社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－DBS－133　(6)
2004－ＦＩ－ 75　(6)
2004／5／13

Webフォーム基盤アーキテクチャの提案及びその応用事例

土田　正士　　芦田　仁史　　谷口　尚子　　高橋　信雄
(株)日立製作所　ソフトウェア事業部

電子申請システム，ワークフローシステムなどでWeb化が進展している．本報告では，Webフォーム基盤のアーキテクチャとして機能配置方法，記述方法，及びデータ連携・交換方法を提案し，それを実装し電子申請システム，ワークフローシステムに適用した事例を報告する．

Proposal of web-based form architecture and its applications

Masashi TSUCHIDA　Hitoshi ASHIDA　Naoko TANIGUCHI　Nobuo TAKAHASHI
Hitachi, Ltd., Software Division

Web-based systems, such as electric application system and workflow system, are emerging gradually. We propose web-based form architecture that consists of functional component layout, description method, and data transfer technique. We talk about its applications of electric application system and workflow system.

１．はじめに

Webアプリケーションで構築される業務システムは，顧客からの要望に合わせて，短い周期で継続的に拡張や改善が行われている．表示層にはWebブラウザを利用し，HTTPやHTMLなどを解釈できればよいため，PCだけではなくPDAや携帯電話などでもオンライン情報の閲覧，オンラインショッピング，銀行の振込みなどのサービスが利用できるようになってきている．また，アプリケーション層にはJSP(JavaServer Pages)，Servlet，EJB（Enterprise JavaBeans）などのJava™関連技術を利用し，Webフレームワーク（MVCモデル[1][2]）及びコンポーネント部品の機能配置及び記述方法に基き，各々の役割分担に従うことでシステム構築が可能である．[1]したがって，表示層及びアプリケーション層での再利用性が高く，顧客の要望に素早く対応できるWebアプリケーション開発への期待が益々高まってきている．

一方，インターネット及びイントラネットを利用したWebシステムでは入力画面及び出力帳票開発の容易化，入出力データのXML基盤への対応などが要望されている．政府主導によるIT化推進，自治体や民間企業の電子申請システム，ワークフローシステムなどでWeb化が急速に進められている．このように，電子帳票，電子フォーム市場でのWeb化の要望が高まってきている．

これらの背景を踏まえて，Webアプリケーションでの電子フォームシステム・アーキテクチャのコンポーネント，機能構成，及びコンポーネント間インタフェースを開発し，また上記のアーキテクチャをWebフォーム基盤ミドルウェアとして実現した．具体的には，Webフォーム基盤アーキテクチャとして，Webアプリケーションフレームワークを支える機能配置方法，記述方法，及びデータ連携・交換方法を開発した．

本報告では，携帯電話などのモバイル機器を始めとして，RFIDなどユビキタス基盤を支える情報機器への対応も視野に幅広く適用が期待されているWebフォーム基盤のアーキテクチャを提案し，それらを実装し様々な応用事例に適用した結果を報告する．

２．関連動向

本報告と関連する主にWeb基盤でのフォームの研究内容及び標準化動向を紹介する．

中西[3]は，帳票の表示パターンがデータモデル構造から演繹できることを利用して構築された「帳票の概念モデル」でフォーム生成機能を評価して，ユーザの満足度を高めるための改善点とフォーム生成機能の

---

[1] Java™及びすべてのJava™関連の商標及びロゴは米国及びその他の国における米国Sun Microsystems,Inc.の商標または登録商標です．

JavaScript,JavaBeans,JavaServerPagesは米国及びその他の国における米国Sun Microsystems,Inc.の商標または登録商標です．

あり方を議論している．今村ら[4]は，Webブラウザを用いた製品仕様交換を容易にすることを目的として，文書内容制約の入力時チェック機能をもつXML文書入力方式を提案している．文書内容制約のチェックロジックが入力フォームを生成するプログラムから分離できるので，PerlやJavaにより入力フォームを直接プログラムする方式と比べて，文書内容チェック機能の保守を容易にすることができるとしている．野中ら[5]は，対話型ソフトウェアの画面仕様を，Webフォームの意図と表現方式を分離し記述するXForms形式[6]を利用して，対話のセマンティクス記述及びその計算機処理を記述する妥当性を述べている．また，XForms形式の画面仕様を用いて，対話型ソフトウェアの標準機能規模を自動計測する技法も提案している．

