Java言語を使用したイントラネット業務システムの開発

Development of Intranet Backbone System Using Java Technology

中西 隆*
Takashi Nakanishi

(2005年6月1日 受理)

キーワード：システム開発，業務システム，オブジェクト指向，Webアプリケーション，java，データベース，イントラネット

1．はじめに

java言語[1]は，1995年にSunMicrosystems社から発表された比較的新しいオブジェクト指向プログラミング言語*1)である．実行コードがJavaVM*2)内で実行されるため，プログラムミスによる“暴走“は原理的に発生せず，きわめて安定している．さらにJava VM環境が存在すれば，一度作成されたプログラムを様々なCPUやOS上で実行できる，“プラットフォーム非依存”という特徴を持つ．コンパイラやJavaVMを含むJDK*3)がSunMicrosystems社などから無料で提供され，現在ではインターネットやイントラネットのWebアプリケーション*4)のサーバ側言語として広く使用されている．

当研究所では，平成13年度から平成16年度にかけて開発した業務処理システムのポータル（玄関）機能から研究管理，技術指導管理，掲示板等の部分についてサーバ側プログラム言語としてjava言語を使用した．ここでは，java言語の持つオブジェクト指向の考え方を活かし，さらにXML関連ライブラリやフレームワークなどOSS*5)で提供されるjavaクラスライブラリを活用することで，少人数でシステムを完成させることができ，プログラムも柔軟な拡張容易なものとして構築することができた．

本稿では，当研究所の業務システムを職員自らの手で開発した体験から，業務システムを自主開発する利点と実行した開発プロセスについて紹介する．さらに，Webアプリケーションとして業務システムを作成するために使用したjava関連技術とその利用方法，効果について紹介する．

2．開発プロセスとプログラム構造

研究管理や技術指導管理などの業務システムは業務の変更によって変化していく性質のものである．たとえば研究管理システムでは，研究申請書，実施計画書，研究経過報告書，研究終了報告書，研究発表伺い書といった様々な管理帳票があるが，業務手順の見直しに伴って，これらの帳票類の追加や削除が考えられ，さらに各帳票に記載される項目は，より頻繁に見直される可能性がある．このような機能（仕様）変更や機能追加に柔軟に対応できるようにするためには，java言語の持つオブジェクト指向の特徴を生かしたプログラム構造が有効であった．

(1) 開発プロセス

当研究所の業務システム開発では，旧情報システムという手本は存在するものの，ユーザ（研究所職員）

* 情報電子部 制御情報系

*1) プログラム作成を実行手続きよりも，“オブジェクト”という機能単位で組み立てていくという概念を押し出したコンピュータプログラム言語．プログラムの開発効率，保守性の面で優れている．
*2) java Virtual Machine: javaのマシン語を実行するソフトウェア
*3) Java Development Kit
*4) ホームページを閲覧するWebブラウザを端末側ソフトウェアとして使用して様々なサービスを提供するソフトウェアシステム
*5) オープンソースソフトウェア

の実際の活動に合致した無理のないシステムとするため，仕様を根本から見直すようにした．

通常，プログラム開発の手順として従来から提示される開発プロセスは，仕様策定，プログラミング，テスト，検収という，いわゆるウォーターフォールモデルである．ウォーターフォールモデルでは，始めに細かな仕様まで作成するため，仕様策定の段階で誤りがあった場合に開発期間が大幅に超過することが指摘されている．また，複雑なシステムでは机上で仕様を完成させること自体が非現実的である．

今回のシステム開発では，仕様策定からプログラム，テストまでを少人数の開発者で内製するため，プログラムから仕様策定へのフィードバックを迅速に行うことができた．このため，始めは最も基本的（必須）な機能によるプログラムを構築することから開始し，このプログラムを実際に動作させて担当者間で協議し，次の第二次仕様のプログラムへと発展させる．このようなプロセスを繰り返すことで，必要な機能を装備したプログラムになる．

この開発プロセスは，具体的には図１のように，まず最小限の機能を装備したプロトタイプ１のプログラムを提供し，評価を行いながら，新しい機能を追加していく．プロトタイプが進むに従って，評価に加わるメンバーを拡大し，さらに詳細な機能を装備していく．実際にプログラムを実行しながら検討することで，検討メンバーでの意思統一がし易くなり，最終的に必要な機能だけを装備したプログラムになる．

この方法の副次的効果としては，機能を段階的に増やしていくため，プログラムが機能追加に柔軟に対応できる構造になる点がある．もし，将来的に必要になるかもしれない積み残しの機能があったとしても，システム運用後に余裕ができた段階で機能を追加するという方法を採ることも可能である．

