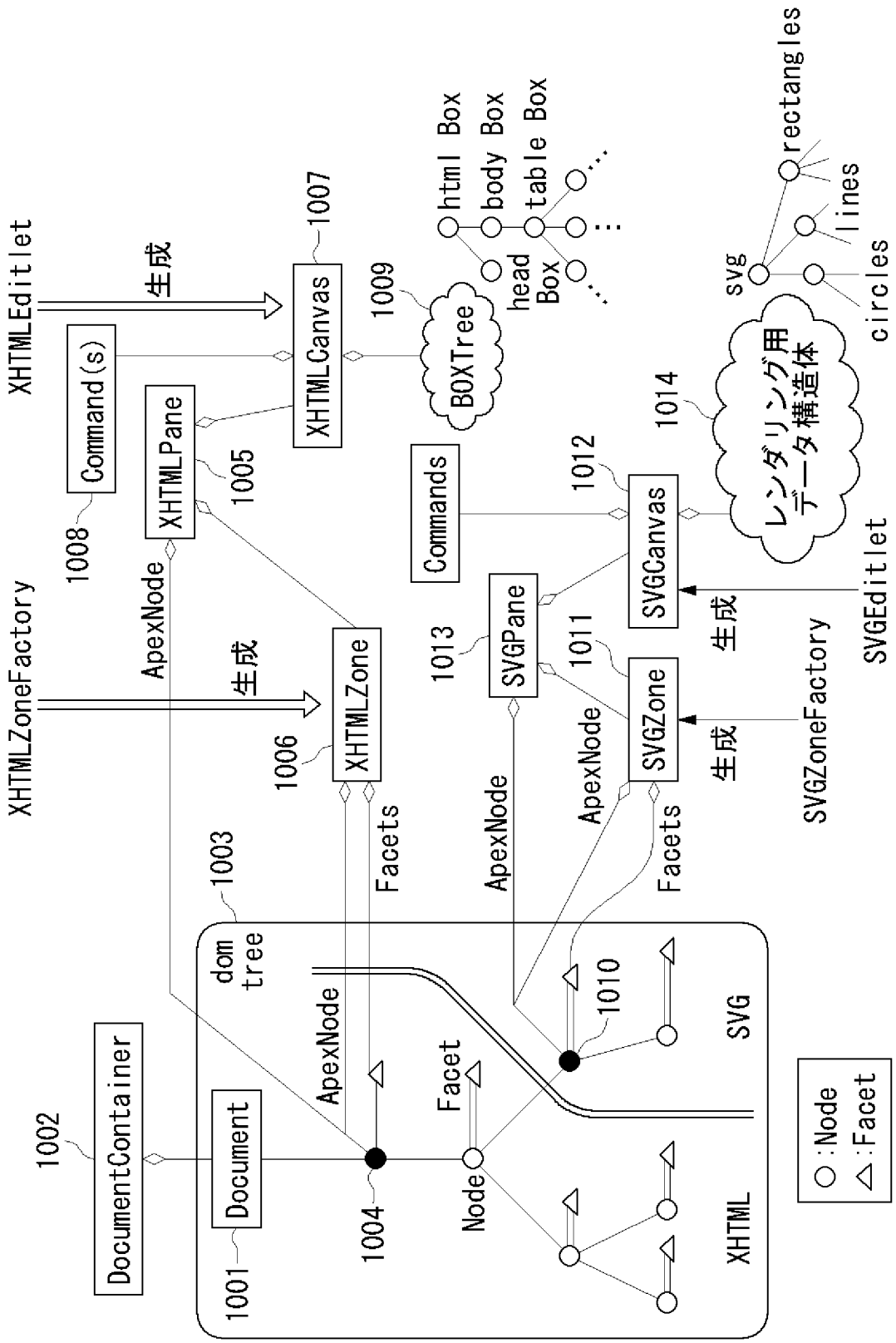

PaneOwner
908
STEP2
STEP3
DocumentManager
904
Pane
907
STEP1
DocumentContainer
903
STEP4
STEP6
Document
909
Zone
Canvas
Command
ApexNode
(XHTML)
905
910
STEP7
STEP5
ApexNode
(SVG)
906
STEP0
XHTML
SVG
レンダリング用
データ構造体
ストレージ
902
901
IOManager

(a)

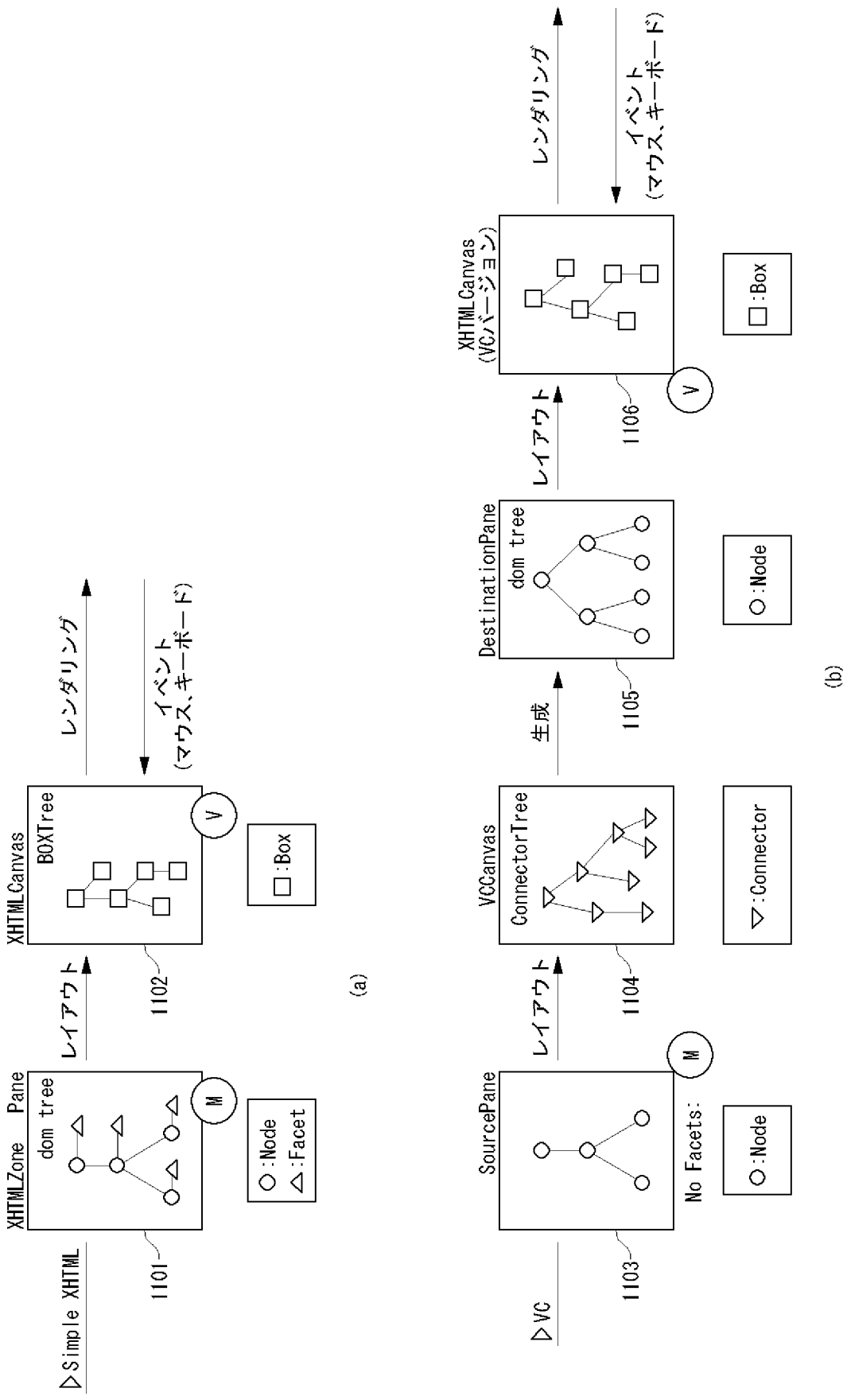

C
Command(s)
ZoneFactory
Editlet
M
V
生成
生成
Zone &
Facet(s)
Canvas &
データ構造体

(b)

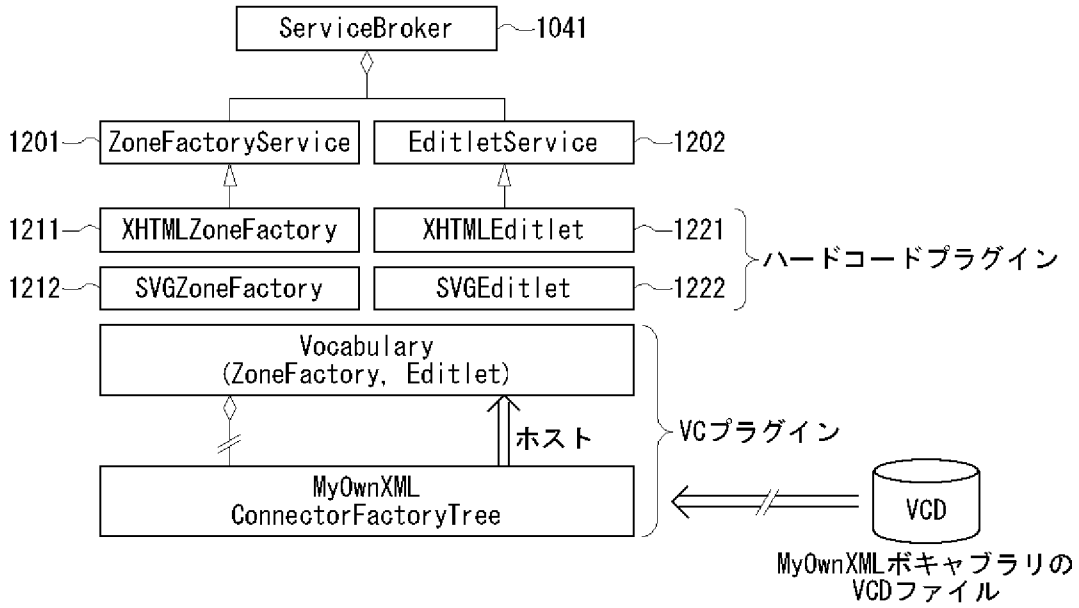
[図20]


XHTMLEditlet
生成
XHTMLCanvas
1007
1009
BOXTree
html Box
body Box
table Box
head Box
svg
rectangles
lines
circles
Command(s)
1008
XHTMLPane
1005
ApexNode
XHTMLZoneFactory
生成
XHTMLZone
1006
Facets
Commands
1012
SVGCanvas
1014
レンダリング用データ構造体
SVGEditlet
1013
SVGPane
1011
ApexNode
SVGZone
Facets
生成
SVGZoneFactory
1003
dom tree
ApexNode
Facet
1010
SVG
1002
DocumentContainer
Document
1001
1004
Node
XHTML
○:Node
△:Facet

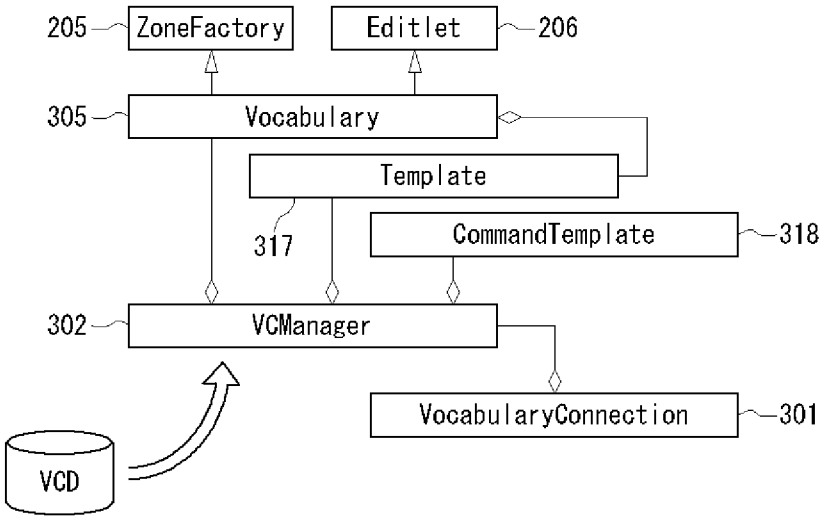
[図21]


