デジタル・ドキュメント 21－3
（1999．11．26）

オブジェクト指向スクリプト言語Rubyによる XML応用システムの検討

吉田正人 大野邦夫 藤田大作 前 芳久 廣瀬貴幸

INSエンジニアリング株式会社

オブジェクト指向スクリプト言語であるRubyによるXML処理系を用いた応用システムについて報告する。本システムは，既存のシステム群をそのままの形で運用しながら、背後でそれらのデータを収集統合し、ナレッジマネジメントを支援するより広汎なレベルの処理を行うもので，XMLデータ受信部，XML統合参照コアシステム部，サービスアプリケーション部とウエブサーバから構成される。今回は既存システムとしてLotus NotesとOracle DBを取り上げ，これらのデータを背後で収集し，統合参照するシステムを構築した。ウエブサーバ以外の部分は，全てRubyを用いて記述した。

Development of XML Application System with Object - Oriented Script Language Ruby

Masato Yoshida, Kunio Ohno, Daisaku Fujita, Yoshihisa Mae, Takayuki Hirose
INS Engineering Corporation
4 - 31 - 18, Nishi - Gotanda, Shinagawa - ku, Tokyo, 141 - 0031, Japan

Development of XML application system with object - oriented script language Ruby is described. First, the purpose and the architecture of the application system that manages to integrate existing separated application systems by gathering their data in order to support knowledge management are introduced. Second, the design policy and development modules of the system which includes XML data receiving module, XML integrated data management module, and service application module with web server are described. Then in the third, practical prototype system which integrates existing Lotus Notes application and Oracle DB application is dveloped, tested. and evaluated. The system is written in Ruby except its web server module.

1. はじめに

先に，RubyによるXMLの処理系について報告したが[1]，ここではこれを用いた具体的なアプリケーションについて報告する。最近XMLへの関心が高まり，幅広い分野に対する適用が進展している。XMLの使われ方としては，従来のSGMLの延長としての文書オブジェクト系と，集合的なデータを系統的に扱うデータオブジェクト系に大別されるが，現在の主流はデータオブジェクト系である[2]。

特にEC，流通といった分野では，サプライチェーンに代表される受発注を基本とするデータ処理にXMLを適用する試みがなされているが，単なるデータ処理であればXMLを使うメリットは存在しない。XMLを適用するメリットは受発注に伴う一連の業務，すなわち顧客データ，商品データ，在庫，物流，企業会計などの幅広い関連業務を系統的に管理するニーズに対してソリューションを提供し得る点に求められるであろう[3]。

一方，多くの企業では，レガシーの個別のシステムが運用されており，一連の業務を再統合するためにシステムの再構築を行うことは現実的には不可能である。そのために，既存のシステムはそのままの形で生かしながら，背後でそれらのデータを収集し，目的の処理を行うようなシステムが提起されることになる。このようなシステムこそXMLの有効性を証明するものであろう。

このようなシステム構築のコンセプトは以前からも提唱されていた。例えば，OMGのCORBAは，本来，異機種分散環境の統合のための枠組みであり，レガシーシステムの統合もその視野に置かれていた[4]。しかしながら，CORBAで統合するには，種々のシステム要素のCORBAオブジェクト化を計らねばならないのであるが，現実の世界はそのようには進展していない。コンポーネントとなるべきコモンファシリティやビジネスオブジェクトの規格化にOMGは失敗している[5]。

最近注目を集めているEAI（Enterprise Application Integration）もこの分野に属する技術である[6]。CORBAがサーバオブジェクトへのAPIのレベルでの異機種上のアプリケーション統合を図ろうとするのに対し，EAIはデータファイルやRDBデータのレベルで，アプリケーションの統合を図る。

XMLによるデータオブジェクトの統合は，EAIの考え方に近いものである。レガシーシステムにおけるデータをXMLインスタンスとしてにXMLによるタグを付けを一元的に管理するものだからである。ただ従来のEAIは，データファイル形式やデータ照会形式の変更により，アプリケーション統合を行うものであったのに対し，XMLによるデータオブジェクトの統合は，タグを通じてデータの意味的な処理を可能とする点に特徴がある。

2. 基本コンセプトとアーキテクチャ

XMLによるデータオブジェクトの統合システムの基本的な考え方は，各種レガシーシステムのアプリケーションデータをXMLファイル化して取り込み，検索，ソーティング等の処理を通じて情報リソースを活用することにある。RDBのへの（SQLによる）照会結果を整形式(Well-formed）のXMLインスタンスとして取得し，これらを統合して処理することも考えられる。以上の処理のプロセスにナレッジマネジメントを融合させることも考えられる。以上のコンセプトを図式化したものを図1に示す。