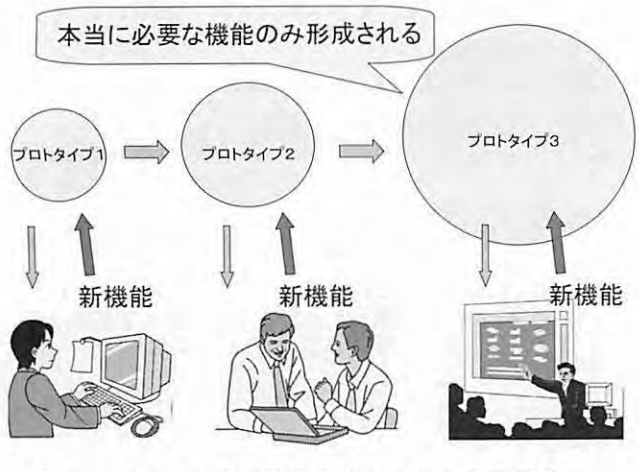


図１　プロトタイプ提供サイクルによる機能追加

(2) プログラム構造

機能拡張や仕様変更の容易性を確保し，機能追加を繰り返す開発プロセスに対しては，java 言語の持つオブジェクト指向の機能が有効であった．

オブジェクトとは，一言で言えばデータ（属性）と機能を持った単位である．オブジェクトは，“継承”によって，元のオブジェクトの機能に新しい機能を追加したオブジェクトを生成することができる．

本システム開発では，機能の単位をできるだけひとつのオブジェクト（java ではクラス）として分離できるような設計を行った．オブジェクト内部のデータへのアクセスは，原則的にメソッドを介して行う（set, get メソッドによる）．図１のプロトタイプ提供サイクルでは，１サイクルでの機能追加が最小限のオブジェクトの拡張（オブジェクト内での変更）で対応できるようにした．

(3) データベースオブジェクト

業務システムの機能変更はデータベース構造の変更を伴うことが多い．このため，プログラムの各所にデータベーステーブルへのアクセスを行う部分が分散していると，データベースの変更によって書き換えなければならない部分が多くなり，バグの混入を招きやすくなる．このため，特定のデータベーステーブルへアクセスするプログラムコードをひとつのオブジェクト（データベースオブジェクト）に集約し，その他のオブジェクトからデータベースへアクセスする必要があるときは，データベースオブジェクトのメソッドを介してデータの受け渡しを行うようにした．

(4) XML テキストの活用

本システムで使用しているデータベース管理ソフトウェアである PostgreSQL はリレーショナルデータベース（RDB）である．RDB は２次元テーブル構造を前提としているため，可変個数のデータをひとつのテーブルに格納したり，階層構造を持ったデータを格納する場合，複数のテーブルを用意し，それらを複雑にリンクする形でデータベースを設計しなければならない．

データベース構造が複雑になると，機能拡張や仕様使用変更に伴って複雑なテーブル構造を修正したり再構築する必要がある．これを避けるため，データベースへの格納形式として XML 形式を使用した．可変個数のデータや，階層構造を持つようなデータは，XML テキスト形式でデータを格納する．このことは，データベースの格納効率やデータ検索速度，および XML 形式への変換のオーバーヘッドなどのデメリットが考えられるが，検索キーが含まれないデータなど，速度的に問題が無いと考えられる部分に対して適用した．

(5) MVC モデルフレームワークの採用

Web アプリケーションでは，一般に MVC モデルというスタイルでプログラム構造を構築するのが効率的と言われている．MVC モデルとは，業務処理のロジックを担当する Model，表示部分を担当する View，クライアントからの要求を受け，Model を起動して処理を実行し，View へ制御を渡して表示を行う Controller という 3 つの部分にプログラムを分けて作成しようとする考え方である．これによって，Model 部分を他のアプリケーションへ利用するなどプログラムの再利用性が高まり，また View 部分をデザインセンスがあり，HTML などのホームページ作成に詳しい人が担当するなどプログラムを分担して行うことが可能になる．

本システム開発では，java で MVC モデル開発を実現するためのフレームワークとして一般的なツールとなっている Struts を使用した[3]．Struts の動作の概要を図 2 に示す．

まず，ブラウザからサーバへリクエスト（URL）が発行される(①)．通常この URL は末尾が".do"となっており，これによって Struts エンジンへ制御が移る．Struts エンジンは，設定ファイルから URL に対応するエントリを探し出し(②)，一致するエントリで指定された ActionServlet クラスを呼び出し，制御を渡す(③)．ActionServlet クラスは，対応したロジックを実行し（Model），制御を Struts エンジンへ戻す（④）．このとき実行結果をステータスとして渡す．Strus エンジンは，ActionServlet から戻されたステータスに基づいて，指定された表示ページへ制御を移す（⑤）．

通常，Struts では表示ページとして JSP(java Server Page)を想定しているが，本システムでは同じ jakarta プロジェクト[2]の成果物である Velocity を導入した．