また，次世代Webフォーム基盤（ポストHTML，XHTMLの代替）としてW3CでXFormsが規定されている．W3Cは，XFormsを次世代Webフォーム規格として，広く一般の利用者及び開発者に対して，XForms規格の実装と相互運用性の検証を呼び掛けている．XFormsが単なるフォーム記述仕様に留まらず，高度なWebアプリケーションを構築するための基盤技術との期待から，この規格の発展方向が注目される．XFormsは，表示される情報機器に合わせて画面を動的に対応させることを可能とする特徴があり，PDA，携帯電話などのモバイル機器を始めとして，将来的にはバーコード，ICタグ，RFIDなどユビキタス基盤を支える情報機器への対応も視野に幅広く適用が期待できる．

３．現状と課題

Webアプリケーションは，アプリケーションの保守性を向上させるために，表示層及びアプリケーション層の各階層で機能分担する．このような機能分担をMVCモデルへ適用すると，いくつか考慮するべき点がある．まず，表示層で行う単純データチェックや単純データ型(形式)変換と，アプリケーション層で行う複雑なデータチェックや複雑なデータ型(形式)変換の配置及び役割分担を，チェック処理間での重複を避け，お互いに矛盾がないように整合性を保ちつつ決めることが重要である．また，データの入出力に伴う画面と画面遷移が一体化してしまうと保守が困難になる．

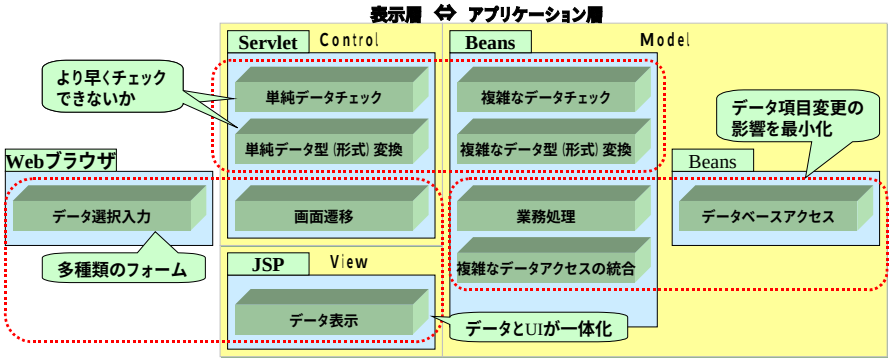

図1　典型的なWebアプリケーションの機能配置

（１）表示と業務ロジック分離の難しさ

View部分のJSPは，JavaBeansあるいはサーブレットによって影響をうけない部分はHTMLで記述し，それらの結果によって動的に変更する部分は<%～%>で囲みスクリプトレットでHTML出力している．つまり，この部分は，業務ロジック（JavaBeans）あるいは制御フロー（Servlet）に影響を受ける．JSPで表示する定型コンテンツの量（この場合はデータ入力項目数など）が多くなってしまうと，<%～%>で囲まれたスクリプトレットが数多くなって，記述されたコードの可読性が悪くなり，再利用性が低くなる傾向になる．JSPは基本的にHTMLの任意の場所に<%～%>で囲まれたスクリプトレットを書ける能力を持つ一方，HTMLとコードの部分が混ざっているので可読性を犠牲にしている．この傾向は，記述量が増えると顕著になる．コードの可読性が低いということは，再利用できるかの判断が困難になることを意味している．また，JSPは本質的にコードであるがゆえに，表示のイメージを直感的に得ることは不可能である．WYSIWYGで見たとおりに印刷するには，さらに別途整形用のプログラミングを必要とする．

業務ロジックに専念して開発を行うべきであるが，画面，制御フロー及び業務ロジックとの連携部分に多大な工数を割かなくてはならない．View部分で入力されたデータは，Model部分で処理されるため，現実的には，画面の設計と業務ロジックの設計の分離が難しい．本来は早期の時点で画面と業務ロジックの整合性を取るのが望ましいが，実際は，利用者のイメージに合わず仕様変更になったりする．表示仕様は，業務要件の変化が激しいために，固定的に作成して終わりということではなく，変更を前提にして表示仕様を考えて行くことが重要になりつつある．一端，作成した仕様を，変更，保守まで考慮した開発の考え方が望まれ

