事例

# 分散オブジェクト技術 Java／CORBAによる 全社人材情報システムの構築

高杉 秀樹　竹内　充　大川　勉
三菱電機（株）情報システム技術センター
岩本　仁　石田 征久
三菱電機システムウェア（株）

本事例報告で紹介するのは，当社資材部門での効果的な人材運用を支援するために開発した人材情報システムである．全社資材部員や各部門の長だけでなく，資材部門に在籍したことがあり経歴を保管しておきたい社員の職務履歴を記録し，その検索機能を実現することにより，部員の経験と能力とを最大限に活かすような人事運営や人材交流を図るための情報を全社に分散する資材部門に提供する．

このシステムは，全国に散らばった全社資材部員が利用すること，人材情報は各事業所が管理していることから，クライアントにインストールする手間が省けるWEBブラウザで利用でき，より使いやすいユーザインタフェースを提供すること，また分散している人材情報を容易に扱え，かつ人材情報のありかを柔軟に変更できることを狙って，Java[1]/CORBA[2]といった技術を，全社システムに適用した．

しかし，実際には，システムのフロントエンドとして開発したJavaアプレットの動作時にメモリリークが発生したこと，Javaアプレットは各事業所のネットワーク回線速度や負荷を考慮してできる限り小さくする工夫をしたがダウンロードに時間がかかる場合があり実用上問題が残ることの2点を理由にJavaアプレットによるフロントエンドは断念し，Javaアプリケーションとして実現することになった．

完成したシステムは当初の予定していた姿とは異なる形となったが，システム自体は資材部門向けの人材情報システムにとどまらず，同様に全社に展開している人事部門や情報システム部門向けのシステムとしても流用可能である．したがって，Java/CORBAといった技術を適用したことについては一応の成果があったと考えている．

## ■システムの概要

### －サービス機能－

資材部門は全社に展開している部門である．その職務上，1カ所にまとめることができず，各事業所に部員が常駐し職務を遂行している．これまでは，各部署の適材適所に，必要な人員を効果的にタイムリーに配置するのに手間がかかった．そこで，全部員の担当職務と能力を資材部門のどの部署からも検索できる情報サービスが望まれるようになった．

本人材情報システムが提供するサービスは，必要とされる能力を持った部員を必要とする事業所に配置したり，特定の業務知識を持っている部員を探したりできるように，資材部員の入社から現在までの担当職務を人材情報データベースに記録し，検索するためのサービスである．