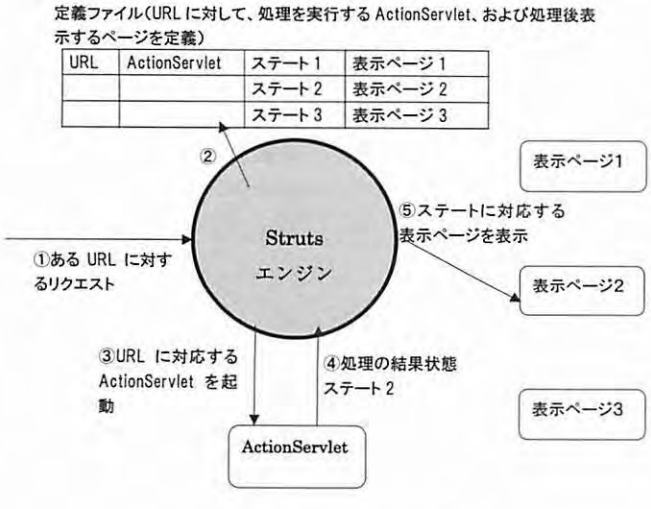


図 2　Struts の動作概要

```
<table border="0" cellspacing="0" cellpadding="4" >
<tr><td>
<i><b>Topic: <%=
meeting.getTopic() %></b></i><p>
<%
eveltlist=meeting.getStoredEventsIterator();
for( i=0; i<eventlist.length; i++) {
  event=eventlist[i];
  yapper =
meeting.getParticipation(event.getFromId()) );
  if ($event.getClass().getName().equals(urlEvent){
    if( repRole.equals(yapper.getRole()) ){
%>
      <font color="<%=blk%>">
        <b><%=yapper.getName()%></b>
      </font>
<%
    }
    else {
%>
      <font color="<%=blu%>">
        <b><%=yapper.getName()%></b>
      </font>
<%
    }
%>
    <a href="<%=event.getStoredData()%>">
        $event.getStoredData()
    </a> <br>
<%
  }
}
%>
<br><%=meeting.getMeetingId()%>:<%=repId%><br>
</td></tr>
</table>
```

(a) JSP によるコード例

```
<table border="0" cellspacing="0" cellpadding="4" >
<tr><td>
<i><b>Topic: $meeting.getTopic()</b></i><p>
#foreach( $event in $meeting.getStoredEventsIterator() )
  #set( $yapper =
$meeting.getParticipation($event.getFromId()) )
  #if( $event.getClass().getName().equals($urlEvent) )
    #if( $repRole.equals($yapper.getRole()) )
      #set( $repId = $yapper.getParticipantId() )
      <font color="$blk">
        <b>$yapper.getName()</b>
    </font>
    #else
      <font color="$blu">
        <b>$yapper.getName()</b>
      </font>
    #end
    <a href="$event.getStoredData()">
      $event.getStoredData()
    </a> <br>
  #end
#end
<hr>
<br>$meeting.getMeetingId():$repId<br>
</td></tr>
</table>
```

(b) Velocity によるコード例

図 3　JSP と Velocity のコード記述の比較

Velocity の言語は表示に特化した機能しか持たず，JSP のソースコードよりも制御構造が捉えやすい特徴がある．図 1 に JSP と Velocity によるコードの記述例を示す．(b)の Velocity コードの方が，HTML ドキュメント中に#で始まる制御コマンドや，$で始まる変数を直接記述できるなど JSP のソースに比べて視認性が良くなっている．さらに，JSP で記述する場合はプログラマが意識しないと表示の役割を越えた機能を盛り込んでしまう可能性もある．

3．システム要素技術

(1) ユーザ認証

システムの認証メカニズムは，Java Servlet 2.4 標準で用意されている認証機構を使用した．特定の URL に対するアクセスには，コンテナの認証メカニズムが働いて図 1 に示すログイン画面が表示される（FORM 認証方式）．ここで，職員番号とパスワードを入力すると，サーブレットコンテナがデータベース内の ID，パスワードを検索し，正しければ認証が成立し，アクセスされたページを表示する．

認証が成立すると，コンテナはセッションを設定し，セッション ID をクライアントの Cookie へ格納する．これ以降のアクセスでは，Cookie からセッション ID を取り出して利用ユーザが識別される（ブラウザで Cookie を有効にしていなければならない）．

Java Servlet 認証では，"ロール"という認証レベルを使用してアプリケーションのアクセス制御を行うことができるが，本システムではロール機構は使用していない．これは，ロール機構はユーザ単位で設定する必要があり，組織変更などでの管理作業が複雑になるためである．

代わりに，アプリケーションごとに組織の所属あるいはユーザ単位でアクセス権を与えるようにした．この方法ではプログラムレベルでアクセス権チェックを

これより先は、ユーザ識別を行います。
以下の内容を入力し、[ログイン]ボタンをクリックしてください.

職員番号(半角):
パスワード:

ログイン | HELP ?

重要！！
ログインできない場合は、イントラネットセキュリティの設定
に従って、セキュリティを設定してください。

パスワードを忘れた方は、下記にインターネットメールアドレス(@trip.pref.osaka.jpが後に付く方)を入力してください。
そのアドレスにパスワードを通知します