ていると言える．このことから，画面（View）と業務ロジック（Model）をもっと分離させて開発する技法が望まれる．

（２）HTMLフォームの限界

HTMLを利用したフォームによってView部分が実装されてきているが，HTMLによるフォームの設計に様々な課題があり，それらの課題を解消する必要性に迫られている．HTMLをフォームの基盤として利用する場合には，次の課題がある．

（ａ）データとユーザインタフェース(UI)が一体化

HTMLでは，フォームで扱うデータとUIが密接に連携している．フォームを定義する場合，データの扱いとUIは一体化してコーディングされ，当然ながら適用するUIは決まってしまう．例えば，予め用意する選択肢の中から排他的に項目を選ぶ場合はラジオボタンあるいはドロップダウンリストのUIを利用し，また同時に複数の項目を入力させる場合はチェックボックスのUIを利用する．HTMLでは一般的なUIだけが提供され，いずれの場合もデータとUIとは一体化して定義されている．必然的にUIが変更になれば，データ部分も影響を受けることになる．

（ｂ）単純なデータチェックにもプログラミングが必須

フォームの中で閉じて動的にデータ間でチェックが必要な場合があり，フォームへのデータ入力時点でできるだけチェックを済ませたい要望がある．例えば，旅費申請で宿泊費の上限があり，その上限額を超えて申請が行われていないか，宿泊期間の開始日と終了日が逆転して矛盾していないかなど，フォームに入力されたデータを申請時点でチェックできることが望ましい．また，入力データの範囲チェック，データ型のチェック，入力候補値の表示などの豊富なUIコントロールなどは，JavaScriptなどでさらにプログラミングを必要とする．さらに，このようなチェックをHTMLで実装する場合，データやUIを動的に変更できないために，別のHTMLフォームを予め作成しておくか，JavaScriptによるプログラミングを駆使しなくてはならない．

（３）多種類の定型フォーム

Webアプリケーションの画面を目的別に分類すると，ナビゲート画面と業務画面に分けることができる．

ナビゲート画面は，利用者の入力を促すための表示部分と利用者からの入力内容とからなり，それらの入力内容に応じて表現構造が可変となる特徴がある．用途としては，ログイン画面，業務選択画面，エラー画面などがある．一方，業務画面は，紙ベースの帳票と同様に表現構造が不変となる特徴がある．帳票を用途で分類すると，オーダエントリなど入力を中心に行うフォームとデータの活用及び加工を中心に行うレポートに分けられる．

表１　定型フォームの特徴

<table>
<tr><td></td><td>フォーム
（データ入力が中心）</td><td>レポート
(データの加工･出力が中心)</td></tr>
<tr><td>静的コンテンツ
（表現構造が不変）</td><td>定型フォーム
（業務画面）</td><td rowspan="2">（目的別集約結果画面）

レポート
(ドリルアップ･ダウン画面)</td></tr>
<tr><td>動的コンテンツ
（表現構造が可変）</td><td>非定型フォーム
（ナビゲート画面）</td></tr>
</table>

また，フォームは，定型フォームと非定型フォームに区別できる．定型フォームは表示される画面が固定化されている．一方，非定型フォームは表示される画面のコンテンツが，プログラム処理で自由自在に加工される．定型フォームには，単純な画面（入力確認，項目承認など）から複雑な画面（項目間での整合性を確認など）まである．業務の中で画面の遷移に対応して各フォームを用意する必要があるので，定型フォームは多くなる．そのため，画面間でのデータの受け渡しが簡単に記述できることが重要である．

以下の図は，典型的な申請業務の画面制御のための処理制御の流れを示しており，典型的な処理内容とデータ種別毎に，画面構成と画面遷移を要件に従って実装するナビゲート画面（非定型フォーム）がJSP，申請画面，承認画面など業務システムで代表的な業務画面（定型フォーム）がHTMLから構成されていることを示している．ナビゲート画面は受注や出荷など時間に関するの業務上の時間制約，順序性を考慮し業務をナビゲートする用途で作成されるので，概して複雑な画面遷移を含むが種別数は数多くならない．一方，業務画面は業務内容毎に作成されるので，画面数が多くなる傾向にある．