▷Simple XHTML
XHTMLZone Pane
dom tree
M
1101
レイアウト
XHTMLCanvas
BOXTree
V
1102
レンダリング
イベント
(マウス、キーボード)
○:Node
△:Facet
□:Box
(a)
▷VC
SourcePane
M
No Facets:
1103
レイアウト
VCCanvas
ConnectorTree
1104
生成
DestinationPane
dom tree
1105
レイアウト
XHTMLCanvas
(VCバージョン)
V
1106
レンダリング
イベント
(マウス、キーボード)
○:Node
▽:Connector
□:Box
(b)

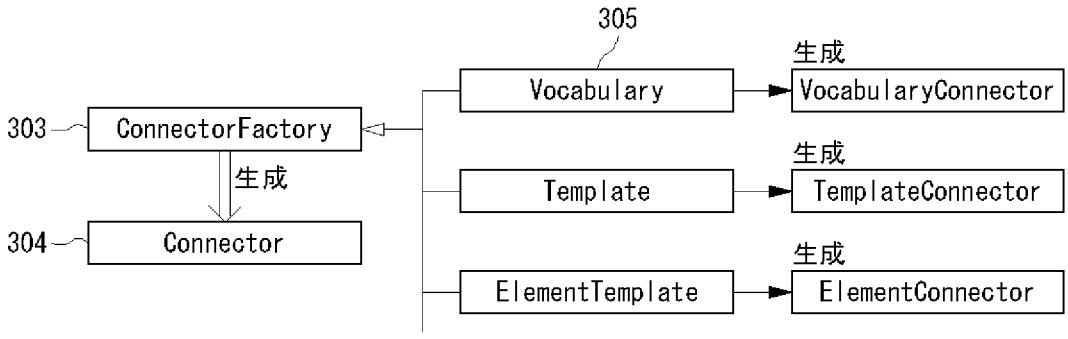
[図22]


ServiceBroker
1041
1201
ZoneFactoryService
EditletService
1202
1211
XHTMLZoneFactory
XHTMLEditlet
1221
ハードコードプラグイン
1212
SVGZoneFactory
SVGEditlet
1222
Vocabulary
(ZoneFactory, Editlet)
ホスト
VCプラグイン
MyOwnXML
ConnectorFactoryTree
VCD
MyOwnXMLボキャブラリの
VCDファイル

(a)

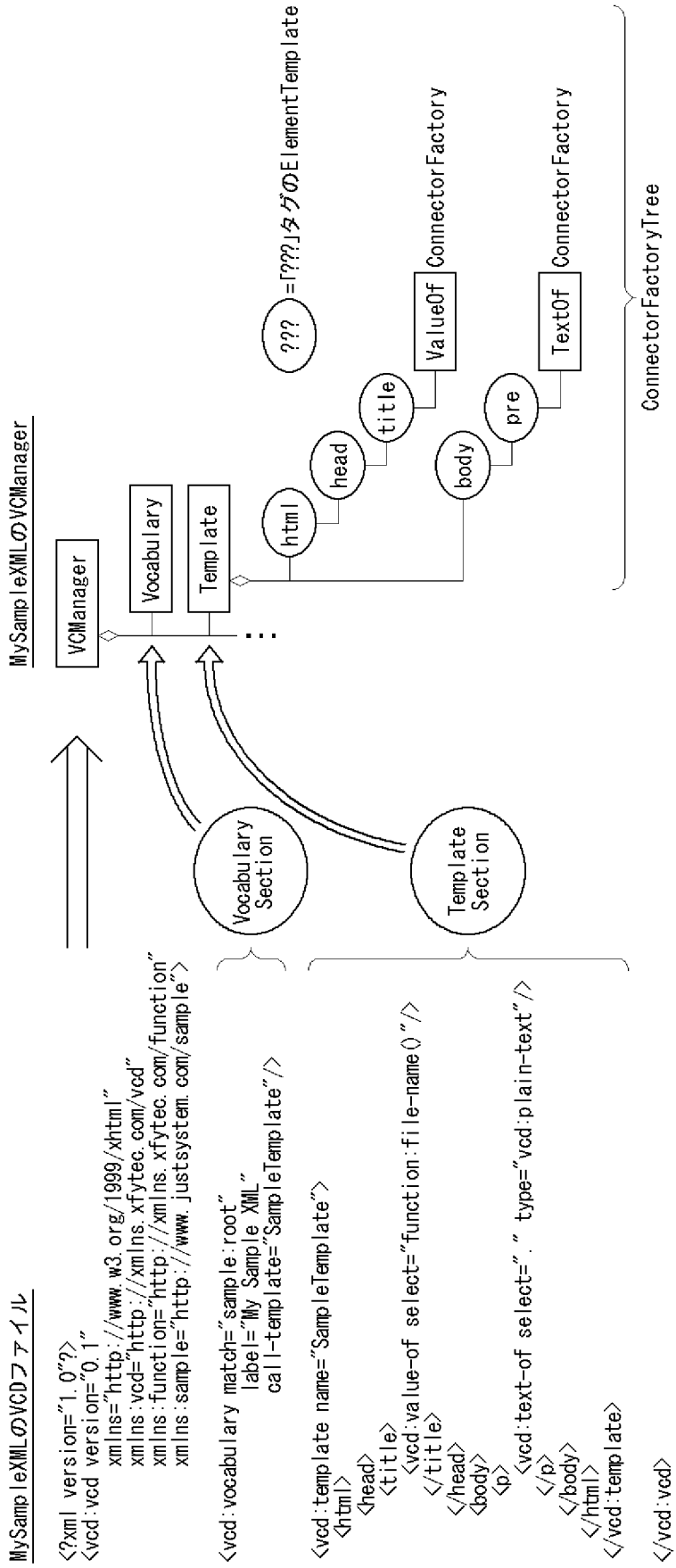

205
ZoneFactory
Editlet
206
305
Vocabulary
Template
317
CommandTemplate
318
302
VCManager
VocabularyConnection
301
VCD

(b)


305
生成
Vocabulary
VocabularyConnector
303
ConnectorFactory
生成
Template
TemplateConnector
304
Connector
生成
ElementTemplate
ElementConnector

(c)

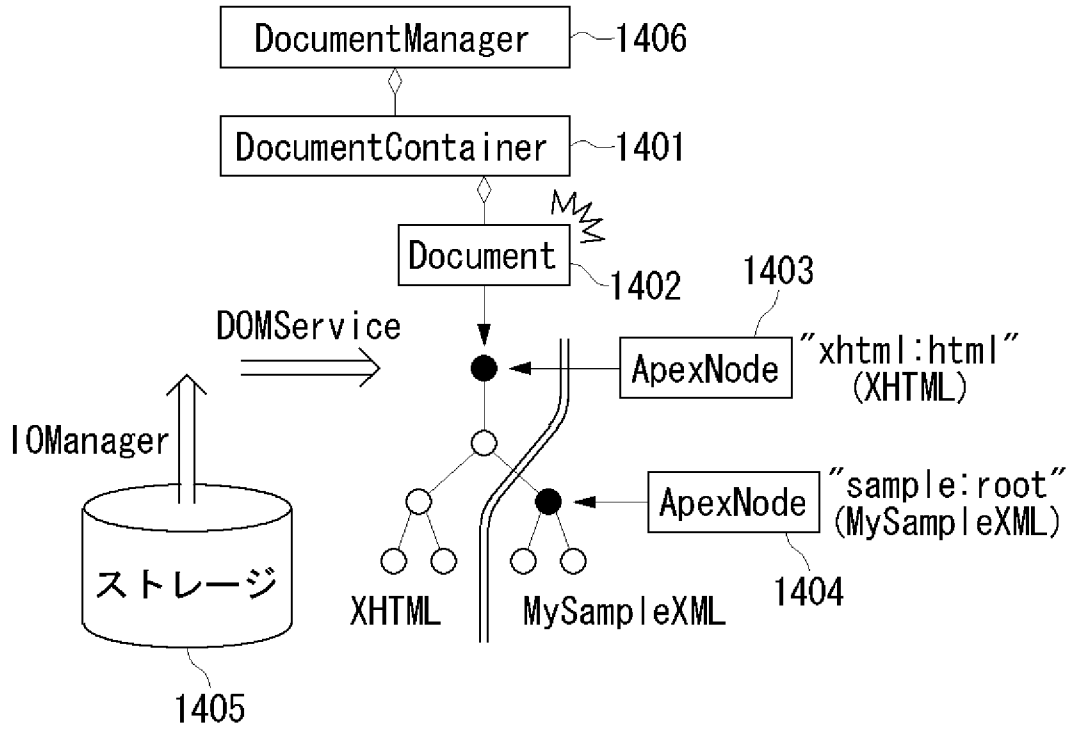
[図23]


MySampleXMLのVCファイル
<?xml version="1.0"?>
<vcd:vcd version="0.1"
xmlns="http://www.w3.org/1999/xhtml"
xmlns:vcd="http://xmlns.xfytec.com/vcd"
xmlns:function="http://xmlns.xfytec.com/function"
xmlns:sample="http://www.justsystem.com/sample">
<vcd:vocabulary match="sample:root"
label="My Sample XML"
call-template="SampleTemplate"/>
<vcd:template name="SampleTemplate">
<html>
<head>
<title>
<vcd:value-of select="function:file-name()"/>
</title>
</head>
<body>

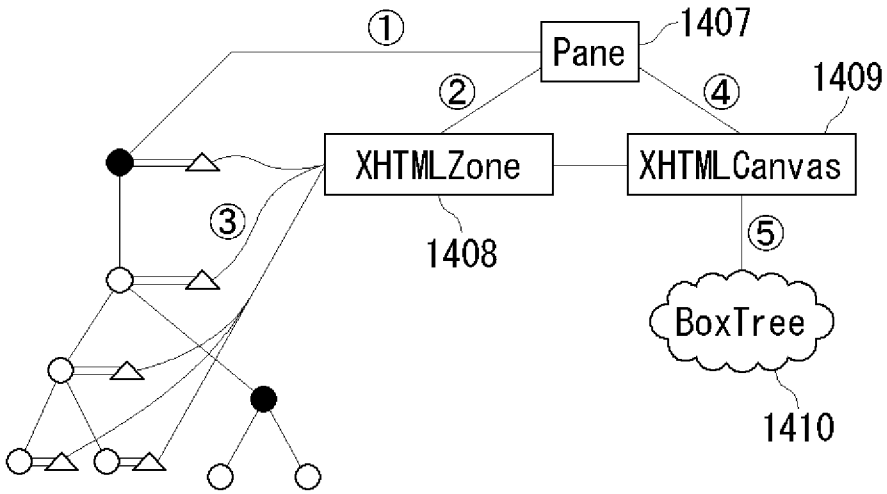
<vcd:text-of select="." type="vcd:plain-text"/>

</body>
</html>
</vcd:template>
</vcd:vcd>
MySampleXMLのVCManager
VCManager
Vocabulary
Template
Vocabulary Section
Template Section
html
head
title
body
pre
???
ValueOf
TextOf
ConnectorFactory
ConnectorFactory
???=「???」タグのElementTemplate
ConnectorFactoryTree

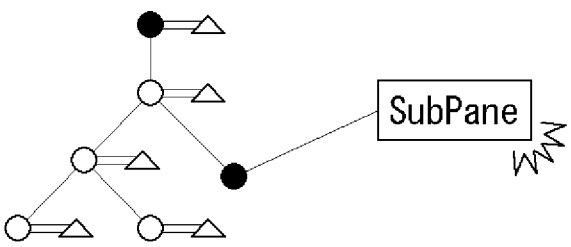
[図24]


DocumentManager
1406
DocumentContainer
1401
Document
1402
1403
DOMService
ApexNode
"xhtml:html"
(XHTML)
IOManager
ストレージ
1405
ApexNode
"sample:root"
(MySampleXML)
1404
XHTML
MySampleXML

(a)