メールアドレス(半角): @trip.pref.osaka.jp
送信

図 4　ログイン認証画面

来　所　対　応　票

受付日時　平成17年 5月 9日　16:33

担当職員氏名	前回担当職員氏名	前回来所日	前回来所目的
中西太郎	中西太郎	平成17年 4月20日	クレーム対策

T R I 番 号	K003347
氏　　名	産技研太郎（サンギケンタロウ）様
会　社　名	大阪府立産業技術総合研究所（オオサカフリツサンギョウギジュツソウゴウケンキュウショ）
所 属 部 署	情報電子部
役　　職	
電 話 / F A X	TEL:0725-51-2525　FAX:0725-51-2522
E　メ　ー　ル	email@addr
住　　所	大阪府和泉市あゆみ野2-7-1
業　　種	公務
資　本　金	○−5千万　5千万超−1億円／ 1億超−3億円／ 3億円超
従 業 員 数	1−5　6−20／ 21−50／ 51−100／ ○101−300／ 301−
来 所 目 的	
試験（機器使用）の目的	1．技術の向上　2．品質管理　3．証明書 4．クレーム対策　5．その他（　　）
試　料　名	提出試料（　　）点

試験コード	試験名	試験係数	条件係数	試料数	備考

報告書受領	要　否	試料別報告	要　否	試料返還	要　否	報告書郵送	要　否

図 5　来所対応票(PDF 文書)

行う必要があるので，プログラムの過程でチェック漏れが発生する可能性があるが，アクセス権設定の作業の簡易性を優先した．

(2) 帳票出力(FOP ライブラリの利用)

業務システムでは決まった体裁の帳票が必要となるケースが多い．本システムでは，たとえば図 1 に示すような来所対応票を出力している．

端末として様々な機種の PC やプリンタが使用される Web システムでは，常に定まった形式の印刷出力を得るために Adobe 社が開発した PDF 形式を利用することが標準になっている．PDF 形式のファイルは，利用するコンピュータや OS，およびプリンタが異なっても同じ印刷出力を得られることを目的として作られた形式である．PDF 形式で帳票を出力することで，印刷はもちろん，帳票をファイルで保存したりメールで送信するなど様々な利用をすることができる．

Java アプリケーションで PDF ファイルを出力するには，いくつかライブラリが利用できるが，本システムでは Apache Software Foundation の XML プロジェクトから提供されている OSS の FOP ライブラリを使用した．

FOP では，帳票文書のページサイズや罫線，表組み，表示フォントなどの体裁を記述した XSL ファイルによって帳票のレイアウトなどを制御できる．複数の体裁ファイル(XSL ファイル)を用意しておくと，ひとつのデータから様々な形式の帳票を出力することができる．このことは，帳票の体裁ごとにプログラムを用意する必要がないことを意味する．

(3) バーコード[4]

電子メディア以外とコンピュータを結びつける方法には磁気ストライプやバーコードがある．磁気ストライプは磁気カードとして多く使用されているが，紙に使用できない点や出力に専用の装置が必要などのデメリットがある．一方，バーコードはプリンタで出力でき，読み取り装置も安価に購入できる．

当研究所では，従来は顧客カード(TRI カード)として磁気カードを使用してきたが，今回のシステムでは，より汎用的に使用できるバーコードに変更した．また，処理書などに識別コードをバーコード印刷することで，紙媒体と電子情報との照合をミス無く簡単に行えるようになった．

バーコードには表現できる文字数によって多くの体系が存在しているが，本システムでは英数字を扱えるCode128 を採用した．

バーコードはサーバ内で画像として生成し，呼び出し側のリクエストに応じて画像を返すサーブレットとして作成した．たとえば，**http://サーバ/Barcode/barcodeimage?code=<コード>**という URL にアクセスすると，<コード>を表現する Code128 のバーコード画像（JPEG）を生成する．

生成するページ内に希望のコードを指定した URL を（IMG タグで）埋め込むことで Web ページにバーコードを表示できる．また PDF 出力する帳票でも，体裁ファイルの記述文法に同様の機能があり，図５の来所対応票のバーコードはこの機能を使用して表示している．

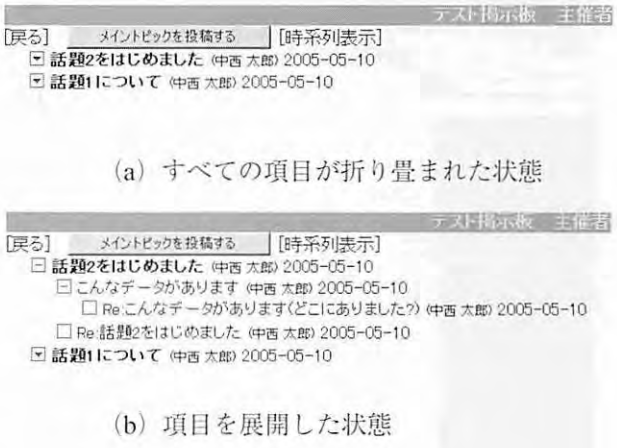
このようにバーコード生成機能を Web サービスとして用意することで，連携している別サーバのアプリケーションからも利用できる．

(4) ブラウザプログラミング

クライアント側に Web ブラウザを使用するイントラネットアプリケーションでは，アプリケーションの操作性を良くするために通常の HTML だけでなく，javaScript や VBScript，および DynamicHTML などブラウザ側で動的な画面を実現する機能を活用する必要がある．本システムでは，DynamicHTML と javaScript を利用して動的な画面を実現した．