システム名称を「KMサーバ」としたのは，将来的にはナレッジマネジメントを支援することを意図していることによる。その第一歩である既存の各種アプリケーションを連携させるデータオブジェクトの統合システムの具体的なアーキテクチャは，その目的により種々の方式が考えられる。

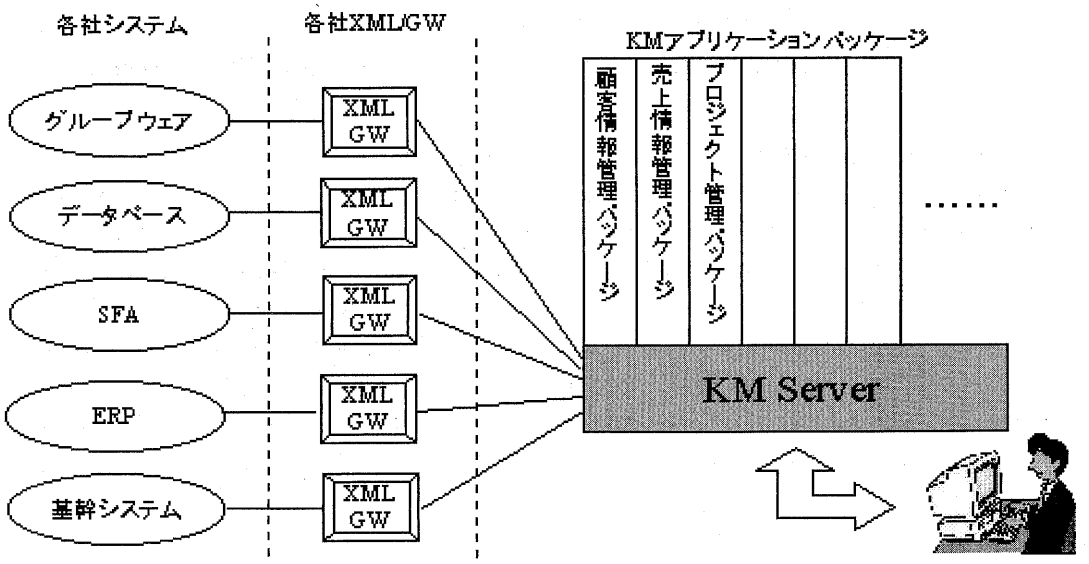

図1 KMサーバのコンセプト

単にファイル形式を統一するだけならば，UNIX系のファイルシステムに変換し，NFSを用いた分散統合システムを効率的に構成することが可能である。ファイルだけでなく，コマンド（オペレーション）レベルを含めたシステムの統合を行う場合には，先に述べたCORBAによるラッピング技術を用いた分散システムを構築することにより目的は達成される。しかし，これらはシステムレベルの技術的なアプローチに過ぎない。

具体的なアプリケーションの統合においては，意味内容レベルの枠組みが必要になる。その場合の一般的な方式は現在のところ存在しないが，個々のアプリケーションにおいて関係する意味情報をキーにしてそれらを相互に対応付けることが基本となるであろう。そのような意味的な処理がナレッジマネジメントの一部となり得ると考えられるのである。そのための具体的な手法としては，各種アプリケーションで広く用いられる意味的に共通なデータに着目する手法が考えられる。例えば，個人，日時，場所，対象などである。これらの情報は，OMGの具体的なビジネス・オブジェクトのインタフェースに近いものである。

個人や日時のデータについては，IETFが "vCard"[7]や "iCalendar"[8]と言った規格を策定し，Microsoftの"Outlook"やLotusの"Organizer"が部分的に採用している。最近は，これらの規格のXML化も提案されているので[9][10]，これらの規格案との連携も考えられるであろう。

3. システム設計

3.1 想定ユーザとニーズ

イントラネットを利用する社内OAシステムでは，異なるシステム上で運用される顧客名簿やスケジュール表の共有などがしばしば話題になる。例えば，複数の部門が関係する特定の顧客への商談状況を系統的に把握するためには，ワークフロー管理システム上の報告書の顧客データとSFA上の最新の顧客データを相互に参照するような必要性が生じるであろう。そこで先ずは，ワークフロー管理としてのロータス・ノーツとOracleデータベース系のシステムの統合を検討することとした。システム構成を図2に示す。