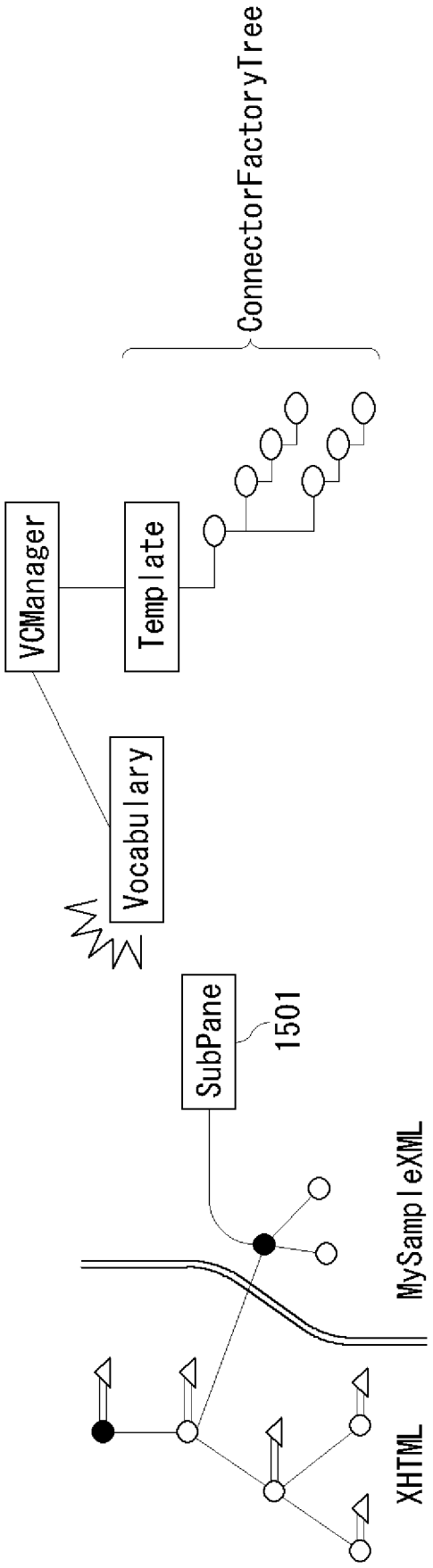

①
Pane
1407
②
④
1409
XHTMLZone
XHTMLCanvas
③
1408
⑤
BoxTree
1410

(b)

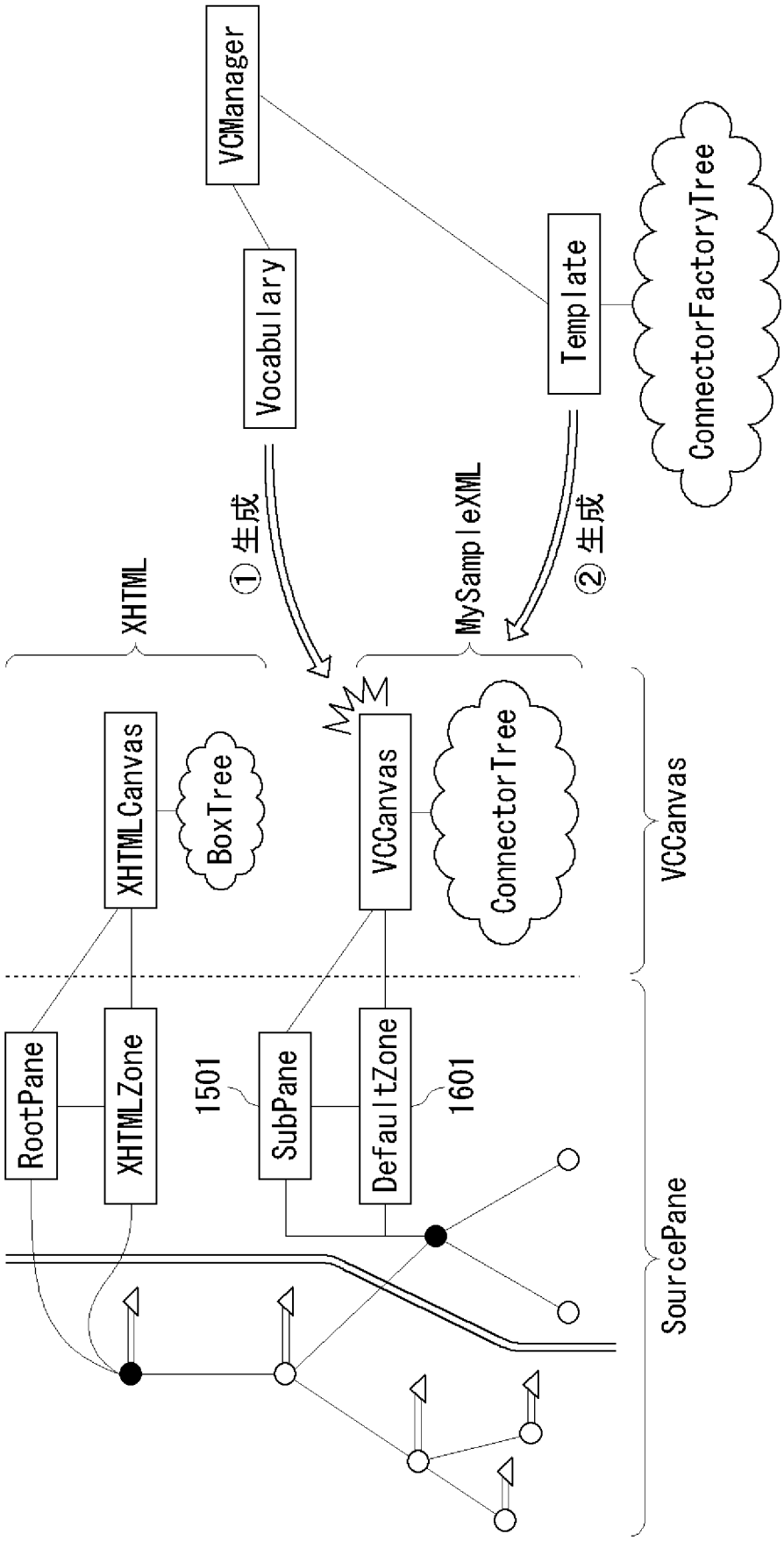

SubPane

(c)

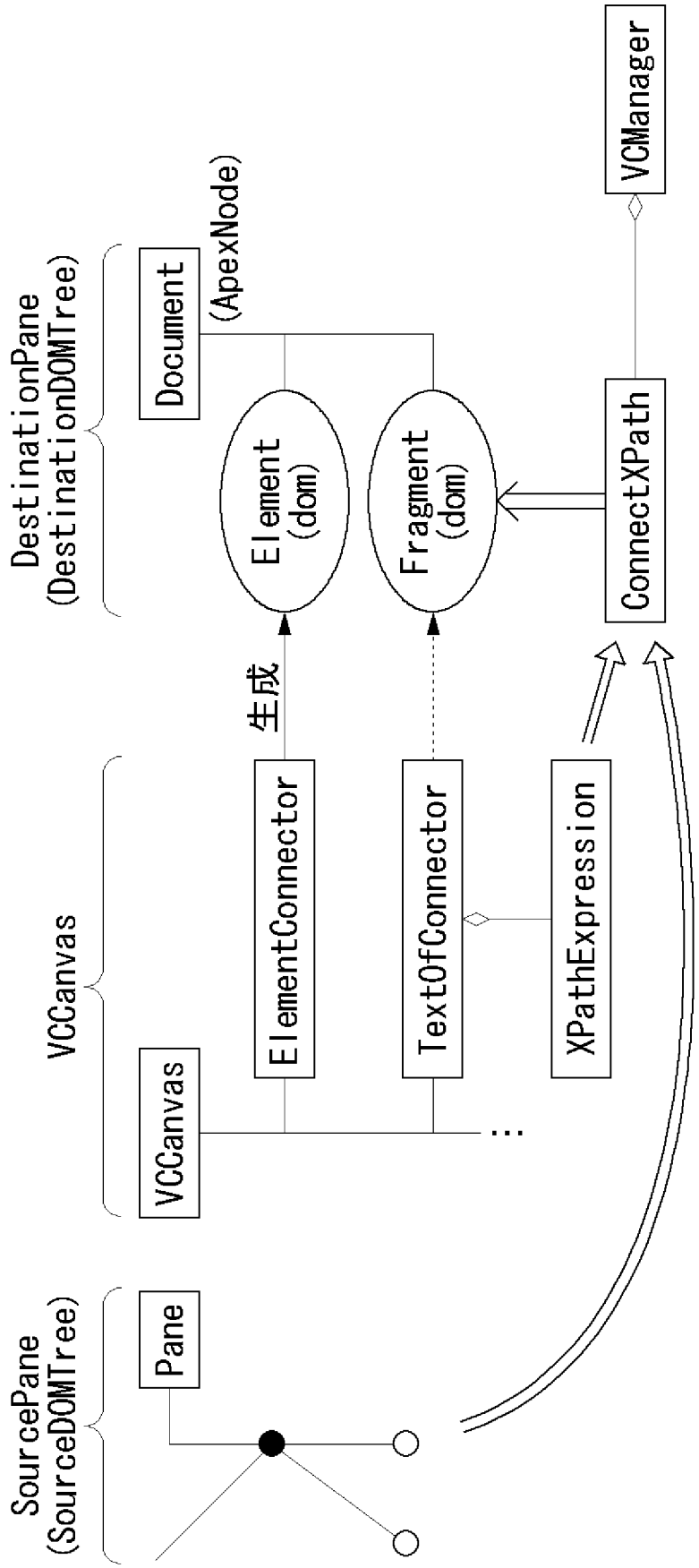
[図25]


ConnectorFactoryTree
VCManager
Template
Vocabulary
SubPane
1501
MySampleXML
XHTML

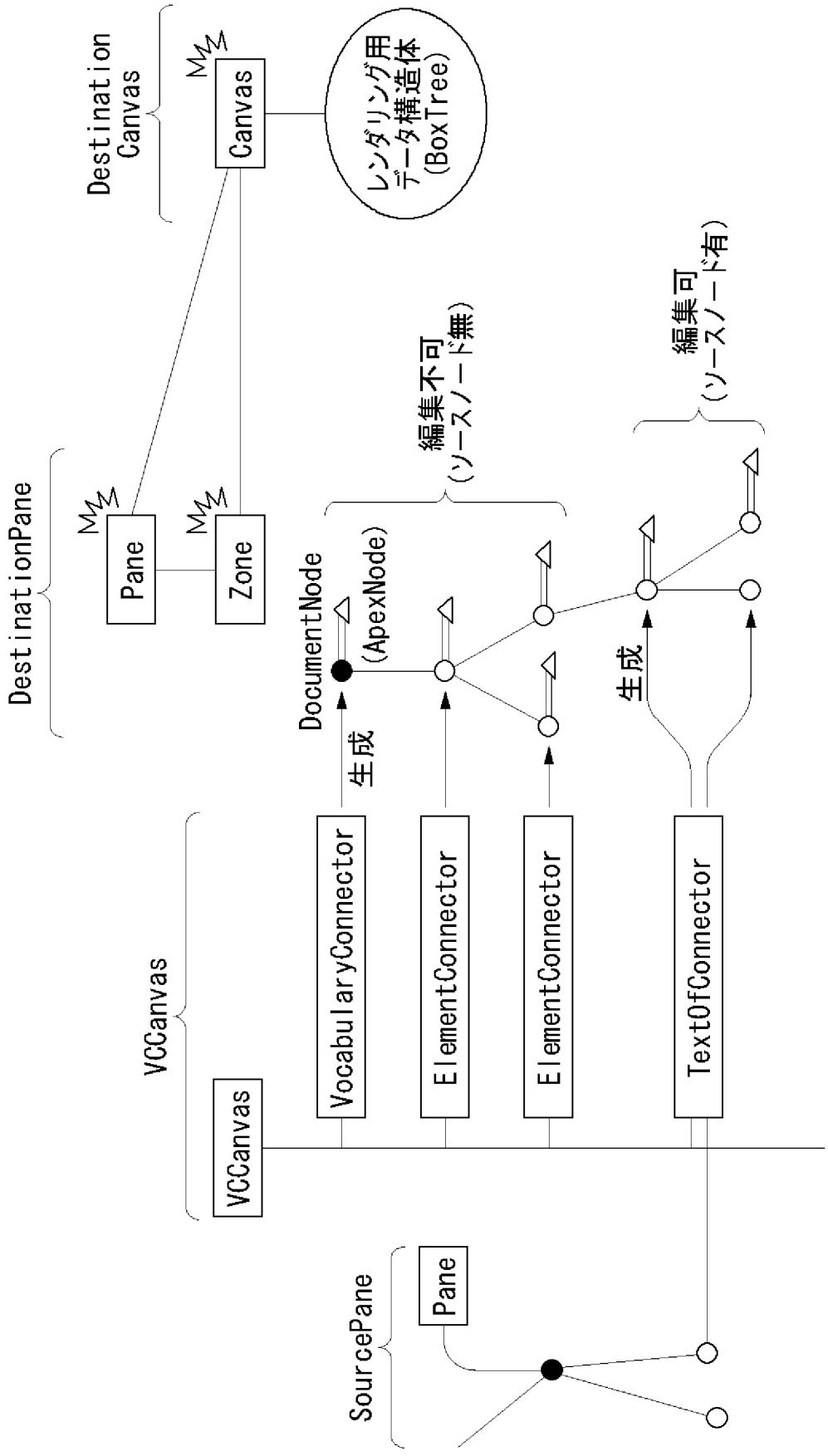
[図26]