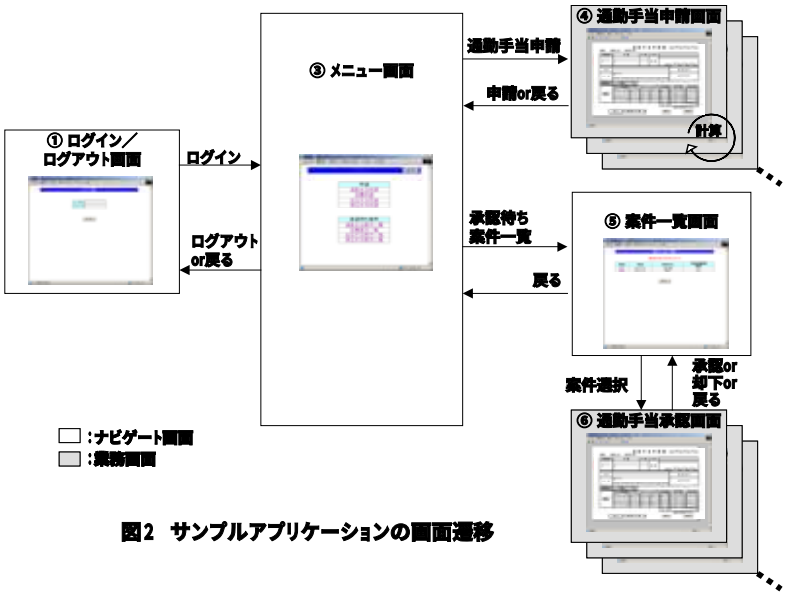

図2　サンプルアプリケーションの画面遷移

（4）定型フォームと業務ロジックの依存性

画面（View）と業務ロジック（Model）の依存性を極力排除して，分離させることが重要であることを述べてきたが，さらに利用者からフォームに入力されたデータとサーバで処理されるデータを分離させる必要もある．この課題を解決するために，なぜXMLが基盤として必要であるか，考えてみたい．定型フォームを利用するWeb環境での業務システムを開発する場合，開発者は何に留意するべきであろうか．

利用者からは，入力のチェックを柔軟に行いたいとか設計時点とテスト時点で画面を変更，追加して欲しいなどと様々な要望がある．また，実際にHTMLフォームでは，HTTP基盤を利用してWebブラウザから入力されたデータがサーバへ転送され，定型フォームで入力されたデータを業務に受け渡すだけではなく，データベースに格納することまでも必要としている．そのため，そのデータを処理するサーバの業務システムでは，データの項目名及びデータ型が変更されるために，データ構造がフォームに依存する問題がある．当然ながら，変更されたフォームに対応する業務システムのプログラムも影響を受けることになる．

このように，画面に着目する利用者と業務仕様に着目する開発者との間で，相互に意見を調整して対処することになる．開発者からは，業務システムのキーとなる開発項目をあらゆる場面で支援することが要請されていると言える．このようなデータ交換・格納用途に合致する基盤として，利用者及び開発者から見てXMLを基盤として採用することは必然である．

4．アプローチ

Webアプリケーションの開発基盤にWebフォーム基盤が加わると，開発する手順及び構成が次のように変わる．

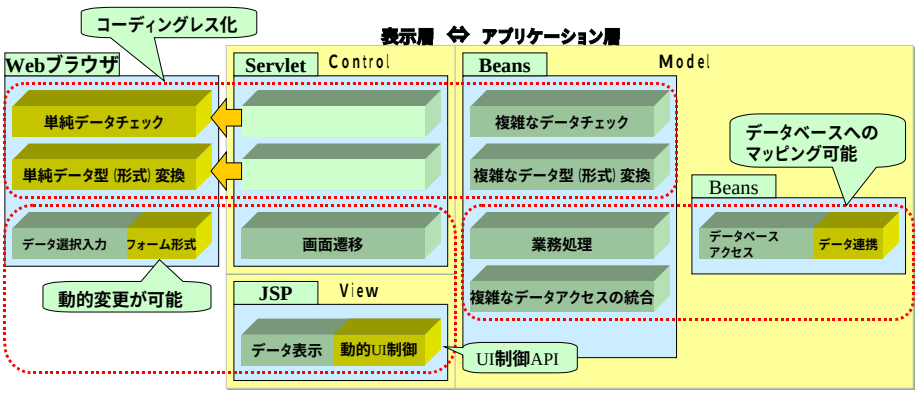

図3　Webフォーム基盤を適用したWebアプリケーションの機能配置

MVCモデルをWebフォーム基盤に適用したWebアプリケーションの開発では，MVCモデルのControl及びViewに適用されることになり，Modelである業務ロジックとは無関係に影響を与えず開発効率や操作性を向上さ