人材情報サービスは次の2種類のユーザを対象にしている．

①人員配置を改善することにより資材部門全体の作業効率を上げたい管理者．

②自分の抱えている問題が解決できそうな職務経験を持つ部員を探したい一般部員．

この2種類のユーザの要望に応えるべく，本人材情報サービスは2種類の検索サービスを用意している．

①管理者向けには，非公開情報を検

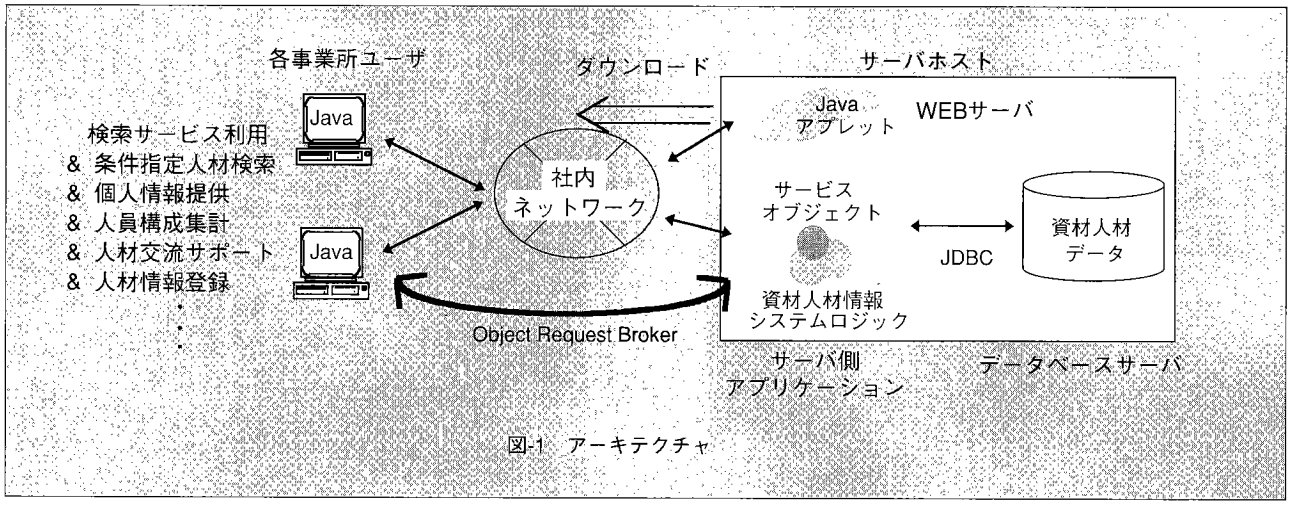


図-1 アーキテクチャ

索する機能を提供する．非公開情報とは，資材部員の職務履歴，学歴，現在保有資格などである．また，業務に対する経験度など，職務履歴から算出される情報も含まれる．

この非公開情報検索機能が，本人材情報サービスで中心をなすサービスである．

②一般部員向けには，公開情報を検索する機能を提供する．公開情報とは，資材部員の誰にでも，必要があれば設計部門，製造部門にも公開できる情報である．たとえば，部員の得意分野，所属部署，内線番号，メールアドレスがこれに当たる．得意分野をキーとして目的の人材を探すことができる．

## －システムアーキテクチャ－

データベースを中心として，本人材情報システムはJava/CORBAを使った3層式クライアントサーバシステムとして実現した（**図-1**）．第3層目のデータベースサーバの他に，データベースを検索し更新する機能をCORBAの分散オブジェクトとして実現した第2層目のサーバ側アプリケーション，そして第1層目のユーザインタフェースを持ったフロントエンドが存在する．データベースサーバとサーバ側アプリケーションとは，WEBサーバと同一ホスト上で動いている．このサーバホストを使った情報サービスを利用するためのGUIを実装したフロントエンドがユーザのパソコン上で動いている．フロントエンドとサービスオブジェクトとはORB（Object Request Broker）を通して通信する．

本人材情報システムでは，サービスオブジェクトもそのフロントエンドもJavaを用いて開発した．期待したメリットは，次の3点である．

①インストールの負担を省く：

全社システムであるため，インストールの負担を省くことができるようにWEBブラウザ上のアプリケーションとして構築できる．

②WEBブラウザ上でよりよいユーザインタフェースを提供する：

HTMLで記述したユーザインタフェースより高度なインタフェースによって検索サービスをより有効な形で提供したい．

③全社にサービスを提供するためのサービスオブジェクトとフロントエンドの開発／運用コストダウン：

IDL（Interface Definition Language）から生成されたJavaによるスタブとスケルトンは，データ型変換とメモリ管理がC++言語によるスタブとスケルトンに比べて，プログラマには理解が容易である．

さらに，Javaには3層式クライアントサーバシステムを開発するのに必要な機能をひと通り備えたクラスライブラリがある．特に，HTTPを使った通信，CORBA，マルチスレッドなど分散オブジェクトの実装に必要な機能がライブラリに含まれているのがありがたい．

上記のことから，開発時だけでなく，運用時のトラブル対処や機能追加／変更においても効果が期待できる．

また，CORBAを採用した理由は，サーバ上のシステムロジックやデータベースが，たとえば複数の事業所に分散したり，関係会社や海外部門との連携をとったりなどサーバアーキテクチャを柔軟に変更できる点を考慮したからである．