VCManager
Vocabulary
Template
ConnectorFactoryTree
① 生成
② 生成
XHTML
MySampleXML
XHTMLCanvas
BoxTree
VCCanvas
ConnectorTree
VCCanvas
RootPane
XHTMLZone
1501
SubPane
DefaultZone
1601
SourcePane

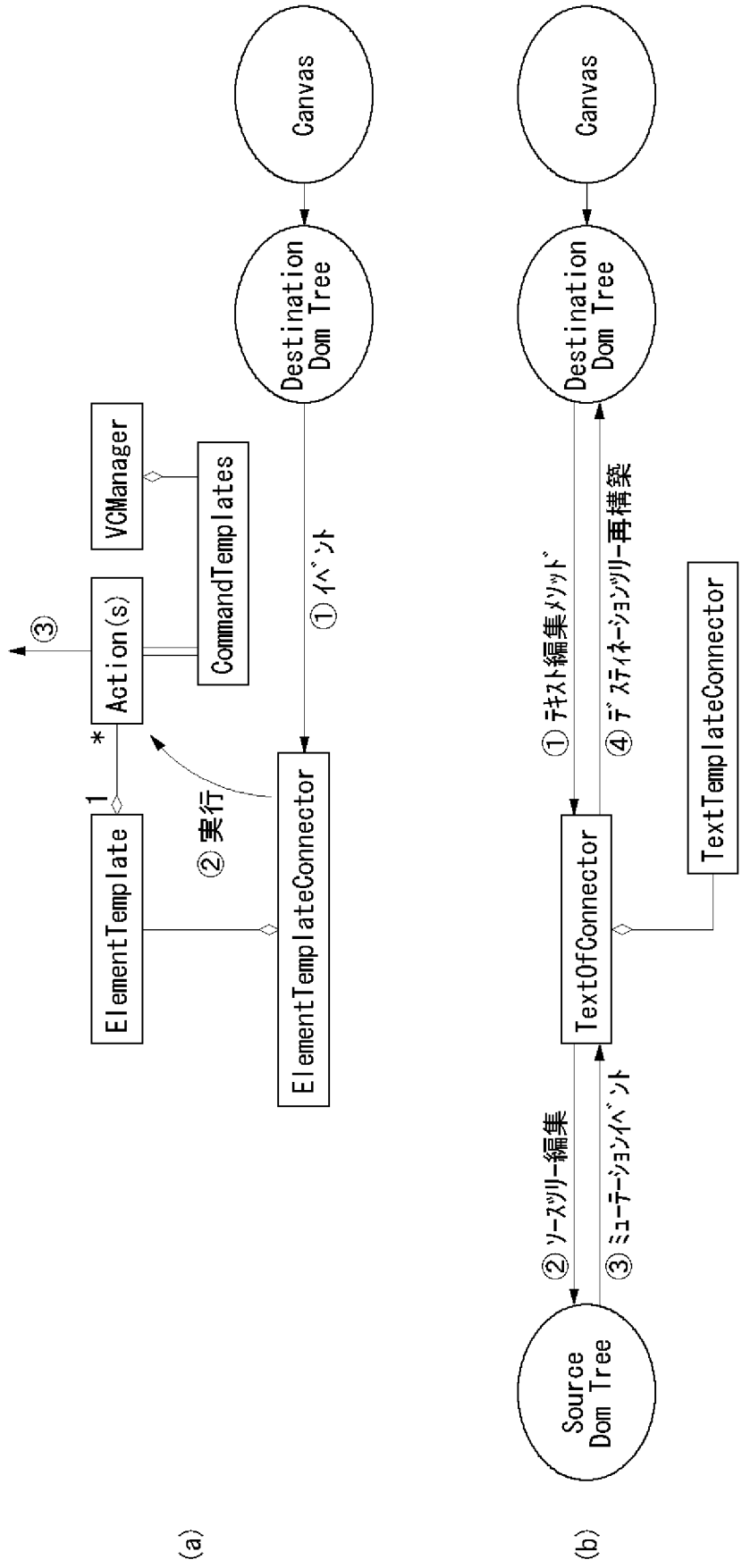
[図27]


VCManager
DestinationPane
(DestinationDOMTree)
Document
(ApexNode)
Element
(dom)
Fragment
(dom)
ConnectXPath
生成
VCCanvas
VCCanvas
ElementConnector
TextOfConnector
XPathExpression
...
SourcePane
(SourceDOMTree)
Pane

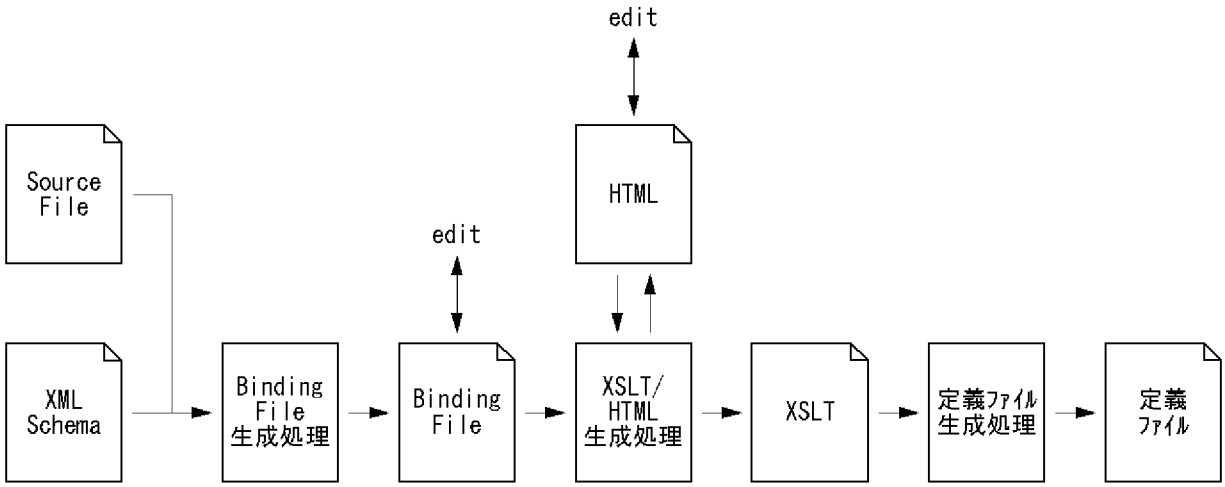
[図28]


Destination Canvas
Canvas
レンダリング用データ構造体 (BoxTree)
DestinationPane
Pane
Zone
編集不可 (ソースノード無)
編集可 (ソースノード有)
DocumentNode (ApexNode)
生成
生成
VCCanvas
VCCanvas
VocabularyConnector
ElementConnector
ElementConnector
TextOfConnector
SourcePane
Pane

[図29]


Canvas
Destination Dom Tree
① イベント
VCManager
CommandTemplates
③
Action(s)
*
1
② 実行
ElementTemplate
ElementTemplateConnector
(a)
Canvas
Destination Dom Tree
① テキスト編集メソッド
④ デスティネーションツリー再構築
TextTemplateConnector
TextOfConnector
② ソースツリー編集
③ ミューテーションイベント
Source Dom Tree
(b)

[図30]


edit
Source File
HTML
edit
XML Schema
Binding File 生成処理
Binding File
XSLT/ HTML 生成処理
XSLT
定義ファイル 生成処理
定義 ファイル

[図31]

```
<?xml version="1.0"?>
<xsd:schema xmlns:sfa="http://xmlns.xfytec.com/samples/sfa" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="customerList" type="customerListType"/>

    <xsd:complexType name="customerListType">
        <xsd:sequence>
            <xsd:element name="sfa:listID" type="xsd:string"/>
            <xsd:element name="sfa:totalEstimate" type="xsd:decimal"/>
            <xsd:element name="sfa:totalNumber" type="xsd:integer"/>
            <xsd:element name="sfa:customer" type="customerType" maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="customerType">
        <xsd:sequence>
            <xsd:element name="sfa:name" type="xsd:string"/>
            <xsd:element name="sfa:uid" type="xsd:string"/>
            <xsd:element name="sfa:project" type="xsd:string"/>
            <xsd:element name="sfa:activity" type="activityListType"/>
        </xsd:sequence>
        <xsd:attribute name="sfa:id" type="xsd:string"/>

    </xsd:complexType>

                                    .
                                    .
                                    .

</xsd:schema>
```

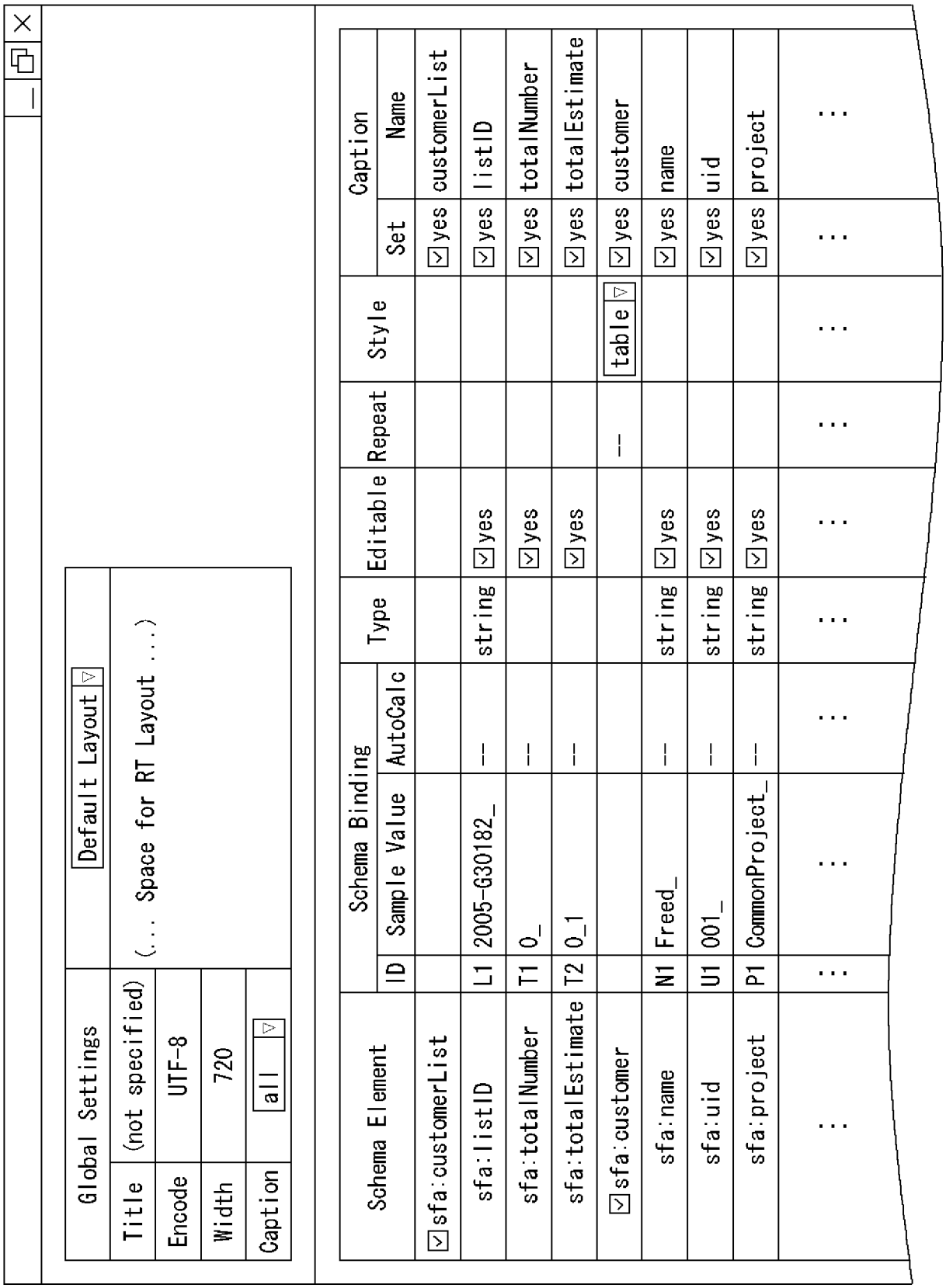
[図32]

```
<?xml version="1.0"?>
<sfa:customerList xmlns="http://xmlns.xfytec.com/samples/sfa"
        xmlns:sfa="http://xmlns.xfytec.com/samples/sfa">

    <sfa:listID>2005-G30182</sfa:listID>
    <sfa:totalNumber>3</sfa:totalNumber>
    <sfa:totalEstimate>1000</sfa:totalEstimate>

    <sfa:customer id="ZX10A">
            <sfa:name>Freed</sfa:name>
            <sfa:uid>001</sfa:uid>
            <sfa:project>CommonProject</sfa:project>
            <sfa:activity>
                    <sfa:date>2005-06-24</sfa:date>
                    <sfa:currentstate>Not Available</sfa:currentstate>
                    <sfa:estimate>1000</sfa:estimate>
                    <sfa:action>Nothing</sfa:action>
            </sfa:activity>
    </sfa:customer>
    <sfa:customer id="ZX20A">
            <sfa:name>StFreed</sfa:name>
            <sfa:uid>002</sfa:uid>
                    ...
    </sfa:customer>
    <sfa:customer id="ZX19A">
                    ....
    </sfa:customer>

</sfa:customerList>
```

