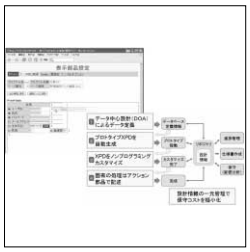

企業内システムにおけるAjax技術の利用

山下哲郎*・黒江　剛・池田和壽

Utilization of Ajax in Enterprise Intranet Systems —— by Tetsuo Yamashita, Takeshi Kuroe and Kazuhisa Ikeda ——
Enterprise 2.0 refers to the recent trend of applying the web technologies widely deployed in Internet services (such as Google and Yahoo!) to enterprise intranet systems so as to improve system designs based on user experience. One such web development technology that is especially attractive for application in enterprise system interface designs is Ajax.
Unfortunately, utilizing the Ajax technology to enterprise systems is challenging, primarily due to the fact that the core of Ajax is JavaScript, and system development using JavaScript raises many issues regarding productivity, quality assurance and maintainability. Additionally, most Ajax-based tools widely available today have issues to overcome such as response time. Therefore, these tools are not suitableto be used as is.
This paper reports on how the authors solved the issues and succeeded in practically applying the Ajax technology into enterprise systems by taking the approach of encapsulating the Ajax technology into the Company's web system development platform “RAKURAKU Framework II”.

1. 緒　言

昨今話題となっているエンタープライズ2.0の要素技術であるAjax（Asynchronous JavaScript＋XML）技術を利用したユーザーインターフェースは、企業内システムにとっても非常に魅力的である。しかしAjax技術の根幹を成しているJavaScript技術は、企業内システム、特に基幹情報システムに求められる開発生産性・品質・保守性を実現することは困難でありそのまま適用することはできなかった。

そこで我々は、Ajax技術を利用するにあたり、基幹情報システムで必要と考えられる適用範囲・機能を絞り込み、なおかつ性能を確保し、ブラウザ間の互換性を維持するための専用部品を開発した。これを当社のWebシステム開発基盤である楽々Framework®Ⅱに実装することにより、従来の開発手法を変えることなく、生産性・品質・保守性を確保しながら、これまで実現できていなかった高いレベルのGUI開発技術を確立し、今後の基幹情報システムにおける開発可能な範囲を広げることができた。

本論文では、その方法論とAjaxを利用した具体的な機能に関して紹介する。

2. 開発の背景

2－1　Ajax技術の登場　インターネットではユーザー参加型ウェブ2.0が広まったが、その要素技術のひとつであるAjaxは、Google Mapsなどで周知のようにユーザーインターフェースの操作性向上に大きく貢献した。さらに最近はウェブ2.0の技術を企業内システムへ適用するいわゆるエンタープライズ2.0の検討が進んできている。その中でも、Ajaxにより基幹情報システムの操作性を向上させることはもっとも考えやすい適用方法のひとつである。

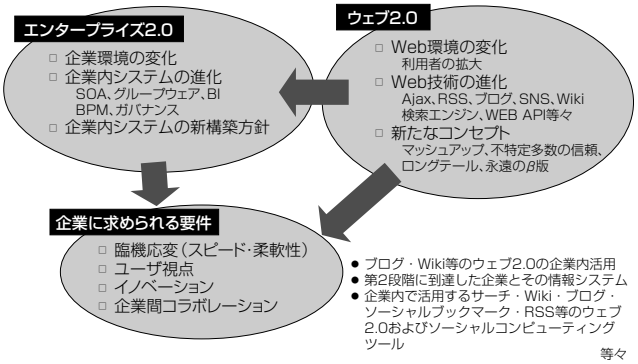

図1　エンタープライズ2.0とは

しかしながら、開発効率の悪いAjaxはGoogle Mapsのように多少コストをかけても多くのユーザーが集中して利用するウェブ2.0には向くが、限られたユーザーのために多数の業務処理プログラムを、QCD※1を確保しながら開発する必要のある基幹情報システムには不向きだった。

そこで、Ajax技術をそのままプログラマに開発させるのではなく、適用範囲や適用方法を絞り、あらかじめ部品化しておくことで、QCDを損なうことなく、Ajaxの恩恵を受けられないかと検討を開始した。

2－2　当社におけるシステム開発と楽々Framework®

情報システム部および住友電工情報システムを含めた当社情報システム部門では、97年からすべての基幹情報システムをWebベースで開発してきた。さらに99年からはJavaによる開発ツール楽々Frameworkを自社開発し、すべての基幹情報システムについてこれをベースに開発してきた。そのメジャーバージョンアップ版である楽々FrameworkⅡでは、開発生産性・品質を向上させるために、プログラマにはできるだけソースコードを書かずにプログラムを作成できる、部品組み立て型の開発手法を確立した。

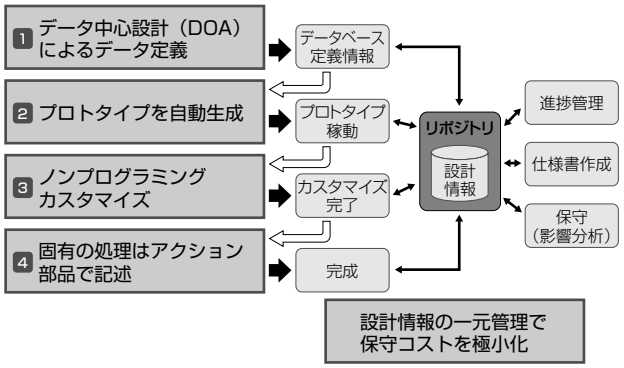

図2　楽々FrameworkⅡの開発手順

2－3　利用部門および開発部門からの要望

このような状況の中で、基幹情報システムを利用するエンドユーザーとシステム開発を行う開発者の両者から下記のような要望が出てきた。

(1) 業務画面の操作性向上

エンドユーザーの多くは、従来からマウスを利用しないホスト系システムを利用してきたため、ブラウザ上でマウスを利用した操作で入力を行うシステムには慣れていない。そのため、**図3**のようなHTMLで作成した表形式入力や一覧表示画面においてもカーソルキーやファンクションキーを活用したマウスレス操作を求められた。また通常のHTML画面では、データの一覧表示を行うと画面全体がスクロールしてしまい、項目タイトルが消えてしまうのが非常に使いづらいとの指摘もあった。