せることができる．即ち，Webアプリケーションの開発基盤として，JavaあるいはJ2EE(Java2 platform Enterprise Edition)のフレームワークを利用してWebフォーム基盤で実現する[2]．HTMLの画面では実現できなかったきめ細かなユーザインタフェースを提供することで，Webアプリケーションの開発課題を解決する．

Webフォーム基盤を利用するWebアプリケーションの特徴は以下の通りである．

- JSPでHTMLによって記述されるビュー部分のロジックが，Webフォーム基盤の定型フォーム表示機能で置き換えが可能である．それによって，HTMLで画面の配置，背景色など開発するコード量を大幅に減らすことができる．
- UIコントロールが業務仕様に沿っており,画面設計時点で設定されるUIコントロールがWebアプリケーション実行時に，柔軟にかつ動的に制御選択可能であり，利用者の木目細かな要望に対応することができる．HTMLフォームでは，Webブラウザで単純なデータチェックやデータ型（形式）変換を行っていたが,Webフォーム基盤ではGUIベースでそれらの処理をコーディングレスで設定が可能である.即ち，新たにUIコントロールの実装のために別途JavaScriptによるプログラミングが不要である．
- Webアプリケーションの中でServlet（画面遷移），JSP（データ表示），及びWebフォーム基盤（データ選択入力，単純データチェック及び単純データ型（形式）変換）と役割分担が明確に分離されるので，各処理部分の再利用が行い易い．

以下では，Webフォーム基盤の開発内容について詳解する．

（１）データとユーザインタフェース(UI)を分離

Webフォーム基盤はデータとUIを分離する構造となっている．フォームを定義する際に，フォームに関連するデータ型及びデータ構造を指定し，個別のプロパティを除いてUI種別を設定する．フォームをWebブラウザで実行する際に，UIの個別プロパティを設定して各利用者に応じたUIのビューが決められる柔軟性を持っている．具体的なプロパティとして，入出力許可属性の設定，選択リスト及びバルーンヘルプの設定，配色属性などがある（表２参照）．基本的に，Webフォーム基盤ではWebブラウザの実行時にUIの詳細を決めてフォームを設定しておけば，表示方法の拡張などにも合わせることも可能である．

また，一般的に画面の設計では，データ入力部分の表示項目名，桁数や入力形式などのデータ型，データ構造がフォーム毎に異なっており，画面部品の再利用を困難にしている．また，これらの木目細かな要件に対応して，各々JSPを駆使してコーディングすることが必要である．一方，Webフォーム基盤では，表示項目名やデータ型，データ構造を定型化したUIコントロールを導入することによって，表示される部品を簡単に貼り付ける操作だけで細かな位置決めを行ったり，テキストボックスの入力値をチェックしたりボタンが押されたらデータベースにデータを送るなどが出来るようになる．基本的にGUIベースの考え方で，入力項目の属性と処理を細かな部分まで設定することができる．従来まではJSPでコーディングされていた処理部分を，Webフォーム基盤設計時点にUIコントロールを貼り込む操作によって代替されていると言える．即ち，人手のコーディング部分は，フォームのソースファイルとして実装されている．このために，画面はコーディングレスで実現することが可能である．

下記はGUIベースでWebフォーム基盤設計時点にUIコントロールの貼り込む操作によってどのようなWebフォーム基盤のソースファイルが作成されるかを図示したものである．画面イメージで表示される情報は，二つに区分されている．まず，画面イメージの裏紙というべき背景レイアウトからなる「プレゼンテーション」である．この「プレゼンテーション」は，帳票ツール，オフィスツールやスキャナなどから読み取られて正確な位置情報とともに記述されている．一方，Webブラウザからデータ入力され，予め初期値として設定された「データ」とフォームを表現するデータ項目名，データ型，データ構造である「スキーマ」とからなる「モデル」である．Webフォーム基盤のソースファイルでは，「データ」とUIである「プレゼンテーション」が分離されて，個別のXMLタグが割り当てられて記述されている．例えば，通勤手当申請書の氏名の表示部分と「日立太郎」というデータは，Webフォーム基盤のソースファイルではマッピングによって関連付けが行われている．サーバに転送されるXMLは「データ」部分だけであるが，必要に応じてXSLTによるデータ変換処理が容易に実装できる．