## －データベース－

システムで重要なのは，サービスの素が格納されているデータベース（人材情報）である．人事部門が管理する人事データベースと情報システム部門が管理する住所録データベース（ディレクトリサービス）の，2つの既存の大規模データベースから抽出したデータに，資材部員の職能記録を追加して，資材部門専用の人材情報データベースを構築した．特に，元となる2つのデータベースには，本人材情報システムに求められた全部員の職務の歴史的記録が欠けていたために，職務履歴を収めたテーブルを新たに加えた．このテーブルには，人事異動や職務変更が起きるたびに「いつから誰がどこで何を扱ったか」を記録した職務履歴情報が追加される．

データベースのスキーマは，意思決定支援システムの定石通り，スタ

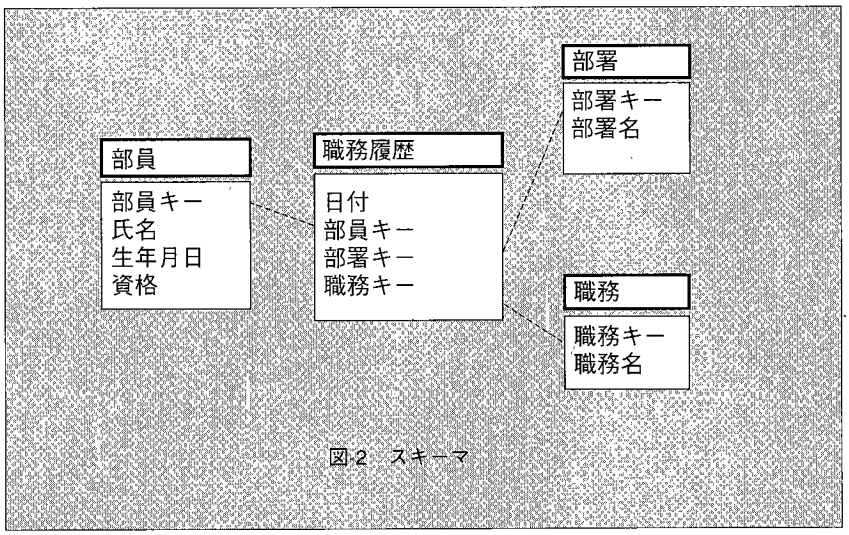

図-2　スキーマ

ースキーマ[4]である（図-2）．スタースキーマでは，特定の情報（たとえば，資材部員）をそのキーとなる項目で1つのテーブルにまとめることができるので，テーブルで表されたデータ構造が理解しやすい．中心となる職務履歴テーブルには，「いつ誰がどこで何をしたか」というレコードが並んでいる．そのテーブルが指すのが，「誰が（部員）」，「どこで（部署）」，「何を（職務）」の3つのテーブルである．

職務履歴テーブルは，本人材情報システム開発のために新たに作られたものであり，人事異動のたびに，人材情報システムの更新機能を使って，全社資材部門の更新担当者によって更新される．

周りの3つのテーブルは，人事部門と情報システム部門が管理する既存のデータベースから抽出されたデータを元に作られている．したがって，元となるデータベースが更新されるたびに，これらのテーブルも更新される．更新のあったデータはファイルにまとめられ，人材情報システムの一部であるバッチプログラムにより，このファイルの内容に合わせて人材情報データベースが更新される．

## －ネットワーク－

新幹線の車両がそのままでは在来線を走れないのに似て，Java/CORBAのシステムは，そのままでは社内ネットワークでは動かないことが多い．フロントエンドとサービスオブジェクトとを繋ぐ回線に細いところがあったり，セキュリティポリシーとしてCORBAの通信プロトコルが認められなかったり，さまざまな問題が起きる．

ここで2通りの対処法が考えられる．第1は，新しいシステムに合わせて社内ネットワークを改造する方法である．これにはコストと時間がかかる上に，セキュリティホールを作り込んでしまう恐れがある．第2は，システムを既存のネットワークに合わせることである．この場合は，コストも時間も第1の方法に比べてかからなくてすむが，システムの本来の性能を損なう恐れがある．

実際のシステムの開発では，コストと時間が一番大きな制約である．本人材情報システムの開発も例外ではない．そこで，第2の方法をとった上で，システムの性能劣化をなるべく避ける工夫をした．