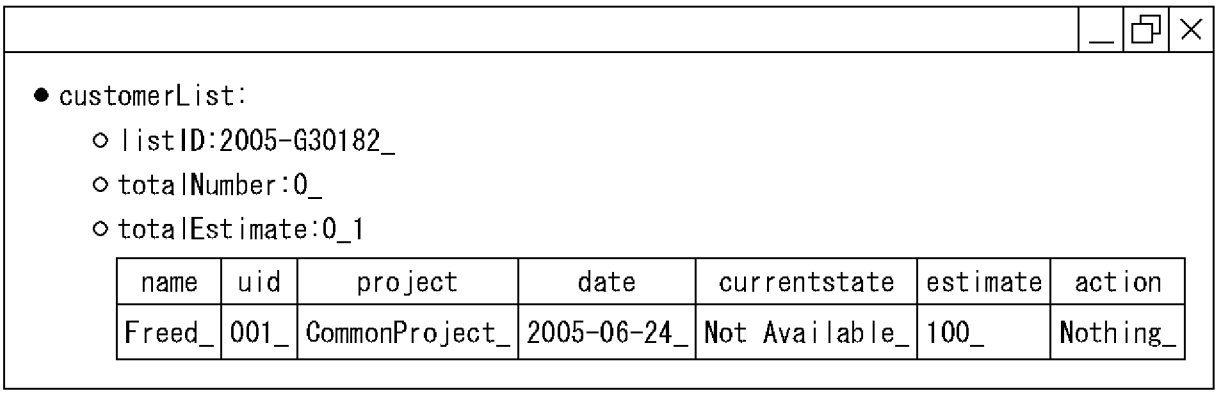
[図33]

```
<xvcd:template match="sfa:customerList/sfa:listID">
<xhtml:li>
 <xhtml:font color="blue" size="-1">
    <xhtml:i>listID:</xhtml:i>
  </xhtml:font>
  <xvcd:text-of select="." type="xsd:string" />
</xhtml:li>
<xvcd:apply-templates select="../sfa:totalNumber" />

</xvcd:template>

<xvcd:template match="sfa:customerList/sfa:totalNumber">
<xhtml:li>
 <xhtml:font color="blue" size="-1">
    <xhtml:i>totalNumber:</xhtml:i>
  </xhtml:font>
  <xvcd:value-of select="." />
</xhtml:li>
<xvcd:apply-templates select="../sfa:totalEstimate" />

</xvcd:template>
```

[図34]


Global Settings
Default Layout
Title
(not specified)
(... Space for RT Layout ...)
Encode
UTF-8
Width
720
Caption
all
Schema Element
Schema Binding
ID
Sample Value
AutoCalc
Type
Editable
Repeat
Style
Caption
Set
Name
sfa:customerList
sfa:listID
L1
2005-G30182_
string
yes
sfa:totalNumber
T1
0_
sfa:totalEstimate
T2
0_1
sfa:customer
table
sfa:name
N1
Freed_
sfa:uid
U1
001_
sfa:project
P1
CommonProject_

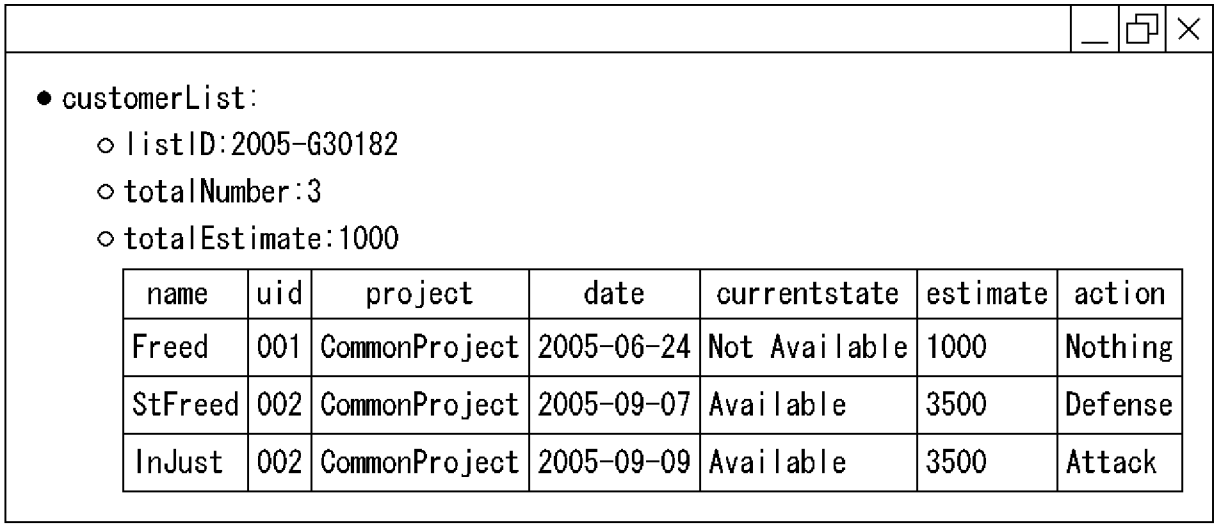
[図35]



- customerList:
  - listID:2005-G30182_
  - totalNumber:0_
  - totalEstimate:0_1

name	uid	project	date	currentstate	estimate	action
Freed_	001_	CommonProject_	2005-06-24_	Not Available_	100_	Nothing_

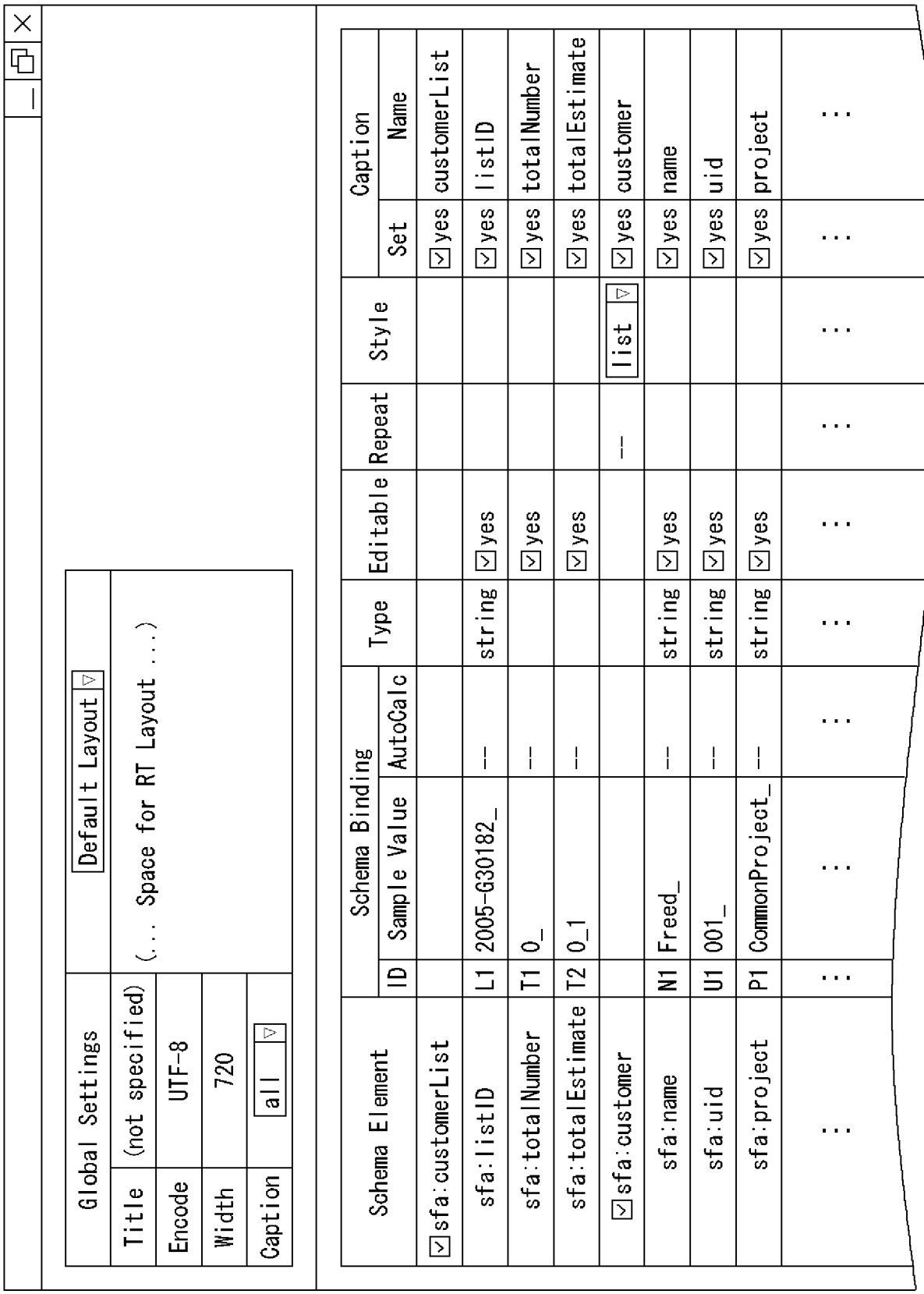
[図36]



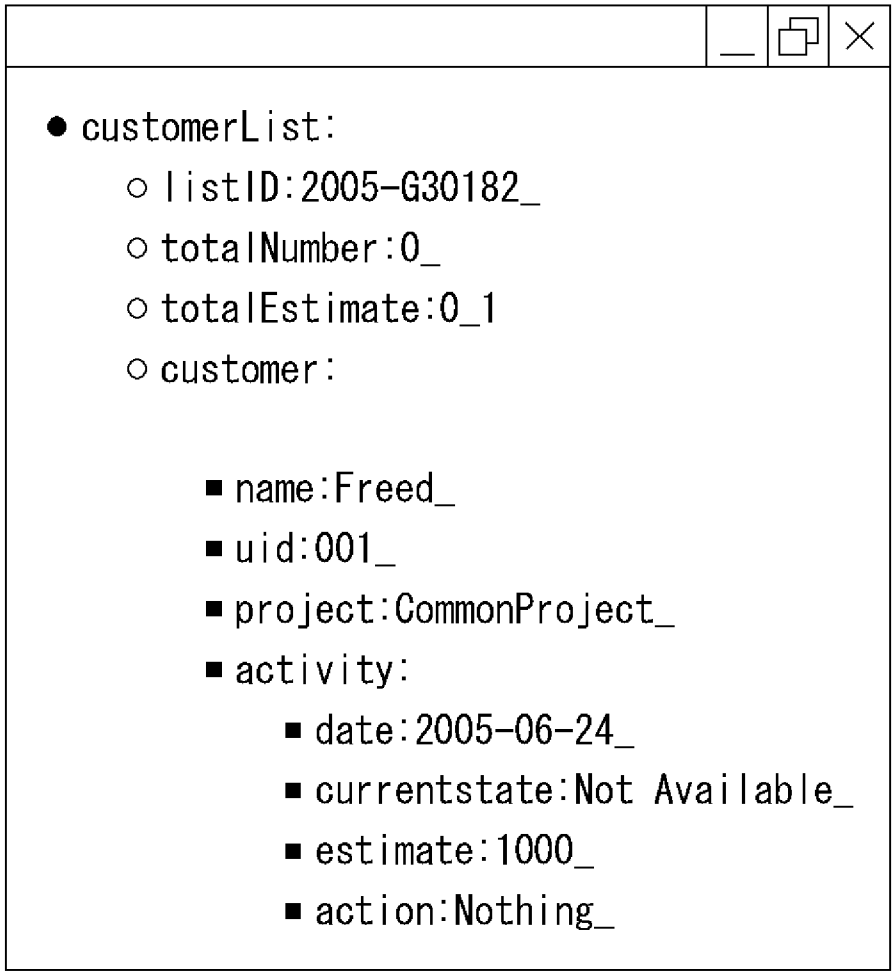
- customerList:
  - listID:2005-G30182
  - totalNumber:3
  - totalEstimate:1000

name	uid	project	date	currentstate	estimate	action
Freed	001	CommonProject	2005-06-24	Not Available	1000	Nothing
StFreed	002	CommonProject	2005-09-07	Available	3500	Defense
InJust	002	CommonProject	2005-09-09	Available	3500	Attack