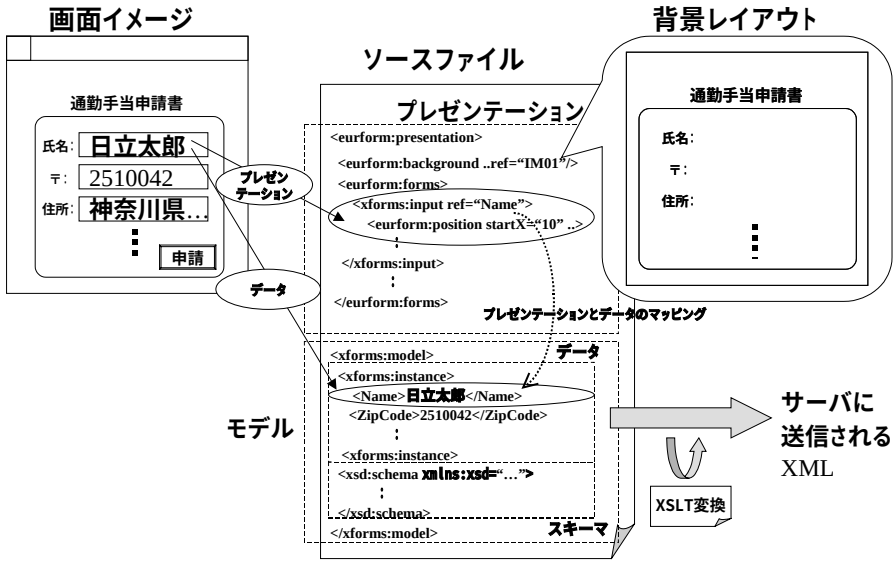

図4　Webフォーム基盤の基本構造

（2）動的に初期値や入力制限を柔軟に変更が可能

Webフォーム基盤では，動的に初期値や入力制限を柔軟に変更することが可能となっている．即ち，初期値設定，入力属性の変更，選択リストの設定，バルーンヘルプの変更，表示色設定，印刷属性の変更などは，フォーム形式は一つであるが，場面によって初期値や入力制限したい範囲や強調したい部分が異なる場合への対応が可能である．例えば，ワークフローで申請者ごとに初期値（名前など）を変更し，また申請者と承認者で強調したい部分や入力制限する部分が異なるなど柔軟に変更したい場合に対応できる．

Webフォーム基盤で提供する様々なUIコントロールによる入力項目の設定のためのAPIは，以下の通りである．

表2　UIコントロールの機能一覧

機能	処理内容	UIコントロール名
UI初期値設定	各UIコントロールの初期値設定	EF,MLEF,DL,RB,CB
入力属性	データ入力可否の設定	EF,MLEF,DL
選択リスト・バルーンヘルプ設定	選択リスト及びバルーヘルプの表示項目を設定	DL,RB,CB
表示色設定	文字色及び背景色の設定	EF,MLEF,DL,RB,CB
印刷属性	印刷対象可否の設定	EF,MLEF,DL
スクリプト機能	データ更新・送信実行時に実行させるスクリプト（項目間チェック，項目間演算結果の代入など）の指定	EF,MLEF,DL
データ属性	データの属性（初期値，データ種別（半角，全角，半角カナ，数字，半角英数字），桁区切り表示，制限指定（最大桁数あるいは最小値～最大値）の設定	EF,MLEF
入力制御機能	入力制御（必ず入力する）の設定	EF,MLEF
選択制御機能	選択制御（必ず選択する）の設定	DL

凡例　EF(EditField)　　：1行分のテキスト入力域
MLEF(MultiLineEditField)：複数行のテキスト入力域
DL(DropdownList)　　：リスト表示された文字列からの選択
RB(RadioButton)　　：ボタンによる単一選択
CB(CheckBox)　　：ボタンによる複数選択

（3）XMLによるデータ管理の柔軟性

Webフォーム基盤では全面的にXML処理基盤を採用している．XMLによって，各種データの仕様が変更された際に大幅な時間とコストを掛けずに異なるアプリケーション間でのデータ連携，交換に対応することが期待されている。開発効率を向上させることができ，また項目の追加・変更が簡単に行えることで開発のスピードが早くなり，データの再利用性が高くなると言われている．業務処理，複雑なデータアクセスの統合（業務間でのXMLデータ変換（例えば，申請業務と審査業務でデータを交換する例）など），及びデータアクセス（データベースへの格納（例えば，申請データがRDBにマッピングされて格納する例）など）への対応が重要である