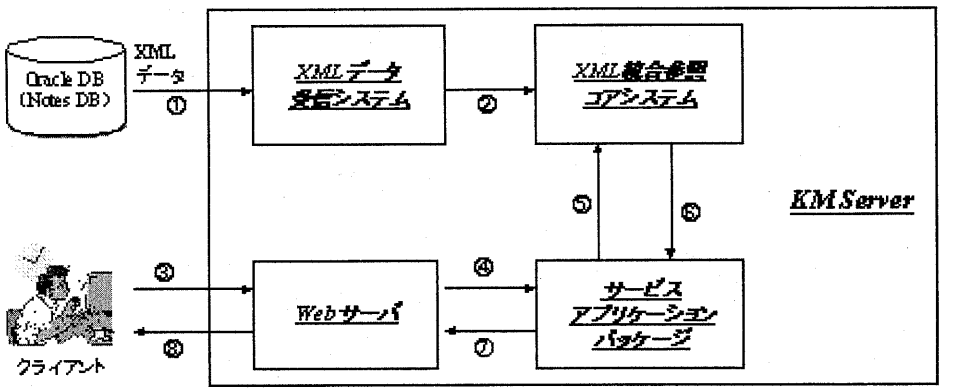

図2 KMサーバの基本システム構成

図から分かる通り本システムは，以下の4つのサブシステムにより構成される。

(1)XMLデータ受信システム：ゲートウエイを通じて既存システムからXMLデータを受信する。

(2)XML統合参照コアシステム：各種XMLデータを統合参照する機能を提供する。

(3)サービス・アプリケーション・パッケージ：コアシステムを参照し，クライアントへサービスを提供するサービスパッケージ。

(4)WEBサーバ：クライアントへの情報入出力サービスを提供する。

今回のプロトタイプにおいては，レガシーシステムをノーツとOracleに設定しているが，システムアーキテクチャ的には，幅広い既存アプリケーションへの対応を可能とするように考えている。