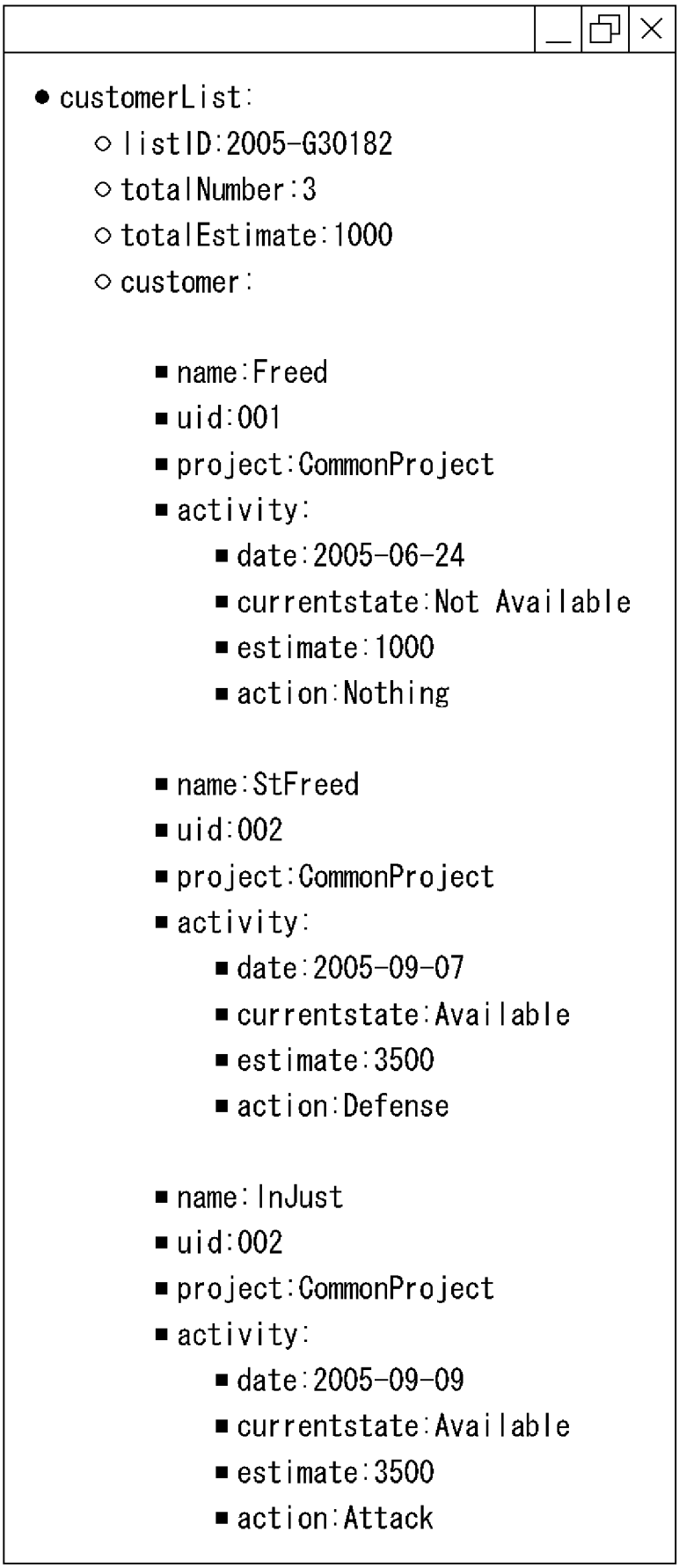
[図37]


Global Settings
Default Layout
Title
(not specified)
(... Space for RT Layout ...)
Encode
UTF-8
Width
720
Caption
all
Schema Element
Schema Binding
ID
Sample Value
AutoCalc
Type
Editable
Repeat
Style
Caption
Set
Name
sfa:customerList
sfa:listID
L1
2005-G30182_
string
yes
listID
sfa:totalNumber
T1
0_
totalNumber
sfa:totalEstimate
T2
0_1
totalEstimate
sfa:customer
list
customer
sfa:name
N1
Freed_
name
sfa:uid
U1
001_
uid
sfa:project
P1
CommonProject_
project

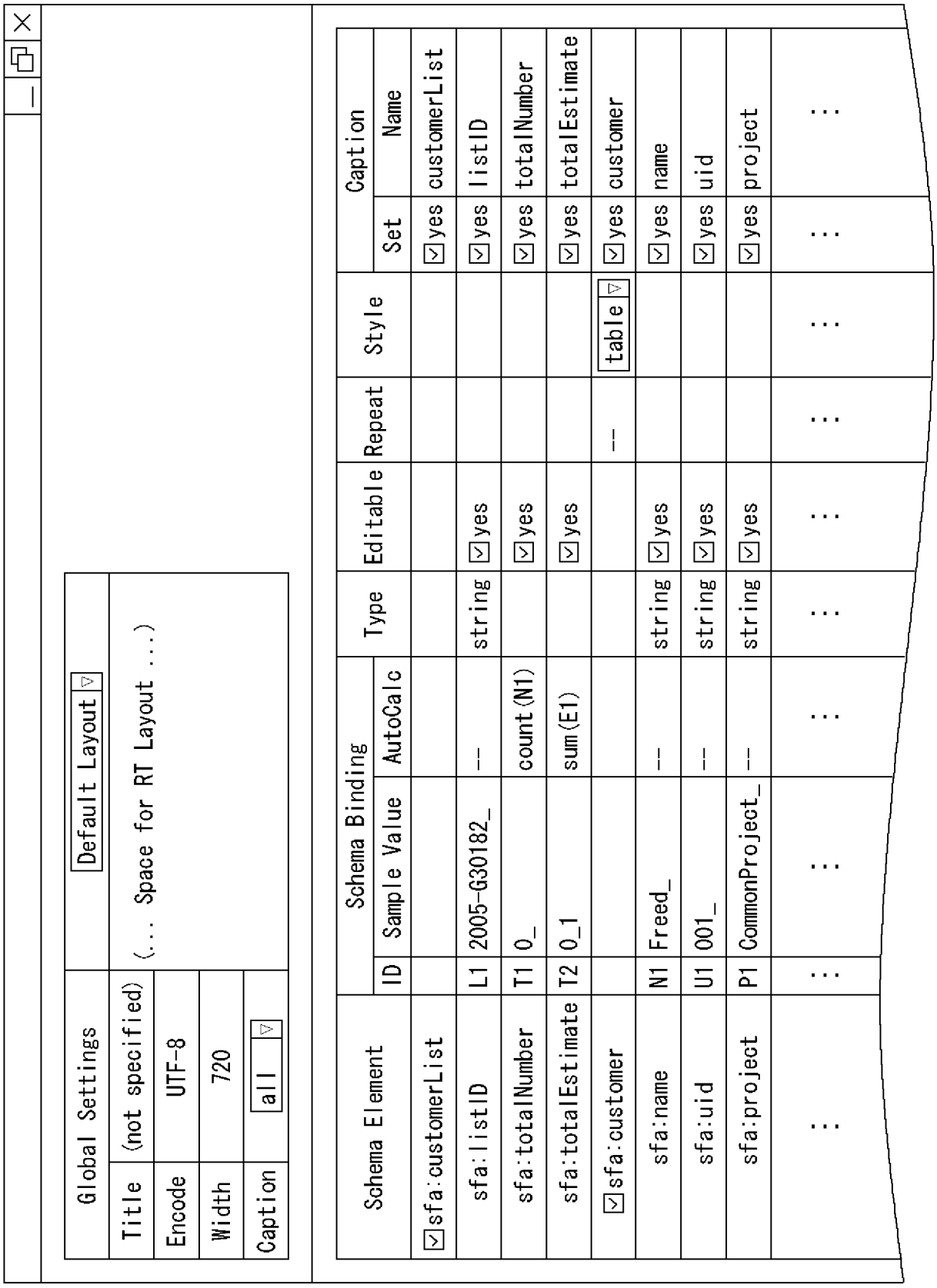
[図38]


● customerList:
○ listID:2005-G30182_
○ totalNumber:0_
○ totalEstimate:0_1
○ customer:
■ name:Freed_
■ uid:001_
■ project:CommonProject_
■ activity:
■ date:2005-06-24_
■ currentstate:Not Available_
■ estimate:1000_
■ action:Nothing_

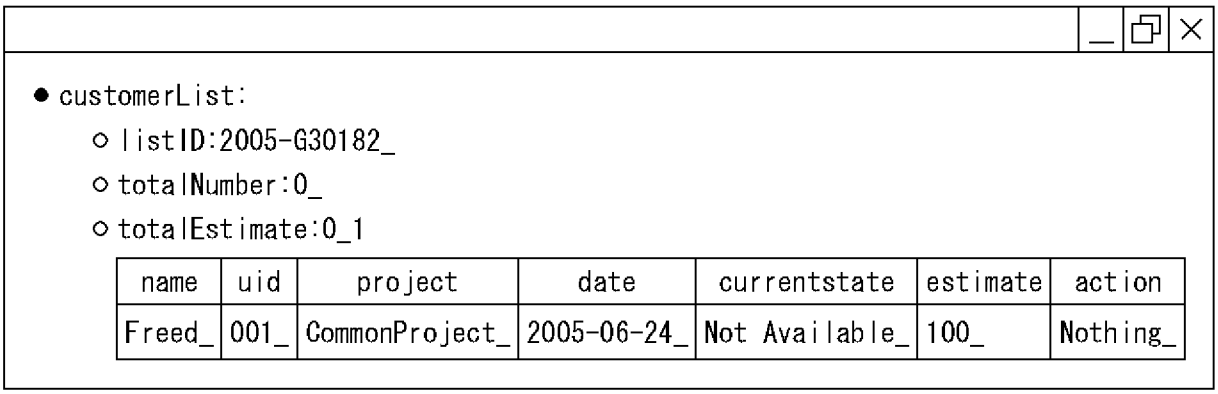
[図39]


customerList:
listID:2005-G30182
totalNumber:3
totalEstimate:1000
customer:
name:Freed
uid:001
project:CommonProject
activity:
date:2005-06-24
currentstate:Not Available
estimate:1000
action:Nothing
name:StFreed
uid:002
project:CommonProject
activity:
date:2005-09-07
currentstate:Available
estimate:3500
action:Defense
name:InJust
uid:002
project:CommonProject
activity:
date:2005-09-09
currentstate:Available
estimate:3500
action:Attack

[図40]


Global Settings
Default Layout
Title
(not specified)
(... Space for RT Layout ...)
Encode
UTF-8
Width
720
Caption
all
Schema Element
Schema Binding
ID
Sample Value
AutoCalc
Type
Editable
Repeat
Style
Caption
Set
Name
sfa:customerList
sfa:listID
L1
2005-G30182_
--
string
yes
customerList
sfa:totalNumber
T1
0_
count(N1)
yes
listID
sfa:totalEstimate
T2
0_1
sum(E1)
totalNumber
sfa:customer
table
totalEstimate
N1
Freed_
customer
sfa:name
U1
001_
name
sfa:uid
P1
CommonProject_
uid
sfa:project
project