①サービスの起動：

社内ネットワークでは，CORBAネーミングサービスが使えない事業所があるため，フロントエンドが人材情報サービスの最初のオブジェクトと接続するために，WEBサーバを使うことにした．つまり，人材情報サービスオブジェクトの文字列化したIOR（Interoperable Object References）をWEBサーバの管理するファイルに保存しておき，そのファイルのURLをフロントエンドに伝えるのである．フロントエンドは，このURLが示すファイルの内容をWEBサーバ経由で入手し，オブジェクトリファレンスを再生するのである．

②フロントエンドとサービスオブジェクトとの通信プロトコル：

CORBAで標準化されたプロトコルであるGIOP/IIOPが通る事業所では，GIOP/IIOPを使う．通らない事業所では，HTTPトンネリングを使うことにした．HTTPトンネリングとは，HTTPのメッセージの中に埋め込んだGIOP/IIOPのメッセージを分散オブジェクトと送受信する方式である．HTTPトンネリングを実現したCORBA製品はいくつかあり，本システムでは，フロントエンドにもサーバ側アプリケーションにもVisiBroker[3]を共通して使うことにした．とはいっても，GIOP/IIOPが通る事業所では，Java実行環境に付属したORBを使った方が，既存のORB製品をインストールする手間が省ける．そこで，フロントエンドがJava実行環境付属ORBでも既存ORB製品でも動けるように，ポータブルスタブを使うことにした．ポータブルスタブは，CORBAに準拠した命令だけを使ってIDLコンパイラが生成するスタブだが，特定のORBに固有の命令を使わない分，スタブが大きくなったり処理速度が遅くなったりする弱点がある．

③回線速度：

社内ネットワークで一番回線が細い事業所でもある程度の応答性能が期待できるように，まず，フロントエンドを構成するアプレットを小さくした．アプレットは最大でも80キロバイトである．さらに，HTTPトンネリングを使った通信があまり遅くならないように，フロントエンドとサービスオブジェクトとのメッセージのやりとりの回数を押さえた．既存のORB製品でHTTPトンネリングを使った場合，サービスオブジェクトの操作を1つ呼び出すと，複数の

メッセージが送られるので，通信の頻度を抑えればかなり操作性が増すのである．

④フロントエンド側のHTTPプロキシサーバ：

HTTPトンネリングを使った場合，フロントエンドとサービスオブジェクトとの間にはHTTPプロキシサーバが立ちふさがる．そして，プロキシサーバに対して行うフロントエンドの設定は，一様ではない．このため手間はかかるが，ユーザの各パソコンにHTTPプロキシサーバの環境を設定した．

## ■サービスオブジェクトの設計と実装

トランザクション処理を生業とするサーバとは違い，人材情報システムのサーバ側アプリケーションは，検索結果の加工がその主な機能である．事実，本人材情報システムのデータベースの中心となるテーブルは，「いつどこで誰が何をしたか」というレコードが並んでいるだけである．ここから，ユーザが欲する情報を作り出すのがサーバ側アプリケーションの主な機能となる．たとえば，特定の原材料を扱った経験の一番長い部員を見つけるために，SQLとJavaとを組み合わせた複雑な集計作業をサーバ側アプリケーションは行う．

### －インタフェースの設計上の制約－

本人材情報システムを既存のネットワークで動かすためには，そのサーバ側アプリケーションのインタフェース設計で，応答性能を下げないようにするため，守らなくてはならない制約があった．

①スタブを小さくする：

既存のネットワークには回線の細い個所があるので，システムの応答性能を上げるには，アプレットとともにユーザのパソコンにダウンロードされるスタブは小さくしなくてはならない．このため，インタフェース設計を工夫した．ここでスタブとは，サーバ側のサービスオブジェクトに要求を送るための通信用メッセージを生成し，戻ってきた返答をJavaのデータ構造に変換するための手続を定義したJavaクラスである．

第1に，人材情報システムのサーバ側アプリケーションのインタフェースは，IDLで定義する仕様を少なくし，Java言語で実装する部分を増やした．これは，IDLソースから生成されたスタブよりも，同じ仕様をJavaで作成したプログラムの方が小さくなるからである．