たとえば，入力項目の最大数を決定できないような場合，入力者が項目の追加ボタンをクリックすることで入力フィールドを増やしたり，不要になったフィールドを削除する，など Web ブラウザ画面で簡単な編集作業を行うような操作性を持たせることができる．

また，ある選択項目（ラジオボタン）の選択状態によって，別の選択項目のリストを変更するといったページを作成することも可能である．



(a) すべての項目が折り畳まれた状態

(b) 項目を展開した状態

図６　DynamicHTML の使用例(掲示板)

さらに多くの情報が表示される画面で，必要な情報をユーザが選択して表示する機能を持たせることができる．図６は，掲示板システムで話題のツリー構造を折り畳み／展開表示する機能を DynamicHTML で実現した例である．

(5) サーバ連携

当研究所の情報システムは，他に２つのサーバで構成している．これらのサーバは，開発担当者によって使用しているプログラミング言語や OS がそれぞれ異なっている．

これらのサーバが担当する作業を行うときには，別のサーバへ接続するが，このときメインのサーバで認証されたユーザ情報をそのまま利用しなければならない．このための仕組みとして SOAP プロトコルでユーザ情報を伝達する仕組みを用意した．

図７に動作概要を示す．３(1)の「ユーザ認証」の節で示したように，サーバと利用者を関連づけるためにブラウザの Cookie に格納された“セッション ID“を読み出してユーザを識別しているが，Cookie は，それ