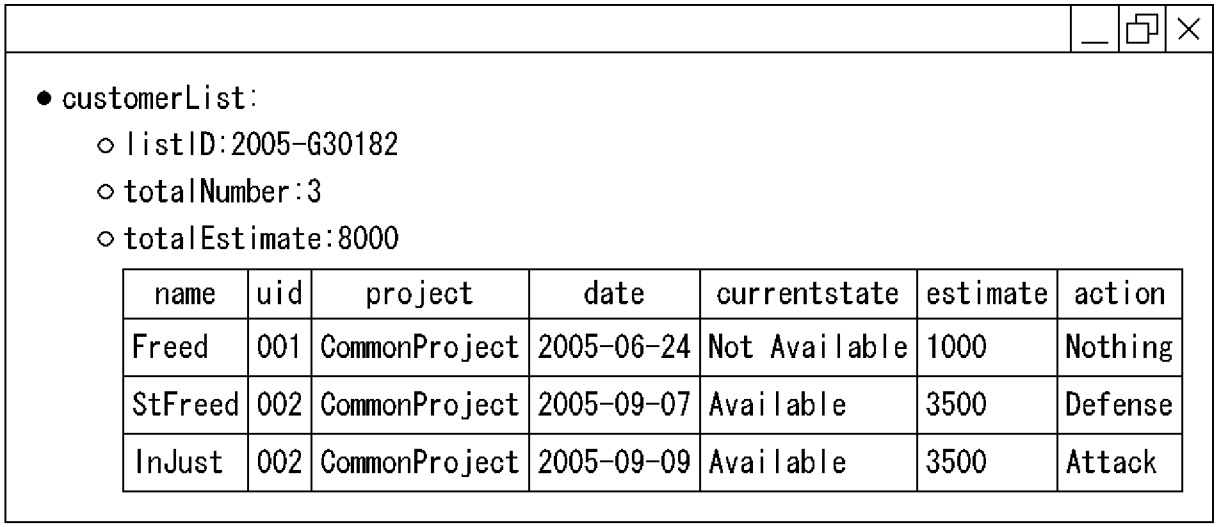
[図41]



- customerList:
  - listID:2005-G30182_
  - totalNumber:0_
  - totalEstimate:0_1

name	uid	project	date	currentstate	estimate	action
Freed_	001_	CommonProject_	2005-06-24_	Not Available_	100_	Nothing_

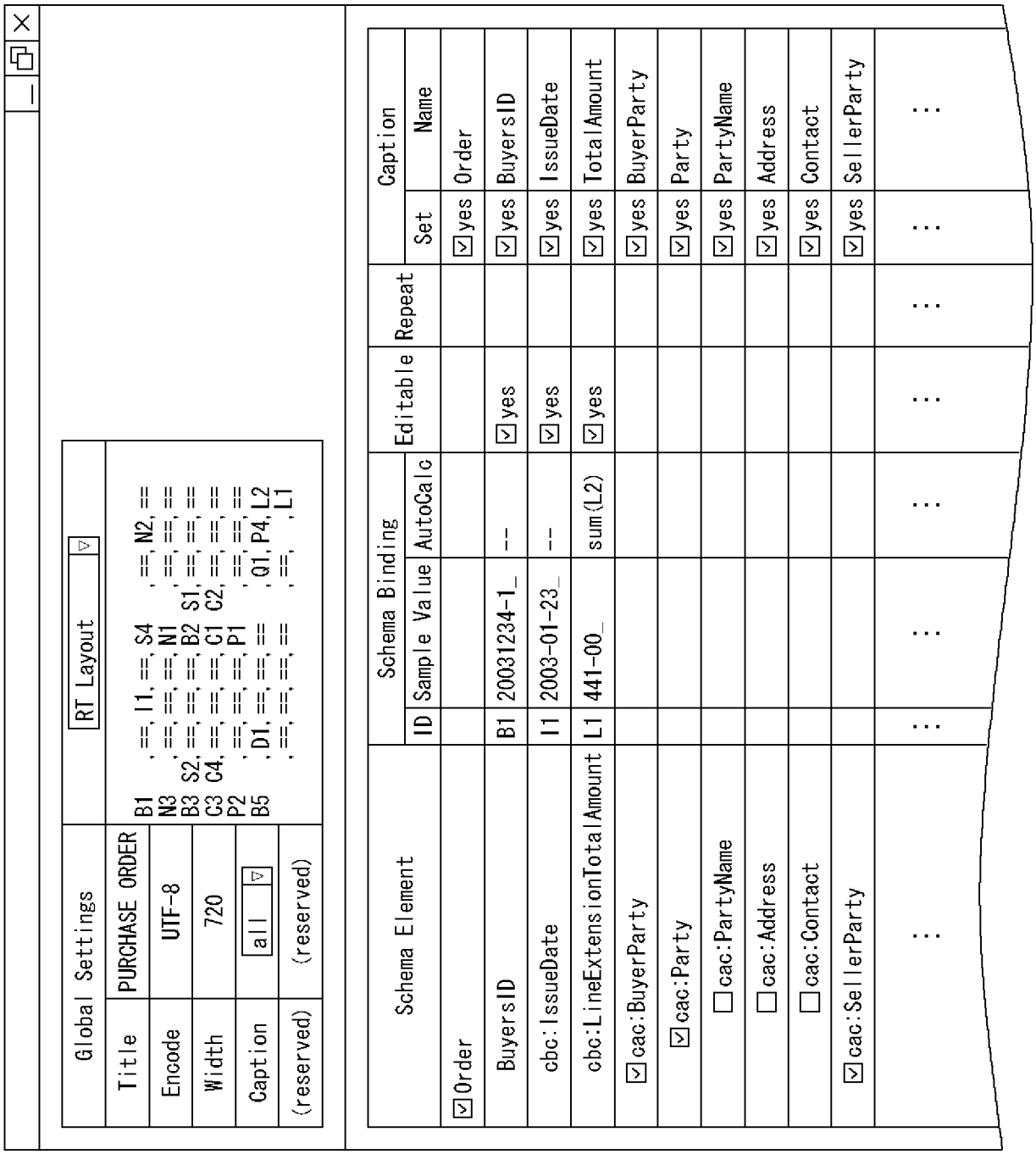
[図42]



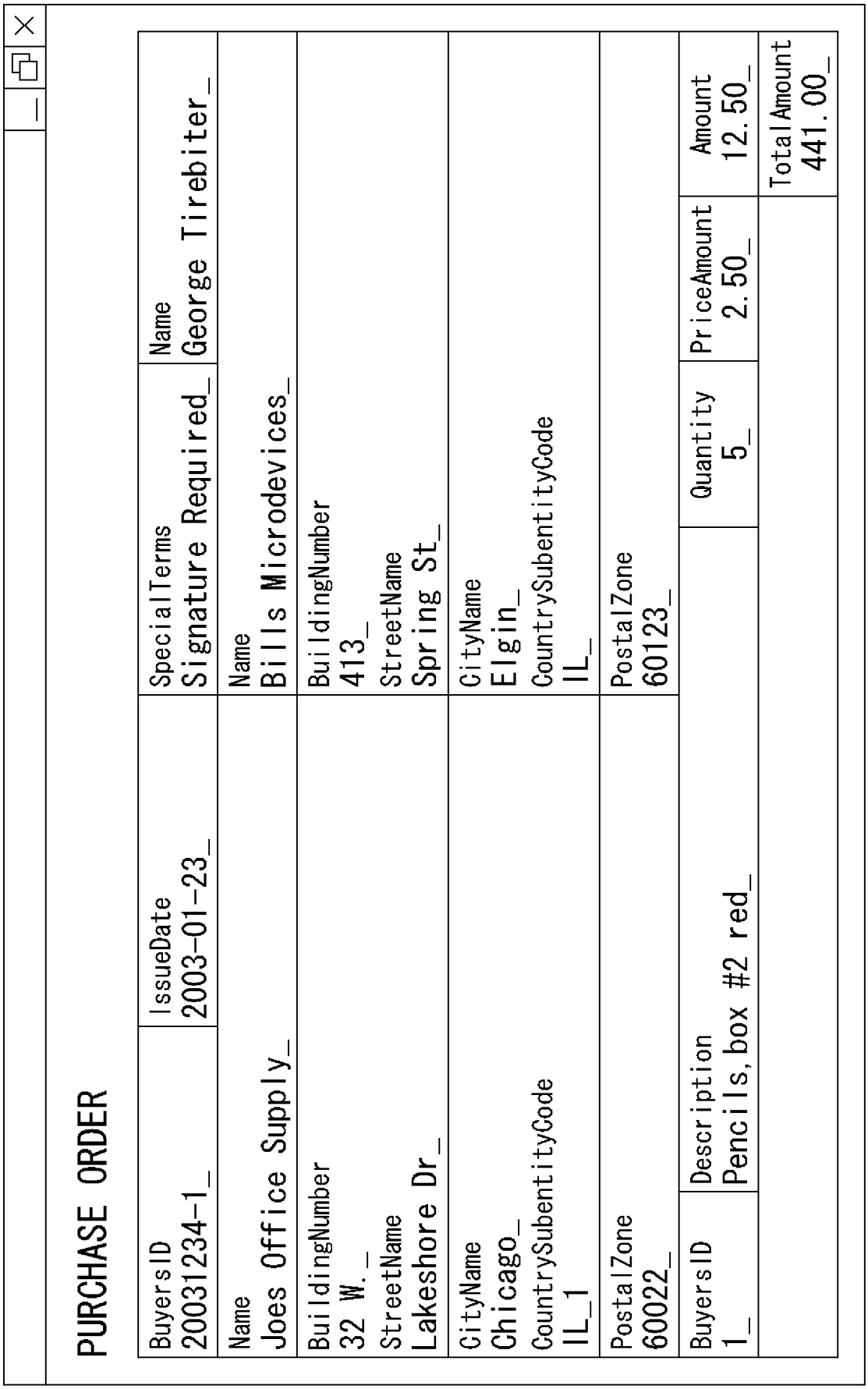
- customerList:
  - listID:2005-G30182
  - totalNumber:3
  - totalEstimate:8000

name	uid	project	date	currentstate	estimate	action
Freed	001	CommonProject	2005-06-24	Not Available	1000	Nothing
StFreed	002	CommonProject	2005-09-07	Available	3500	Defense
InJust	002	CommonProject	2005-09-09	Available	3500	Attack

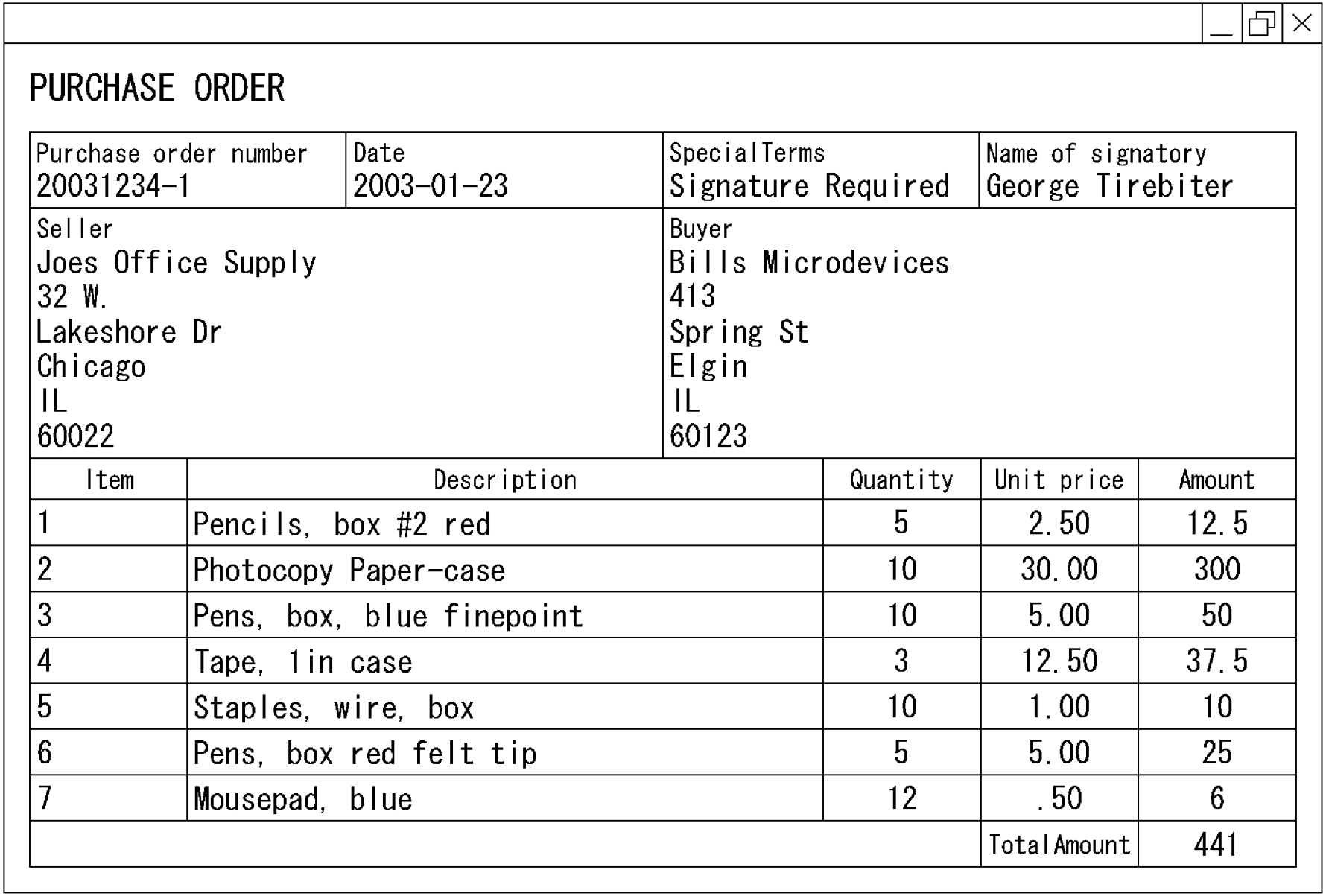
[図43]