第2に，1つのサービスオブジェクトに必要最小限のインタフェースを定義した．不要な操作を定義したスタブは不要に大きくなってしまう．これを避けるために，この手法は有効である．これによって必要とする操作だけを定義したスタブをクライアントがダウンロードできるようにできる．

②メッセージの送受信回数を減らす：

第1に，検索結果は一度のメッセージの送信で得られるように，1つの構造体にまとめてクライアントに渡すようになっている．検索結果を収めたサービスオブジェクトをサーバのメモリ内部に生成し，そのリファレンスをクライアントに返すようなことは，検索結果が大量にならないかぎり行わない．

第2に，インタフェースには手順的結合[5]された操作がある．手順的結合とは，「あれをして，これをして，最後にこれをする」という意味を持つ操作である．本来，3つの独立した操作として定義すべきなのだが，本システムではこれらの操作が必ず一定の順に実行されるので，1つの操作にまとめたのである．

当然，本システムだけで使える操作なのでソフトウェアの再利用性を欠くが，確かにサービスオブジェクトとの通信頻度は減少するのである．

### －クライアントとのインタフェース－

本人材情報サーバのインタフェースは，ファクトリパターン[6]に従った3つのインタフェースで構成されている（図-3）．それぞれのインタフェースに対応してサービスオブジェクトが実装されている．

①コントロール（Control）：

クライアントに対するセッションを管理するオブジェクトである．セッションオブジェクトといった他のオブジェクトを生成する機能を有するファクトリオブジェクトでもある．このインタフェースのIORがクライアントに公開されている．クライアントが人材情報サービスを利用するには，コントロールにログイン操作を送り，セッションオブジェクトを

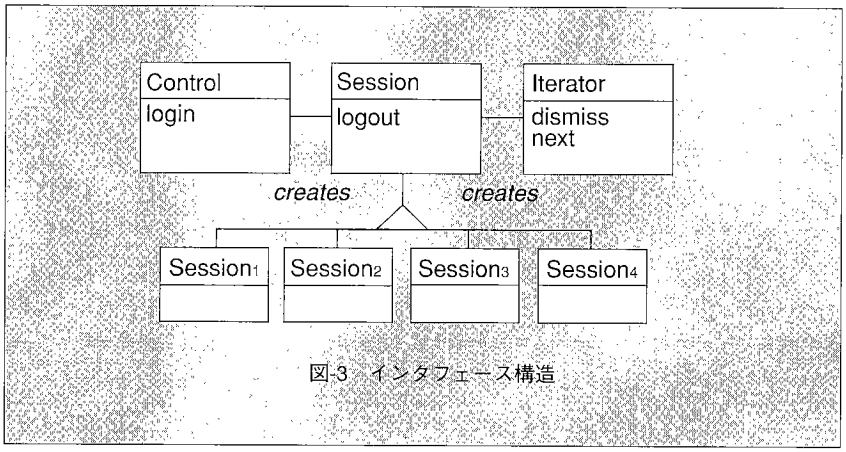

図-3　インタフェース構造

```
<APPLET CODE = "foobar.class" >
<PARAM NAME = "session" VALUE ="$?">
</APPLET>
```

図-4　テンプレートファイルの例

生成する．

②セッション（Session）：

ログイン操作によりサーバのメモリ内部に生成され，ログアウト操作により削除される．このインタフェースは4つのインタフェースに分かれている．それぞれのインタフェースで，操作の流れの局面で必要な操作がまとめて定義してある．スタブを小さくするためである．

③イタレータ（Iterator）：

大量な検索結果を閲覧するための操作を定義したオブジェクトである．セッションが終了したときに削除される．デザインパターンの「イタレータ」に従って定義されたインタフェースである．

### －実装－

サーバ側アプリケーションは，サービスオブジェクトを実装し，HP/UX10.20（サーバ（ME/S）OS）の上でデーモンプロセスとして動作するプログラムである．ほとんどすべての機能がJava1.1で書いてあるが，一部の機能（シグナルの受信，ログデーモンへの出力，デーモンプロセスとしての起動）が，JNI（Java Native Interface）[7]に従ってC言語で書いてある．