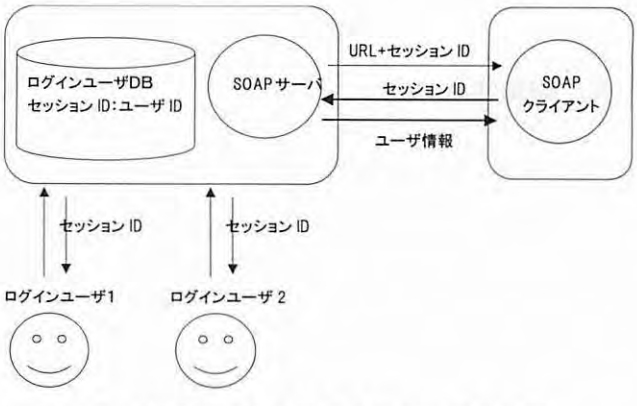


図７　SOAP によるログインユーザの共有

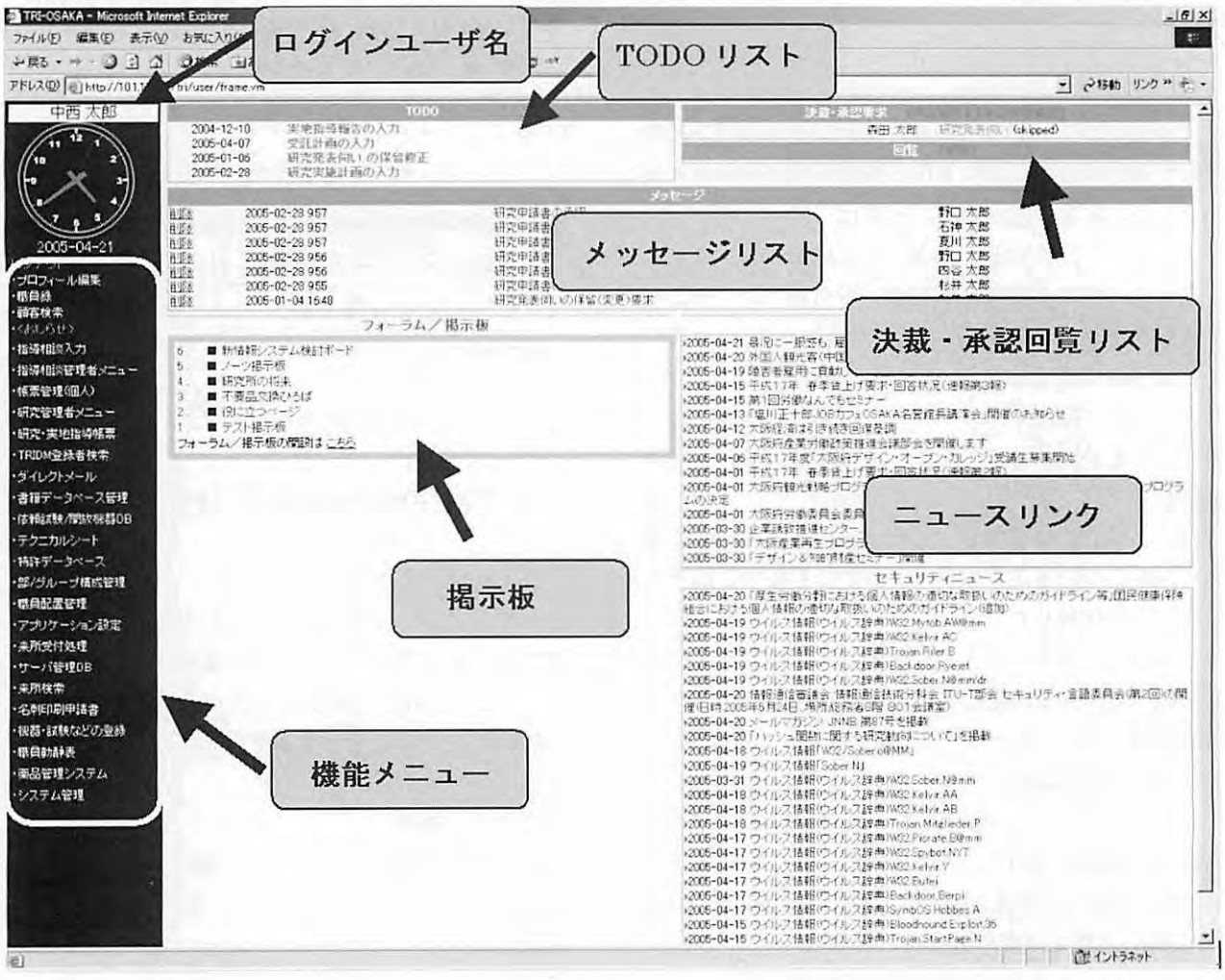


図８　画面構成（ポータルページを表示中）

を作成したサーバからしか読み出すことができない．このため，別サーバにある機能を選択したときは，以下のように sessid というパラメータをつけてセッション ID を伝達するようにした．

http://XXXX/XXXXXX?sessid=a0upHQw4lP76

呼び出されたサーバは，受け取った sessid パラメータを使って，ログイン元(できるだけヘッダの Referer を参照)の SOAP サーバに対して問い合わせを発行することで，ユーザ情報を得ることができる．

4．画面構成（ポータルページ）

システムの画面構成を図８に示す．画面は大きく，左フレームのメニューリストと，右フレームの作業領域に分けた．メニューフレームはシステムでの作業中，常に表示し，任意の時点で異なる作業へ移行できるようにしている．

メニューリストに表示される機能は様々で，職員毎に利用可能な機能を設定する必要がある．これらの機能を各職員が利用できるかどうか設定する場合，個人ごとに指定できるようにすると細かく設定することができるが，業務システムの場合は所属組織単位で扱う方が便利なケースが多い．そこで，各業務ごとに所属（部，グループ）単位での指定と，個人での指定の両方で行えるようにした（図９）．職員が許可された部やグループに所属しているか，個別に許可された職員である場合にメニュー項目が表示される．この方法は，所属変更が発生した場合に各アプリケーションのアクセス制御を設定する必要が無いというメリットがある．

TODO リストには職員が作業しなければならない項目を表示する．このリストをクリックすることで，即座に作業画面を表示する．作業を終えると TODO の項目は自動的に削除される．

承認依頼リストは，研究管理帳票などの決裁／承認の回覧リストを表示する．リストの項目をクリックすることで，帳票の承認画面を表示する．

メッセージリストには，システムからの通知メッセージを表示する（現在は，承認作業における応答メッ

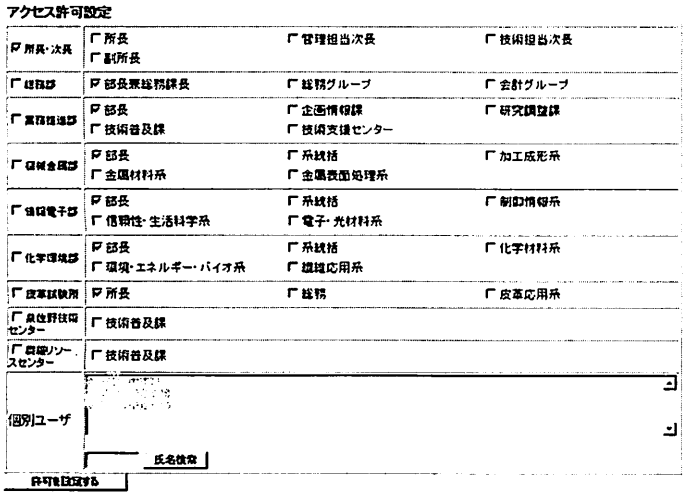


図9 アクセス権設定画面

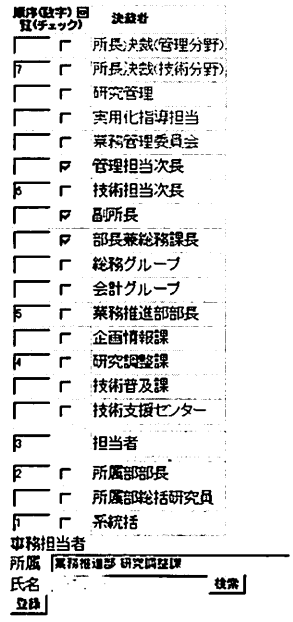


図10 回覧順序の指定画面

セージを表示している).