3.2 システムの基本動作

本システムの動作は，下記の以下の①から⑧のステップにより実現される。

① Oracle DB（Notes DB）からXMLデータを受信する

②受信したXMLデータをコアシステムに渡し，データを統合・管理する

③クライアントはHTTPで処理要求をWebサーバへ送信する

④Webサーバは処理要求をアプリケーションパッケージへ送信する

⑤ 処理に必要なデータをコアシステムに要求する

⑥要求のあったデータをサーバアプリケーションに返す

⑦受け取ったデータをアプリケーションに応じて処理を行い，その結果をWebサーバに返す

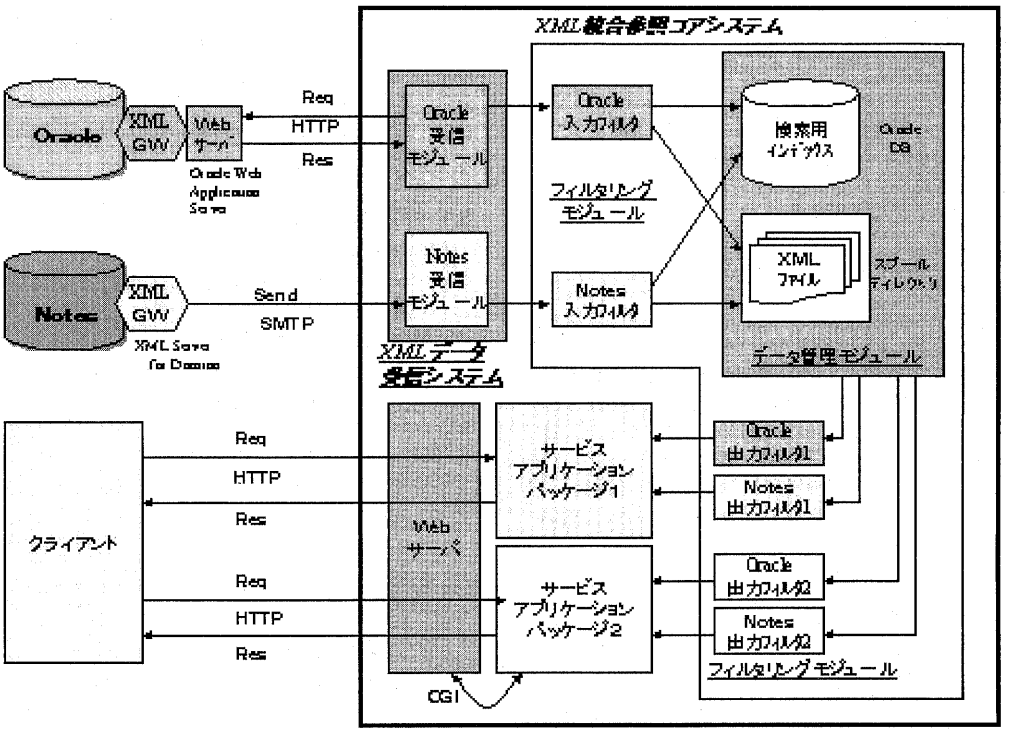

図 3 KMサーバの詳細図

⑧Webサーバは処理結果をクライアントへ返信する

以下に，個々のサブシステムについて解説する。システムの詳細を図3に示す。

3.3 XMLデータ受信サブシステム

レガシーシステムからXML形式でデータを取り出す機能で，前項の①②に相当する。本システムでは，Oracle DBとNotesからXMLデータを取り出すことになるため，Oracle受信モジュールとNotes受信モジュールから構成される。

3.3.1 Oracle受信モジュール

Oracle側のレガシーシステムには，Oracle社製の，「Oracle Web Application Server」を用い，通信機能を持たせることにした。「Oracle Web Application Server」はHTTPプロトコルでアクセスすることによりOracle DBにネイティブに接続することができる。

本モジュールでは，一定時間おきにOracle側システムにHTTPでアクセスを行い，指定したDBの全データを取得する。その取得したデータに対してXMLタグをマッピングする。Oracle側システムからデータの取り出しやXMLダグへのマッピングは，「Oracle Web Application Server」のアプリケーションを用いる。

3.3.2 Notes受信モジュール

Notes側システムには，インフォテリア株式会社開発（Lotus社販売）の，「XML Server for Domino」を用い，通信機能を持たせることにした。「XML Server for Domino」は，指定したNotes DBに新規作成，変更，削除等のイベントが発生するとその内容をXMLデータに変換して，あらかじめ設定している電子メールアドレスにそのXMLデータをSMTPプロトコルで送信する機能を有している。

本モジュールでは，「XML Server for Domino」のXMLデータのメール送信方法のメニュー項目を「メールの添付文書として送信」に設定し，そのメールボックスからメールを受信後，そのメールの添付ファイルからXMLデータを取得する。

3.4 XML統合参照コアシステム

3.4.1 基本動作

レガシーシステムからXMLデータ受信サブシステム経由で取り出したXMLデータを統合・管理し，APIを通してサーバアプリケーションからのアクセスを提供するサブシステムである。このサブシステムは，データ管理モジュールとフィルタリングモジュールから構成される。このシステムの基本動作は下記の通りである。

① コアAPIを通してXMLデータを受信する

②検索キーをフィルタリングして検索用インデックスDBに登録し，XMLデータはファイルとしてディレクトリに保存する

③アプリケーションからコアAPIを通して必要なデータをデータ管理モジュールから呼び出す

④呼び出されたデータはフィルタリングモジュールを通り，必要なデータがアプリケーションに渡される

⑤アプリケーションは受け取ったデータを処理する

以上の動作を図4に示す。

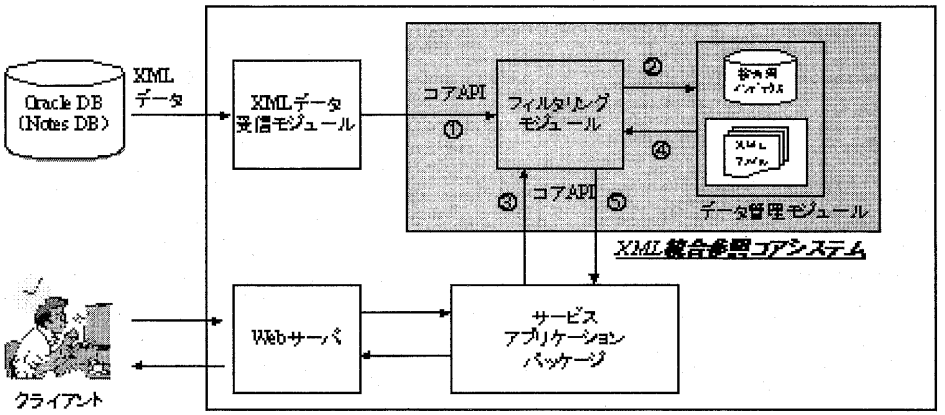

図4 XML統合参照コアシステムの動作

3.4.2 データ管理モジュール

XMLデータ検索用インデックスDBとXMLデータファイルから構成されている。本モジュール機能実現のためには，インデックスDBおよびXMLデータファイルのバックアップ機能が必要である。