具体的に，Webフォーム基盤はXMLを採用したことによって次のメリットがある．

（ａ）データ転送でのXML利用

Webフォーム基盤では，Webブラウザから入力されたデータはXMLとしてサーバの業務システムへ転送される．即ち，データにXMLのタグ付けを行い，業務システムでのデータの扱いに基づいたデータの意味付けが可能であることを意味する．XMLでデータを表現することで，データ構造の変更に左右されずに，XMLタグを指定してプログラムで必要となるデータ項目を取り出せるようになる．フォーム形式を変更しても，送信するデータをXMLで表現するので，プログラムへ影響を与えることはない．しかも，Webフォーム基盤はサーバの業務システムでのデータの扱いに合わせて，データ項目名，データ型，データ構造をXSLT変換へ組み込むことも可能となっており，高い柔軟性を保持している．

また，Webフォーム基盤では，HTTP基盤に加えてSOAP基盤もデータ転送のプロトコルとして利用することができる．Webフォーム基盤の作成時にいずれのプロトコルを利用するかを選択できる．データを取得する業務システムのプログラムは，HTTPの場合はServletあるいはJSPで作成する．また，SOAPの場合は，SOAPによる呼出しを受け付けるサービスプログラムで作成する．今後増えるであろうWebサービス基盤へいち早く対応している．

（ｂ）データベースとの連携

データベースには，業務処理で利用する用途及びデータベース構造の種別に基づき三つにパターン分けして格納する．用途として，業務用データベース以外に，申請されるデータ，あるいは，Web フォーム基盤に表示する初期値データを管理するフォーム用データベースも考えられる．データベース構造には，XMLデータとしてそのまま格納することや RDB の正規化リレーションに格納することを想定している．

・フォーム用データベースにフラット形式で格納
・フォーム用データベースに XML データ形式で格納
・業務用データベースに直接格納

業務処理では，例えば送信データの審査処理が複雑で，データの受付処理と審査処理を個別に非同期でおこなう場合，受付処理で，必須項目の有無など簡単な検証処理を行い，受付番号を発行し，フォーム用データベースに格納する．その後，夜間バッチ処理などにより，申請データを業務用データベースに登録し，審査処理では，データを部分的に取り出し，演算やマスタとの検証処理などを行うこともある．

５．適用事例

５．１　電子申請システム

2001 年 3 月に e-Japan 計画が発表され，「電子商取引等の促進」などが重点政策として挙げられている．[7]この流れを受けて，電子政府の実現を目指し，財務規則等で定められている帳票及び文書を電子化することが要請されている．地方自治体でも電子化すべき帳票様式が，4,000 を超えると言われており，総務省ガイドラインに準拠した受付処理・審査機能を実現する必要がある．[8][9]申請様式を GUI で容易に作成・拡張することで，申請業務の拡大に柔軟に対応でき，自治体毎の様々な申請様式にも対応できる電子申請システムが構築されている．

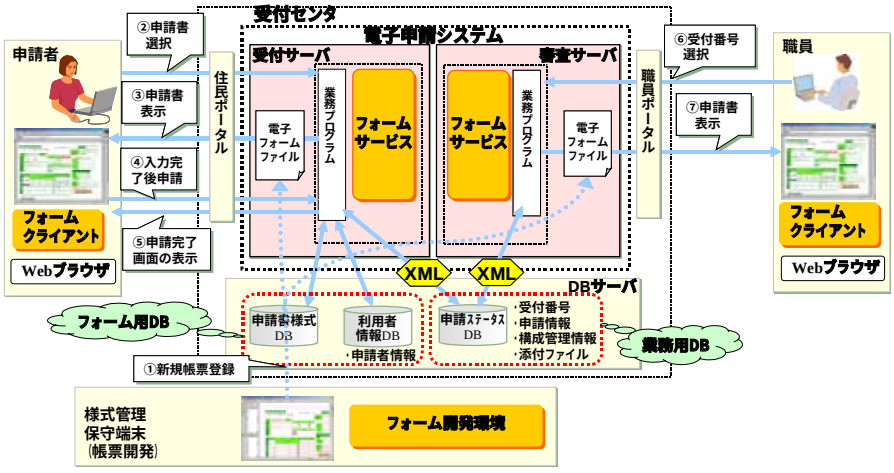

図5　電子申請システムへの適用例

５．２　ワークフローシステム

企業内で利用される各種申請書類，企業間で取り交わされる契約書や注文書，請求書などの帳票類の電子化が序々に進展し，企業内外での業務プロセスをインターネット上で電子的に処理するワークフローシステムが重要になりつつある．紙帳票イメージのままで，しかも帳票スタイルのWeb画面でデータを入力し，直接ワークフローへの投入が可能である．帳票作成からワークフローまでプログラムレスで提供されるので，基本的に業務のワークフローをWebブラウザで電子帳票を利用した申請業務が違和感なく行える．