掲示板リストには，参加可能な掲示板のリストを表示する．最後に訪問してからメッセージが追加されたり，更新された掲示板は色を変えて（赤色などで）表示する．

ニュースリストは，職員に関連のあるインターネットサイトのニュースを抽出し，そのニュースへのリンクを表示する．

5．研究帳票回覧システム（ワークフロー）

研究帳票管理システムは，研究管理業務で必要な各種帳票の作成，回覧，保存，検索などの機能をサポートする．現在使用されている帳票には以下のものがある．

研究申請書	受託研究計画書
研究実施計画書	受託研究終了報告書
研究進捗状況報告書	調査研究計画書
研究終了報告書	調査研究報告書
研究等発表伺	実地指導申請書
	実地指導報告書

これらはいずれも，作成者の所属ラインに沿った管理決裁が必要なものである．

(1) ワークフローテンプレート

帳票は，帳票の種類に対してあらかじめ設定した順序で承認回覧されるようにしている．個別の帳票毎に回覧先を変える柔軟性はないものの，職員が回覧順序について毎回設定する必要が無く，間違いも防ぐことができる．

図 10 は研究実施計画の承認回覧の順序を設定している画面である．承認回覧は数字の小さい順に行い，同一数字は同時回覧を意味する（平行回覧）．

ここで，“担当者”とは帳票に連名で記載されている職員を表す．複数の職員が連名で記載されていれば，それらの職員へ同時に回覧される．

また“所属部部長”，“所属部総括研究員”，“系統括”は，職員が所属しているラインの部門長を表しており，決裁を発行する職員の所属によって回覧される職員が決まる．

これらの書類毎の回覧順序は，ワークフローテンプレートとしてデータベースに格納する．

(2) フローテーブル

一方，個々の承認者の承認作業を制御するためにフローテーブルを定義した．フローテーブルレコードには，承認者ごとに，帳票の種類と識別番号，要求者，承認ステータス（未承認，承認済み）などの情報を保持している．

(3) ワークフローメカニズム

帳票の承認回覧フローを開始する場合，ワークフローエンジンに対して承認回覧フローの開始を指示する．このときワークフローエンジンへは，帳票種別 ID，要求者 ID，連名者 ID を渡す．

ワークフローエンジンは，テンプレートから帳票種別 ID を手がかりとして，フロー順序を参照し，最小の承認レベルの承認者に対するフローテーブルレコードを生成する．

ポータル画面では，このフローテーブルレコードの承認者 ID が自分の職員番号に一致し，承認ステータスが未承認のものを検索し，承認要求リストとして表示される．

承認者が承認要求リスト項目をクリックすると，帳票の内容が表示され，承認者は，それを承認するか保留指示するかを選択する．

本システムでは，承認拒否という選択肢は用意していない．どうしても帳票の承認が不可能な場合は，いったん保留し，申請者に対して承認要求の取り下げを電話や対面で指示することで対応する．

フローテーブルレコードが“承認”されると，ワークフローエンジンはレコードの承認ステータスを“承認済み“に設定し，同じ案件に対する未承認のレコードがないかどうかを調べ，無い場合は現在の承認レベルの次のレベルのフローテーブルレコードを生成する．

このようにして，フローテンプレートの最終承認レベルまで承認された案件は“承認済み“となる．

(4) 帳票オブジェクト

先に示した研究管理帳票類は，それぞれ java のオブジェクト（クラス）として定義した．これらの帳票類には共通部分も多いため，基本オブジェクトを定義し，それを継承・拡張してそれぞれの帳票オブジェクトを生成した．

基本オブジェクトは，添付ファイル機能，決裁（承認）コメント機能を持ち，ワークフローエンジンから制御できるようにするためにワークフローエンジンへのインタフェースを備えている．このインタフェースに従ったメソッドを定義することで，承認プロセスの開始時，承認時，承認プロセスの完了時に必要な処理をワークフローエンジンから呼び出すことができる．

(5) 添付ファイル

帳票はあらかじめ決められた形式に従い，決められた内容をテキストだけで記述するもので，図面やグラフ，写真，などいわゆるマルチメディア情報を扱うことはできない．研究に関する報告を扱う場合には，これらのマルチメディア情報も必要であり，テキストによる情報伝達だけでは不十分である．このため，本システムでは帳票に対してファイルを添付できる機能を持たせた．ファイルはコンピュータで扱えるファイルであれば何でも添付できるようにしている．また，サイズ制限も特に設定していない（ネットワーク通信速度によって実用的な送受信時間が決まり，それによってサイズが決まってくる）．

Web ブラウザを介してファイルをサーバへアップロードする方法は，RFC1867 で規定されているフォームベースのファイルアップロード仕様に基づいてプログラミングを行った．サイズが大きいファイルに対応し，サーバ内での二次的なデータ移動が発生しないよう，クライアントから受け取ったデータを直接指定のファイルへ書き出す機能を持たせている．

6．技術相談管理システム

公設試験研究機関にとって企業に対する技術相談は最も重要な業務のひとつである．当研究所には，平均して毎日数十社程度の企業が技術相談や依頼試験，機器利用に訪れている．また，電子メールや電話等での相談もそれ以上の数に上る．これらの実勢を統計的に把握することは，研究所の業務運営（経営）にとって非常に重要なことである．

しかし，職員にとってこれらのデータの報告作業は非常に煩わしい，よけいな仕事として受け取られてきた．これには，業務運営のシステムもさることながら，以前の情報システムが使いにくく，さらに入力したデータを職員が直接活かすことができなかったことが原因と思われる．