(1) 検索用インデックスDB

インデックス用データベースとしてRDBであるOracle8を使用する。フィルタリングモジュールの入力フィルタ（図3にけるOracle入力フィルタ）経由で，XMLデータの検索キーとして定義され，フィルタリングされたデータを，Oracleに保存する。ユーザが検索する対象は，この検索用インデックスDBに保存されているデータのみである。保存されている個々のデータは，そのデータを抽出した元のXMLファイルに関連付けられている。

(2) XMLファイル

XMLデータ受信モジュールで受け取ったOracle，Notes側システムの実際のXMLデータをファイルとして保存する。XMLファイルは，KMシステムでユニークなファイル名が付けられ，特定のスプールディレクトリに保存される。

3.4.3 フィルタリングモジュール

入力フィルタと出力フィルタがあり，必要なXMLデータをフィルタリングして次の処理モジュールに渡す機能を有する。

(1) 入力フィルタ

コアAPIが入力フィルタファイルを呼び出し，そのフィルタファイルに応じて受信したXMLデータから検索キーに設定しているデータを抽出し，データ管理モジュールの検索用インデックスDBに渡す。実際のXMLデータはXMLファイルとしてスプールディレクトリに保存される。

(2) 出力フィルタ

コアAPI経由で出力フィルタファイルが呼び出されると，データ管理モジュールからXMLデータが取り出され，必要なデータをアプリケーションに渡す。

3.4.4 API

XMLデータ受信モジュールやサービス・アプリケーション・パッケージがXML統合検索コアモジュールを通じてXMLファイルを操作するために，データ管理モジュールへアクセスするためのAPIで，下記の3種類に大別される。

(1) データ収集API：レガシーシステムからのデータを収集するためのAPIで，主にXMLデータ受信モジュールが使用する。

(2) データ登録API：データ管理モジュールに格納するためのAPIで，XMLデータ受信モジュールとサービス・アプリケーション・パッケージ双方から使用される。このAPIは，さらに，ファイルAPI，キーAPI，インデックスAPIに分けられる。

(3) データ検索API：サービス・アプリケーション・パッケージからXMLデータを検索するためのAPIである。

3.4.5 サービス・アプリケーション・パッケージ

本サブシステムは，ユーザのニーズに応じてXMLデータの検索，編集，再利用などを行うサービス・アプリケーションである。単一のパッケージとして提供する場合もあれば，ユーザ毎にカスタマイズして提供する場合も考えられる。

サービス・アプリケーション・パッケージの主たる機能は，コアAPI経由で検索・取得したXMLデータの，処理・編集・並びかえ等を行うことである。処理結果の表示形式は，アプリケーションに依存するが，XSL，CSS，HTML，DHTMLなどが用いられることになる。Webサーバへのアクセスインタフェースは，CGIを使用する。

3.4.6 Webサーバ

クライアントはWebサーバ経由で本システムを使用する。クライアントからの要求，およびアプリケーションの処理結果は，HTTPプロトコルで送受信される。

3.4.7 アクセス制御

本システムが想定する代表的な利用者としては，配下の各部署が使用しているシステムを横通しで検索・参照するような戦略企画部門が想定れている。そのような場合は，データへのアクセス権の設定が必須になる。アクセス権の設定のレベルとしては，データ粒度の観点から見るとマクロにはシステム自体へのログインからミクロにはXMLデータオブジェクトの個々のエレメントの内容や属性に依存した制御に至る各種レベルが考えられる。また，アクセス権の設定対象には，管理者，グループ，ユーザのレベルが考えられる。

オフィスでのアプリケーション的観点からのアクセス権としては，経営者，管理者，社員，一般といった利用者のレベル分けが想定されるが，これらの機能は，サービスアプリケーション関数がサービス内容に応じて処理すべき機能である。とは言っても，全文検索のような機能を考慮すると，この機能は必ずしもXMLのタグにのみ依存するわけにはいかないので，(タグを包含したデータを検索する可能性がある）検索対象とするデータの世界を多重化して，XMLデータが異なる複数のコアシステムへのアクセス制御を行うことも考慮の対象とした。