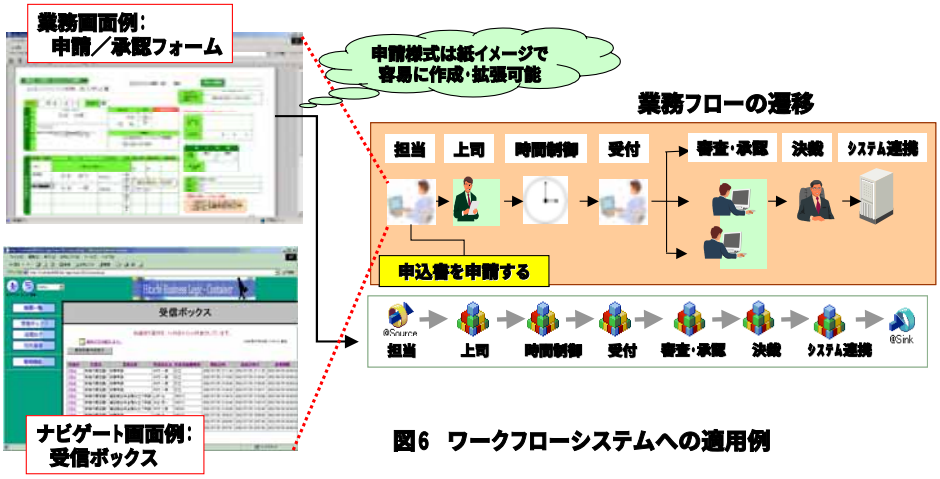

図6　ワークフローシステムへの適用例

６．おわりに

インターネット及びイントラネットを利用したWebシステムで，業務の継続的な拡張，柔軟性の確保によって，さらなる業務効率の向上，スピード経営を目指す要望が益々高くなってきている．これらの要望に応えるため，Java及びXML基盤を基にしたWebフォーム基盤アーキテクチャを提案し，また帳票開発ツールで培った技術と融合させることで，利用者との直接的なインタフェースを支える表示基盤としてWebフォーム基盤ミドルウェアを開発した．既に，Webフォーム基盤ではWebブラウザから入力されたデータを，サーバへSOAP基盤を利用してXMLによるデータ転送に対応している．多様な業務システムの要件に合致させるために，Webサービスへの対応が要望されており，Webフォーム基盤でもWebサービスの進展に合わせて適用して行く．

Webシステムにおいて「帳票」の電子化から，「帳票」と「ナビゲーション」を統合したWebフォームへと進化させ，また様々な情報機器への対応，法制化に伴う帳票長期保存用途への対応など今後の検討課題である．

参考文献：

[1] Crupi, John and et.al.：J2EEパターン－明暗を分ける設計の戦略，ピアソン・エデュケーション (2001).
[2] Kassem, Nicholas and et.al.：Java2 Platform Enterprise Edition アプリケーション設計ガイド －J2EE Blueprint－，Sun Microsystems (2000).
[3] 中西昌武：概念モデルに利用によるフォーム生成ツールの評価--Microsoft Accessの場合--，情報処理学会研究会報告 DBS 131-12 (2003).
[4] 今村誠：WWWブラウザによるXML文書入力方式について，情報処理学会研究会報告 DD 17-1 (1999).
[5] 野中誠：XForms形式の画面仕様書を用いた対話型ソフトウェアの機能規模計測，情報処理学会研究会報告 SE 138-12 (2002).
[6] Dubinko, Micah and et.al：XForms 1.0-W3C Recommendation (2003).
[7] 電子申請推進コンソーシアム編：インターネット電子申請，オーム社 (2002).
[8] 総務省編：地方公共団体における申請・届出等手続に関する汎用受付システムの基本仕様 (2003).
[9] 総務省編：汎用受付システム構築の参考資料(調達編・共同方式の場合)(第1.1版) (2003).