Global Settings
RT Layout
Title
PURCHASE ORDER
Encode
UTF-8
Width
720
Caption
all
(reserved)
(reserved)
B1 ,==,I1,==,S4 ,==,N2,==
N3 ,==,==,==,N1 ,==,==,==
B3 S2,==,==,==,B2 S1,==,==,==
C3 C4,==,==,==,C1 C2,==,==,==
P2 ,==,==,==,P1 ,==,==,==
B5 ,D1,==,==,== ,Q1,P4,L2
,==,==,==,== ,==, ,L1
Schema Element
Schema Binding
ID
Sample Value
AutoCalc
Editable
Repeat
Caption
Set
Name
Order
BuyersID
IssueDate
cbc:LineExtensionTotalAmount
cac:BuyerParty
cac:Party
cac:PartyName
cac:Address
cac:Contact
cac:SellerParty
B1 20031234-1_ --
I1 2003-01-23_ --
L1 441-00_ sum(L2)
yes
Order
BuyersID
IssueDate
TotalAmount
BuyerParty
Party
PartyName
Address
Contact
SellerParty

[図44]


PURCHASE ORDER
BuyersID
20031234-1_
IssueDate
2003-01-23_
SpecialTerms
Signature Required_
Name
George Tirebiter_
Name
Joes Office Supply_
Name
Bills Microdevices_
BuildingNumber
32 W._
StreetName
Lakeshore Dr_
BuildingNumber
413_
StreetName
Spring St_
CityName
Chicago_
CountrySubentityCode
IL_1
CityName
Elgin_
CountrySubentityCode
IL_
PostalZone
60022_
PostalZone
60123_
BuyersID
1_
Description
Pencils,box #2 red_
Quantity
5_
PriceAmount
2.50_
Amount
12.50_
TotalAmount
441.00_

[図45]



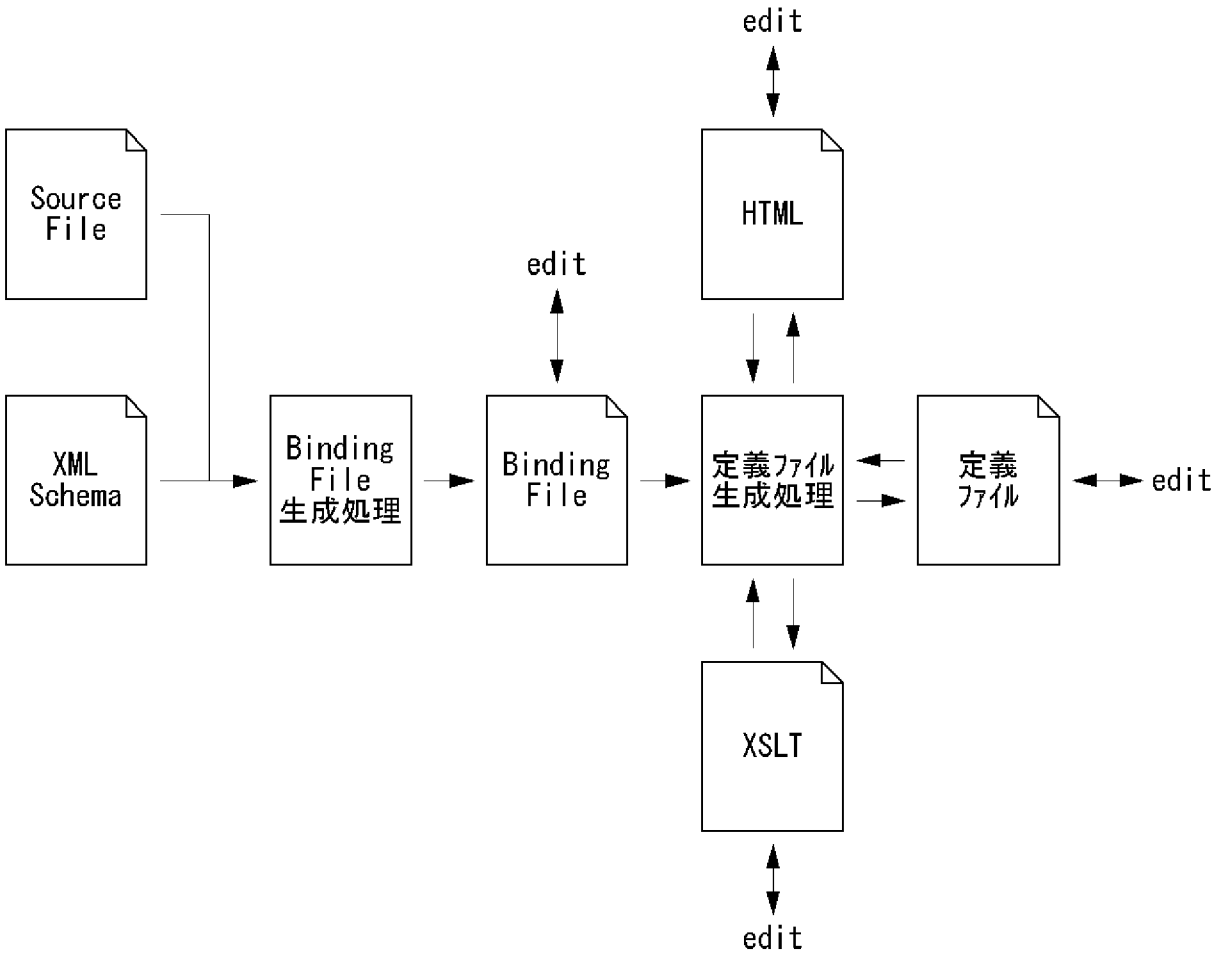
PURCHASE ORDER

Purchase order number	Date	SpecialTerms	Name of signatory
20031234-1	2003-01-23	Signature Required	George Tirebiter

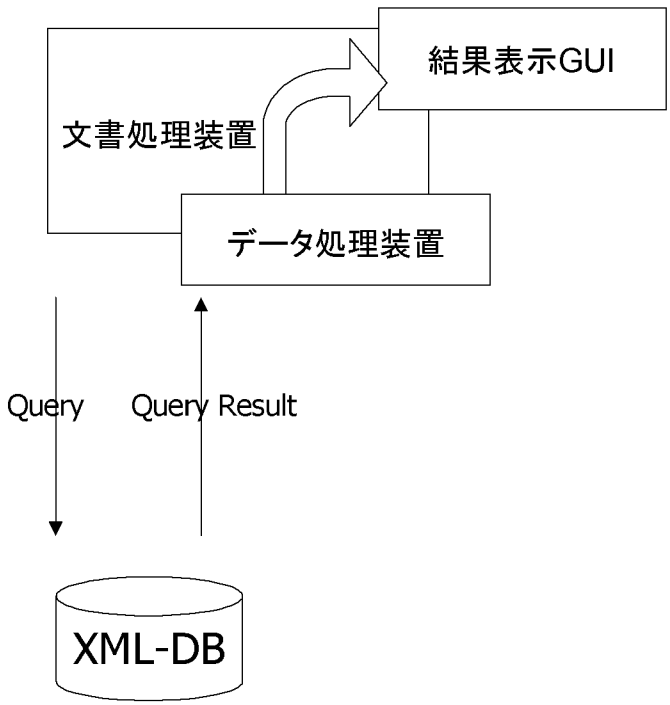
Seller	Buyer
Joes Office Supply 32 W. Lakeshore Dr Chicago IL 60022	Bills Microdevices 413 Spring St Elgin IL 60123

Item	Description	Quantity	Unit price	Amount
1	Pencils, box #2 red	5	2.50	12.5
2	Photocopy Paper-case	10	30.00	300
3	Pens, box, blue finepoint	10	5.00	50
4	Tape, 1in case	3	12.50	37.5
5	Staples, wire, box	10	1.00	10
6	Pens, box red felt tip	5	5.00	25
7	Mousepad, blue	12	.50	6
			TotalAmount	441

[図46]


edit
Source File
HTML
edit
XML Schema
Binding File 生成処理
Binding File
定義ファイル 生成処理
定義 ファイル
edit
XSLT
edit

[図47]


結果表示GUI
文書処理装置
データ処理装置
Query
Query Result
XML-DB

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2006/320488

A. CLASSIFICATION OF SUBJECT MATTER
G06F3/14(2006.01)i, *G06F17/21*(2006.01)i

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)
G06F3/14, G06F17/21

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	1996-2006
Kokai Jitsuyo Shinan Koho	1971-2006	Toroku Jitsuyo Shinan Koho	1994-2006

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.
Y	Technology Overview of Microsoft Office Info Path 2003, Microsoft Corporation, online, 21 August, 2003 (21.08.03), [retrieval date 08 December, 2006 (08.12.06)], Internet<URL: http://www.microsoft.com/technet/prodtechnol/office/office2003/plan/iptechov.mspx>	1-9
Y	Megumi NISHIMURA, "Access & Office 2003 Shinkino Tettei Guide InfoPath no Junanna XML Henshu Kino de Access Database no Katsuyo Han'i ga Kakudai", DB Magazine Vol.1, No.313, Kabushiki Kaisha Shoeisha, 01 March, 2004 (01.03.04), pages 176 to 180	1-9

☒ 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 12 December, 2006 (12.12.06)	Date of mailing of the international search report 19 December, 2006 (19.12.06)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2006/320488

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Project KySS, "Jissen Workshop InfoPath+XML ni yoru Office 2003 Renkei Programming", 1st edition, Kabushiki Kaisha Shuwa Shisutemu, 28 June, 2004 (28.06.04), pages 201 to 226	5-9
Y	Satoshi HOSOI, "XML Kanzen Taio no Shogeki Office System Donyu no Scenario", NETWORK WORLD Vol.9, No.4, Kabushiki Kaisha IDG Japan, 01 April, 2004 (01.04.04), pages 196 to 201	7-9

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２００６／３２０４８８

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　G06F3/14(2006.01)i，G06F17/21(2006.01)i

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.　G06F3/14，G06F17/21

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	１９７１－２００６年
日本国実用新案登録公報	１９９６－２００６年
日本国登録実用新案公報	１９９４－２００６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
Y	Technology Overview of Microsoft Office InfoPath 2003, Microsoft Corporation, [online], ２００３．０８．２１, [検索日　２００６．１２．０８], インターネット <URL:http://www.microsoft.com/technet/prodtechnol/office/office2003/plan/iptechov.mspx>	１－９

☑　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	国際調査報告の発送日
１２．１２．２００６	１９．１２．２００６

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）　５Ｅ　３９７９
圓道　浩史
電話番号　０３－３５８１－１１０１　内線　３５２１

Ｃ（続き）．　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
Ｙ	西村めぐみ，Ａｃｃｅｓｓ＆Ｏｆｆｉｃｅ２００３　新機能徹底ガイド　ＩｎｆｏＰａｔｈの柔軟なＸＭＬ編集機能でＡｃｃｅｓｓデータベースの活用範囲が拡大，ＤＢ　Ｍａｇａｚｉｎｅ　第１３巻第１３号，株式会社翔泳社，２００４．０３．０１，ｐ．１７６－１８０	１－９
Ｙ	プロジェクトキッス，実践ワークショップ　ＩｎｆｏＰａｔｈ＋ＸＭＬによるＯｆｆｉｃｅ２００３　連携プログラミング　第1版，株式会社秀和システム，２００４．０６．２８，ｐ．２０１－２２６	５－９
Ｙ	細井智，ＸＭＬ完全対応の衝撃　Ｏｆｆｉｃｅ　Ｓｙｓｔｅｍ導入のシナリオ，ＮＥＴＷＯＲＫ　ＷＯＲＬＤ　Ｖｏｌ．９　Ｎｏ．４，株式会社ＩＤＧジャパン，２００４．０４．０１，ｐ．１９６－２０１	７－９