サーバ側アプリケーションが，最新版のJava2ではなくJava 1.1で書いてあるのは，次の理由による．

①サーバの実行環境であるHP/UX 10.20にJava 2が載っていなかったため．

②GUIを持たないサーバ開発用にはJava1.1のクラスライブラリで十分だったため．

③HTTPトンネリングを実現したプロキシサーバなど，既存ORB製品に特有な機能がサーバ側アプリケーションには必要なので，逆にJava 2に装備されたORBが邪魔になるため．

HTTPトンネリングを行うためORBには特定のORB製品を使っているが，サーバ側アプリケーションのソースプログラムはできるだけ他のCORBA製品にも移植できるように気を使っている．たとえば，起動モードはCORBA製品によって方式が異なることがあるので，サーバ側アプリケーションは最も一般的な永続モードで起動するように実装されている．クライアントから要求が届いたときではなく，サービスを始める前にサーバを起動しておくのである．実際は，サーバの実行環境がUNIXなので，他のデーモンプロセスと同じように，オペレーティングシステムを起動した際に，サーバも自動的に立ち上がるように作られている．また，割り込みを受け取って終了できるように，シグナルハンドラがC言語で実装してある．

## ■フロントエンドの設計と実装

本人材情報システムには，2種類のフロントエンドがある．1つは，Javaアプレットとして実装されたフロントエンドであり，もう1つは，ブラウザのヘルパアプリケーション[8]としてJavaで書かれたフロントエンドである．もともと，フロントエンドは，サービスごとの複数のJavaアプレットとして設計したが，ブラウザを操作している最中にアプレットを作ったり壊したりしていると，ブラウザが動かなくなってしまった．具体的にはいくつかのサービスを利用しようとJavaアプレットを切り替えることを繰り返しているとメモリが次々と消費されていってしまったのだ．Java Developer Connectionのバグパレードでも Javaアプレットの実行環境がメモリリークを起こすことが公開されていた．我々には実行環境そのものを修正することができない以上，この不具合を避けるには，アプレットではなく単独のアプリケーションとしてフロントエンドを実装する方式も実現した．

### －アプレット版－

細い回線でもWEBサーバからダウンロードするのに時間がかからないように，フロントエンドは，次のような機能を持った5つのJavaアプレットに分かれている．Javaアプレットは大きなものでも80キロバイト，小さなものは30キロバイトしかない．最初のアプレットがセッションを開始する．残りのアプレットは，操作の流れに合わせてユーザのパソコンにダウンロードされる．ログインしてからログアウトするまでの間，アプレットは1つのセッションを共有する．

①ログインと操作選択

②一般部員向け情報の検索

③管理者向け情報の検索

④情報の更新

⑤データベースの保守

異なるページに貼りついたアプレットどうしで1つのセッションを共有するには，アプレットから参照できる場所にセッションオブジェクトのIORを保存しておかなくてはならない．アプレットだけの機能ではこれは実現できない．同じJVM（Java Virtual Machine）で動いていても，staticな領域の内容はアプレットごとに初期化されることが多いので，事実上アプレット側のメモリ空間にはIORを保存することができないのである．メモリ以外のファイルやクリップボードなどの資源も，セキュリティ上の制限でアプレットには使えない．

本人材情報システムでは，この問題を解決するために，アプレットを

起動するHTMLファイルを生成する機能をサーバ側アプリケーションに追加した．セッションオブジェクトのインタフェースには次の操作createHTMLが定義してある．第1引数nameにはファイル名，第2引数argsには文字列のシーケンスを指定する．この操作は，次のアプレットを起動するアプレットタグを含んだHTMLファイルを生成してそのURLを戻り値として返す．

```
string createHTML
(in string name, in SeqString args);
```