以上の考えに基づき，XML統合検索コアシステムについては，図5に示すように，多重構成とすることが可能なアーキテクチャとした。システムの具体的な動作は，以下のようになる。

①コアAPIを通してそれぞれのフィルタリングモジュールが受信モジュール経由でXMLデータを受信する

②それぞれのフィルタファイルの設定に基づき，検索キーをフィルタリングして検索用インデックスDBに登録し，XMLデータをファイルとしてディレクトリに保存する

③サービス・アプリケーションは，コアAPIを通して，クライアントのアクセス権に応じたコアシステムから必要なデータを呼び出す

④呼び出されたデータは，それぞれのフィルタリングモジュールを通り，アプリケーションに渡される

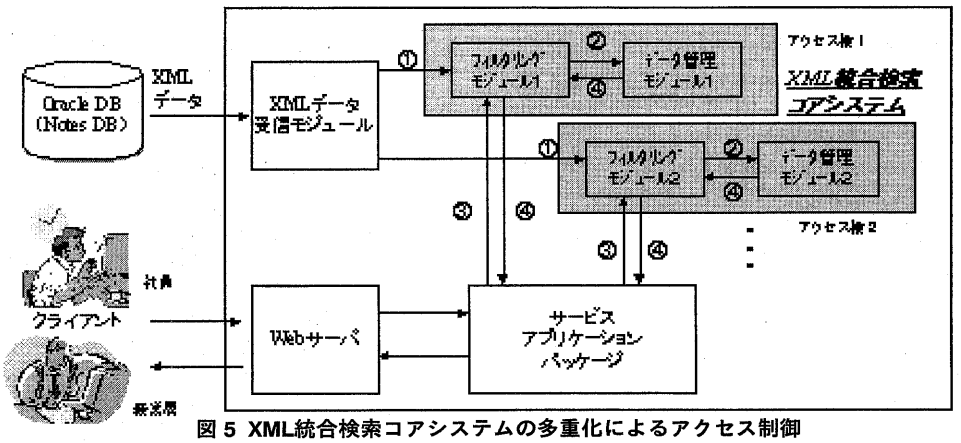

図5 XML統合検索コアシステムの多重化によるアクセス制御

4. プロトタイプシステムの構築

4.1 システム実装方針

以上のコンセプトに基づき具体的な構成の仕様が固められ，以下の方針に基づき開発に取りかかった。

(1)XML形式で統合・管理対象となる当面のレガシーシステム環境をOracle　R8.0.5とNotes R5.0に絞る。

(2)クライアントが使用するブラウザをXML対応ブラウザであるIE5とする。

(3)開発言語として，オブジェクト指向スクリプト言語であるRuby1.4を使用する。

(4)XMLの処理には，James Clark 氏のExpat をRuby に組み込んだXMLパーサを使用する[1]。

4.2 プラットフォーム環境

4.2.1 Oracle側のシステム環境

以下の環境のレガシーシステムを対象とする。
(1)プラットフォーム：PC/AT互換機

(2)OS：Windows NT Server 4.0 SP5

(3)Oracle環境：Oracle8 Workgroup Server R8.0.5 for Windows NT

(4)ゲートウエイ：Oracle Web Application Server 3.0.1 Advanced for Windows NT

4.2.2 Notes側のシステム環境

以下の環境のレガシーシステムを対象とする。
(1)プラットフォーム：PC/AT互換機

(2)OS：Windows NT Server 4.0 SP5

(3)Notes環境：Lotus Notes R5 for Windows NT

(4)ゲートウエイ：Infoteria XML Server for Domino 1.0

4.2.3 KMサーバ側のシステム環境

以下の環境を対象とする。

(1)プラットフォーム：PC/AT互換機

(2)OS：TURBO LINUX 4.0 日本語版

(3)コアシステム：Oracle8 Workgroup Server R8.0.5 for Linux

(4)Webサーバ：Apache 1.3.6

4.3 モジュール実装

システムについては，既に図3で紹介しているが，ここではさらにその詳細の実装を説明する。

4.3.1 XMLデータ受信モジュール

Oracle受信モジュールとNotes受信モジュールから構成される。この部分については，対応する通信モジュール（Oracle Web Application ServerとInfoteria XML Server for Domino）が存在するので，それに合わせたプログラム構成となる。基本的にはRubyで記述し，通信部分については，対応するモジュールの関数群を用いている。

プログラム構成としては，個別のレガシーシステム毎にサブモジュールを作成することとなるが，オブジェクト指向の特徴を生かし，サブモジュールの再利用性にも注意を払っている。