従来のシステムでは，来所相談と電話相談で入力画面の入り口が異なり，さらに画面インタフェースが洗練されていないため入力に時間がかかった．たとえば，選択項目を入力するためには，選択項目を表示するボタンをクリックし，さらに表示された内容をスクロールして該当する項目をクリックするという操作が必要であった．また，繰り返し同じ内容で相談を行うケースが多いにも関わらず，毎回，同じように相談内容などを入力しなければならなかった．

今回のシステムは旧システムでの弊害を考慮して，職員ができるだけ短時間に入力でき，入力したデータを直接日々の仕事に生かせる履歴データとして参照できる機能を装備した．

図 11 に指導相談対応入力画面を示す．ここで工夫した点は，選択すべき項目はスクロールなどすることなく 1 画面で表示し，最適な項目を選択しやすくしたことである．たとえば，選択項目が 10 個以下程度のもの（『対応の方法』，『所要時間』，『業務の種類』）ならば，ラジオボタンまたはチェックボタンですべて画面に表示する．それより多い項目を選択する必要があるもの（『対象物』，『相談内容』）については，ボタンをクリックすると選択項目をすべて一覧表示する別ウィンドウを表示し，該当する項目を選択することで入力画面に反映されるようにした．

今回の相談が以前の継続である場合は，『関連相談を指定する』というボタンをクリックすることで，職員が入力した履歴を一覧表示し，選択することで以前の内容が入力画面に流れ込むようにしている．

以前は，相談報告は後でまとめて入力するケースが

図 11　指導相談対応入力画面

ほとんどで，管理側にとってもリアルタイムの情報を得ることができなかった．今回のシステムでは，病院における電子カルテのように，対応している場面で相談報告を同時に行うという運用を想定し，入力画面で相談者の過去の履歴を検索表示できるようにした．入力画面の『指導相談歴』というフィールドには，入力している職員以外が対応したものを含め，相談者の以前のレコード（履歴）を表示する．これをクリックすることで，相談者が以前にどのような問題を持って研究所を利用したかを参照できる．

7．運用評価

本システムは，原稿執筆時点で本格運用開始後ほぼ1年が経過している．これまで，本システム自主開発の目的のひとつである操作性向上は，非常に満足されている．職員（筆者）が自分で使用するシステムであるので，目的や入力作業の問題点について詳しく把握できていた点が大きかったと思われる．また，DynamicHTML による動的ページは，入力者の負担や画面構成の柔軟性の点で非常に有効であった．

本格運用してからこれまでに大きな変更点として，ワークフロープロセスの変更があったが，ワークフローエンジンとして分離していた構成や，データベース操作をひとつのクラスに集約していた構成が有効に機能し，短時間でシステムを変更することができた．

さらに PDF 出力帳票の形式変更に対する対応も幾度かあったが，ほとんどは体裁設定ファイルのみの変更で対応することができた．

プログラムの変更に関して java で作成したプログラムを修正する場合，サーブレットコンテナの再起動が必要なため，ユーザセッションが喪失してしまう．このため，システムが使用されない夜間から早朝にかけてや，昼休み時間のタイミングでしかプログラムを更新できない点が少し不便である．表示部分を担当する Velocity の変更については，このような問題がない

ので細かな画面表示の修正はシステム稼働中にいつでも実行することができる．

8．おわりに

システム開発のボトルネックは要求仕様の作成である．システムを外注する場合は，予算を決定し，一度にシステム構築を行う必要がある．そのため，要求仕様を明確にしてからソフトウェア会社へ発注しなければならない．しかし，以前の文字端末の時代と比べて最近のGUIはより複雑で，PCやサーバ同士がネットワークで複雑に連携しており，仕様を詳細に決定にすること自体が非現実的である．自主開発を行った今回のシステム開発でも，実際に“仕様“というものの設定の難しさを痛感した．現場と管理側のシステムへの要求が大きく違っており，話し合いだけで妥協点を見出すことは不可能と感じた．

自主開発の利点は，システム開発を長期的に考えられる点であろう．小さいシステムから運用を開始し，実際に動作するシステムを見ながら議論を進めていくことができる．その間にシステムを成長させていけば失敗の危険も少くなる．

従来，システムの自主開発は部門内の限定されたソフトウェアに限られていたが，近年のソフトウェア開発技術の進歩は著しく，この既成概念を覆す可能性が高い．本稿の部分ではjavaを使用したが，より簡単と言われているPHP言語も広く使用されている．言語だけでなく，これらの言語によるライブラリがインターネットを通して，しかも無料で入手できるようになった点がさらに大きい．大げさな表現で言えば，システム開発がDIY(Do It Yourself)で行える環境が整ってきたと言える．

今後は，本業務システムの機能をさらに強化するとともに，システム開発での成果物をプログラム部品として提供できる形でも発信したい．

参考文献

1) java，SunMicrosystems：URL http://java.sun.com/
2) Apache jakarta プロジェクト：URL http://jakarta.apache.org/
3) Struts プロジェクト：URL http://struts.apache.org/
4) バーコードとは，日栄インテック株式会社：URL http://www.barcode.co.jp/about_barcode/index.html