次に起動されるアプレットに，この操作を使って，セッションオブジェクトのIORを伝える仕組みについて述べる．生成されるHTMLファイルの元となるテンプレートファイルを，サーバ側アプリケーションが管理するディレクトリに用意しておく（図-4）．セッションオブジェクトのIORは，HTMLを生成する際に，文字列化してcreateHTML操作の第2引数でサーバ側アプリケーションに渡しておけば，テンプレートでマクロ変数がこのIORと置き換わったHTMLファイルをサーバ側アプリケーションが生成し，次のアプレットに先のIORが伝わるのである．

### ーヘルパアプリケーション版ー

ヘルパアプリケーションとは，ダウンロードされたファイルを処理するために，MIMEデータ型に合わせて，ブラウザから起動されるアプリケーションのことである．アプレットではないフロンドエンドは，必ずしもヘルパアプリケーションでなくてもよかったのだが，アプレット版と同様に，WEBページを操作して起動されるようなプログラムとした．WEBブラウザ上のリンクをクリックすると，Javaアプリケーションが起動される．

ヘルパアプリケーションとして実装されたフロントエンドとMIMEで対応づけられたファイルは，データ形式としてはテキストファイルであるが，内容は，サーバ側アプリケーションとフロントエンドとが通信するのに必要なパラメータがJavaのプロパティとして定義してある．このようなパラメータは，アプレットではアプレットタグのパラメータに指定されるものである．このファイルの型をMIMEとして登録して，フロントエンドと対応づけておく．

通信用パラメータを収めたファイルを指したWEBページ上のリンクをブラウザでユーザが選択すると，ブラウザはこのファイルをサーバホストから作業用ディレクトリにダウンロードし，そのファイルを処理するためにフロントエンドを起動する．

## ■評価

Java/CORBAによって全社的に利用する人材情報システムを構築した．ここでは，その狙いと期待した効果に対して，実際どうだったかについて評価する．

①インストールの負担を省く：

当初Javaアプレットとしてフロントエンドを開発したが，Java実行環境のメモリリーク，Javaアプレットのダウンロード時間を短くする工夫が及ばなかったこともあって断念した．さまざまなネットワーク環境のもとで実用的なフロントエンドをJavaアプレットで構築するためは，さらなる工夫が必要である．

②WEBブラウザ上でよりよいユーザインタフェースを提供する：

HTMLで記述されたものよりは検索結果を閲覧するといった目的に合ったフロントエンドが構築できた．フロントエンド側にデータベースのデータを保持しておけることで，さまざまな角度から人材情報を検討するために発生するサーバへの要求を減らすことができるといった効果もあった．

③全社にサービスを提供するためのサービスオブジェクトとフロントエンドの開発／運用コストダウン：

システムを利用する際の認証，アクセスコントロール，セッション管理などを行うサーバ側アプリケーションが手軽に開発できた．

運用コストダウンという点では，トラブル対処については現在トラブルが発生していないため不明であるが，機能追加／変更については現在実施中であるのでしばらく後に評価可能である．

## ■今後の展開

全社へ展開するシステムとしてJava/CORBAを適用した．必ずしも当初の狙い通りに実現はできなかったが，資材部門以外向けの人材情報システムとしても活用が図れるといった別の面でのメリットが期待できる．今後は，Javaについては，Javaアプレットとしての実用化検証やサーバ側アプリケーションとして適用拡大を促進していく．CORBAについては，全社展開にあたってはIIOPでの通信に対する課題が残っているが，プラットフォームに依存しないという長所を生かしたシステムモデル作りや，レガシーからオープン系までの既存のシステムとの有効な連携ができる点を活かして，システム構築・再利用のための土台として利用拡大を図っていく．


参考文献

1) http://java.sun.com/
2) http://www.org.omg/
3) http://www.borland.com/visibroker/
4) Kimball, R.: The Data Warehouse Toolkit, John Wiley & Sons (1996).
5) Myers, G.: Composite Structured Design, VanNostrand (1978).
6) Gamma, E. et al.:Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley (1995).
7) Liang, S.:The Java Native Interface Programmer's Guide and Specification,Addison-Weslay (1999).
8) http://home.netscape.com/assist/helper_apps/index.html


(平成11年11月12日受付)