4.3.2 XML統合参照コアシステム

このサブシステムは，フィルタリングモジュールとデータ管理モジュールに大別される。データ管理モジュールは，データベース管理とファイル管理の基本機能をサポートするだけなので，プログラム部分は少ないが，プログラム自体はRubyで行っている。

フィルタリングモジュールは，ExpatパーサとRubyで記述されている。フィルタへのAPIは，SAXと同様なイベント処理によるものであるが，具体的には先の報告を参照して頂きたい[1]。

4.3.3 サービス・アプリケーション

サービスアプリケーションは，ユーザ対応の具体的なアプリケーションを記述するモジュールであり，将来的にはパッケージとして個別製品化する位置づけのものである。具体的には，CGIからユーザ要求を受け付け，検索用インデックスを参照して，検索，編集等の操作を加えて，結果をWebブラウザ経由でクライアントに返す処理を行う。このプログラムもRubyを用いて記述されている。

4.3.4 Webサーバ

クライアントとのやり取りを行うためのWebサーバとしては，Apacheをそのまま用いている。前項のサービスアプリケーションは，このサーバのCGIクライアントとして位置付けられる。

4.4 テストとデモンストレーション

一通り実装を完了させてから，OracleとNoteのシステムに具体的に接続し，テストを行った。接続に関しては，以前から試行的に検討・評価していたので問題は生じなかったが，WebブラウザのGUIを中心としてシステムの使い勝手に関し種々の議論が発生した。

GUIを含むユーザインタフェースの変更は，場合によっては，サービスアプリケーションのロジックやフィルタリングのロジックにまで，手を入れることになるが，Ruby言語は，インタプリータベースなのでプログラムの修正やデバグが容易であり，迅速な対応が可能であった。このシステムをC++やJavaで実装していたら，とても短期間では実装出来なかったであろう。

5. 考察

5.1 Rubyの効果

本システムでは，実装のためのプログラム言語として，Rubyを用いたが，先ずその有効性と問題点について考察する。前項で述べた通り，Rubyは，ユーザ要求の追加・変更や，システム環境の変更に伴う修正やデバグにたいしては迅速な対応が可能であり，本システムのようなプロトタイピング的なシステム構築に関しては適切な環境と言える。

本システムの構築に，CやC++を用いると，プログラム修正の都度コンパイル，リンクを繰り返さねばならず，メモリ管理にも気を使わねばならないので極めて煩雑であり，かなりの稼動を覚悟せねばならないであろう。Javaを用いる場合には，メモリ管理は不要であるが，コンパイル，リンクの手順はC，C++と同様に要求される。

PerlやPythonのようなスクリプト言語は，Rubyと同様な特徴を有するが，前者はブジェクト指向的なメリットに欠け，後者は日本語処理機能に欠陥がある。

Ruby自体は新しい言語なので，コミュニティが小さく，ドキョメント類も少ないと言った問題点もあるが，このような用途のための言語としては，大きな可能性を秘めていると言えるであろう。Rubyに関する解説本も最近出

版されたので[11]，今後の実用システムへ向けた発展と普及を期待したい。

5.2 ナレッジマネジメント

ナレッジマネジメントは明確な定義は困難な用語である[12]。明確化が困難であるが故に「暗黙知」というような概念が生み出され，それを明確化した「形式知」と対比され，相互の変換プロセスのダイナミズムに企業活動の源泉を求めるのが最近の動きであろう[13]。

本システムは，ナレッジマネジメントに関しては，不十分と言わざるを得ないのであるが，種々のプラットフォーム上のレガシーシステムにける意味的なデータをキーにして統合，編集，再利用などを行う枠組みは出来上がっており，ナレッジマネジメントの支援程度は可能なシステムと言えるであろう。

5.3 エージェントシステムへの展開

本システムのように個々のアプリケーションにおいて関係する意味情報をキーにして相互を対応付けるより一般的なモデルの構築が検討されるべきであろう。そのための具体的な手法としては，各種アプリケーションで用いられる「個人」，「日時」，「場所」といった共通のデータに着目する必要があることは先に述べた通りであ，個人や日時情報についてはIETFでXMLのDTDによる標準化も検討されていることも述べた。「場所」についてもデータベース振興センタがGXMLとして標準化することを検討しており[14]，これらのメタデータが規格化されると，本システムのようなデータの扱いは容易になる。

レガシーシステムのデータ検索のより一般的な枠組みは，今後はエージェント技術に求められるであろう。FIPAは，エージェント技術の中心的な課題として，ACL（AgentCommunicationLanguage）を定めている。これは，エージェント同志が通信するための言語であるが，モデルをスピーチアクト理論に置き，個々のパーフォマティブに対応したオントロジとコンテンツでエージェント間の通信を実現する。このオントロジとコンテンツに，XMLのDTD（スキーマ）とXMLインスタンスを適用する試みが報告されている[15][16]。本システムにおけるKMサーバの情報探索メカニズムも将来的には，エージェント技術を採り入れてゆくことになるかもしれない。ナレッジマネジメントとエージェント技術の融合も今後の課題として検討する必要がある。

5.4 N層クライアント・サーバ通信モデル

本システムは，Oracle側のシステムに対しては，HTTPで，Notes側のシステムに対しては，SMTPで通信する。このような方式は，単純なクライント・サーバ・システムではモデル化が困難であるが，3層方式のデータ層にレガシーシステムを包含するモデルや，ミドルウエア層，データ層を多様化するN層方式などで扱われるようになってきた[17]。エージェント通信でもこのような多層の通信モデルが提起されている[16]。

このような通信モデルが，標準化される可能性もあるが，そのためにはエージェント通信のモデル化が必要であろう。FIPAのACLはその有力なモデルになり得るかもしれないが，オントロジとしてのXMLのDTDやスキーマが作成されることが前提になるであろう。

6. おわりに

以上，Rubyによる具体的な応用システムについて述べた。プロトタイプとしての評価に基づき，今後の実用システムとしての開発を進める予定であるが，それをRubyで構築するか，より実用的なJavaやC++で開発するかは，具体的なユーザとのコンサルティングに基づいて決めたいと考えている。

最後に本システムの開発を進めるにあたり，叱咤激励して頂いた，大嶋事業開拓室長に感謝します。また経営的立場からの助言と利用者としての立場からのコメントを頂いた高橋社長に御礼申し上げます。

文献および参照情報

[1]吉田，大野;“オブジェクト指向スクリプト言語RubyによるXMLの処理”, 情報処理学会デジタルドキュメント研究会研究報告，DD18 - 3, (1999.5)
[2]大野;“XML応用の最近の動向”,情報処理学会デジタルドキュメント研究会研究報告，DD17 - 7, (1999.3)
[3] Patricia O' Sullivan; “XML in Information Technology Supply Chain - Rossettanet”, Proc. on XML'98 EC Track (1998.11)
[4] Richard Mark Doley, et. al.; “Object Management Architecture Guide 1.0”, OMG TC Document 90 - 9 - 1
[5] 河込, 中村, 大野, 飯島; “分散オブジェクトコンピューティング”, ISBN4 - 320 - 02775 - 2 C3341，共立出版, (1999)
[6] 日経コンピュータ; “EAIツールが続々登場”, 日経コンピュータ, 1999.9.13, pp. 32 - 34, (1999)
[7] F. Dawson; “vCard MIME Directory Profile”, IETF RFC 2426
[8] F. Dawson; “Internet Calendaring and Scheduling Core Object Specification (iCalendar)”, IETF RFC 2445, (1998.11)
[9] http://www.ietf.org/internet - drafts/draft - dawson - vcard - xml - dtd - 03.txt
[10] http://www.imc.org/draft - dawson - ical - xml - dtd
[11] まつもとゆきひろ, 石塚圭樹; “オブジェクト指向スクリプト言語 Ruby”, ISBN4 - 7561 - 3254 - 5 C3004, アスキー出版局, (1999)
[12] 大野; “メガ競争時代におけるデジタルドキュメントの役割”, 情報処理学会デジタルドキュメント研究会研究報告，DD19 - 1, (1999.7)
[13] 野中郁次郎；”組織的知識創造の新展開”, ダイヤモンド・ハーバード・ビジネス, Vol.24, No. 5, 通巻139号, pp.38 - 48, (1999.9)
[14] http://gisclh.dpc.or.jp/gxml/
[15] C. Vermeulen, B. Bauwens ”Software Agents using XML for Telecom Service Modelling: A Practical Experience” , Conference Proc. on SGML/XML Europe'98, pp.253 - 262
[16] Bart Bauwens; “XML - based agent communication: VPN Provisioning as a case study”, Proc. on XML '99, Core Technologies Track (1999)
[17] J. Allard; “Evolving Web Application with Windows DNA & XML”, Proc. on XML'98 at Chicago, Keynote Speech, (1998.11)