また登録・更新・検索などでユーザーが何らかの入力を行った際には、必ずサーバーへの「送信」ボタンを押し、データをサーバーに送信する必要がある。Webシステムでのサーバー／クライアント間の送受信では、データだけでなく表示されているHTML全体をやりとりする必要があるのでレスポンスが悪いケースも発生する。

図3　表形式入力画面の例

例えば、**図4**は検索画面で検索条件キーワードを入力してデータが1件もなかった場合の画面だが、データがなくエラーメッセージが表示されるだけの画面にも関わらず、送信ボタンを押して始めてエラーチェックが行われるため、全データを入力し終わらないとエラーチェックができない、なおかつ画面全体を書き換えるため、画面がちらつく現象が発生する。

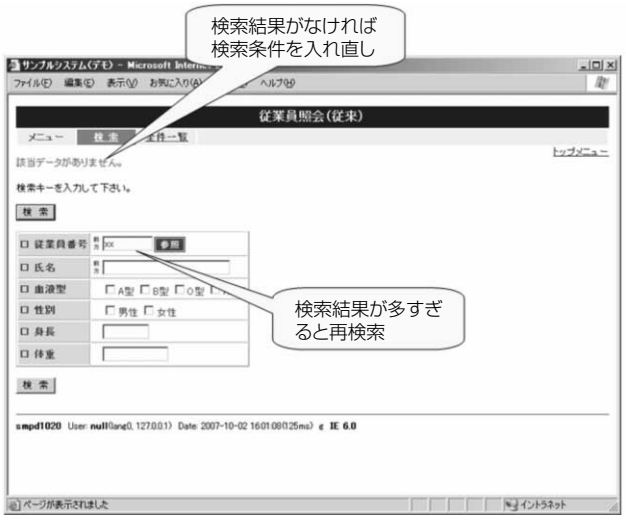

図4　検索画面の例

さらに、従来Web画面では実現できていないガントチャート型データ入力・表示など業務画面への対応も求められ始めた。

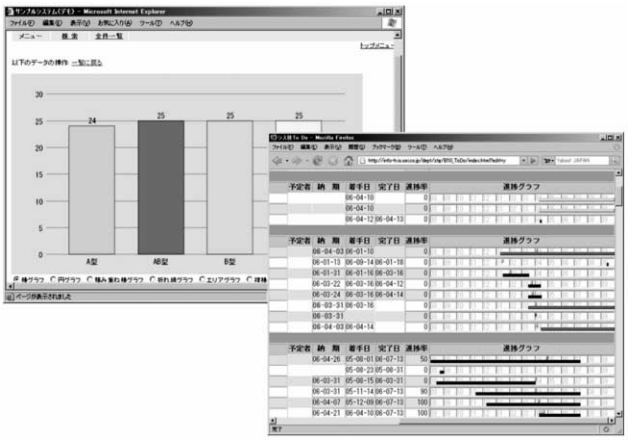

図５　グラフ・ガントチャート表示の例

(2) 開発画面の操作性向上

楽々Framework®Ⅱでは、機能単位で用意している部品とそれに必要なパラメータをXML形式のファイルをテキストエディタで記述することで開発を行っていた。

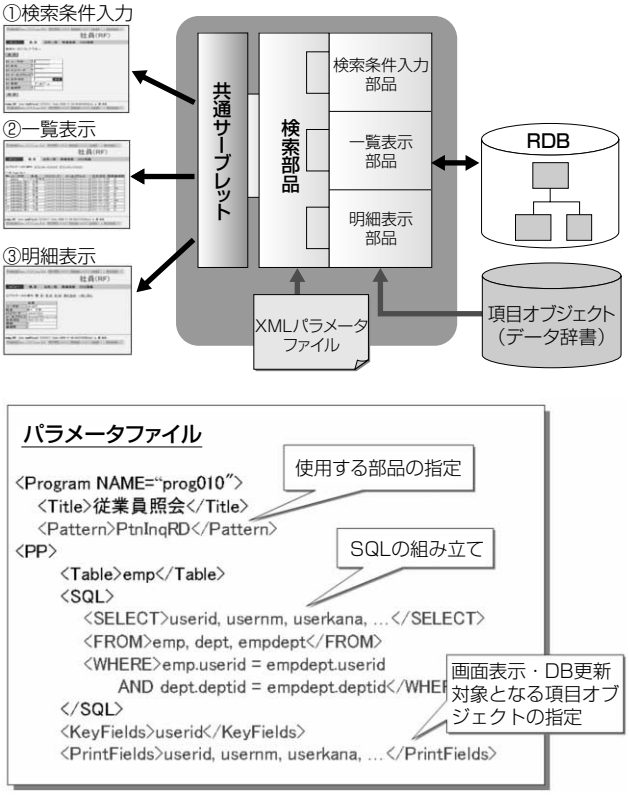

図６　楽々Framework®の基本的な仕組とパラメータファイル

画面レイアウトに関しても、**図７**のように項目のデータ部ごとにセルの連結数を指定する形で指定する。そのため、開発者が最終画面を想定しながら連結数をカウントする必要があり、なおかつ項目の増減によって再度計算しなおす必要があるなど、直感的に画面設定ができず生産性を落としているという指摘があった。

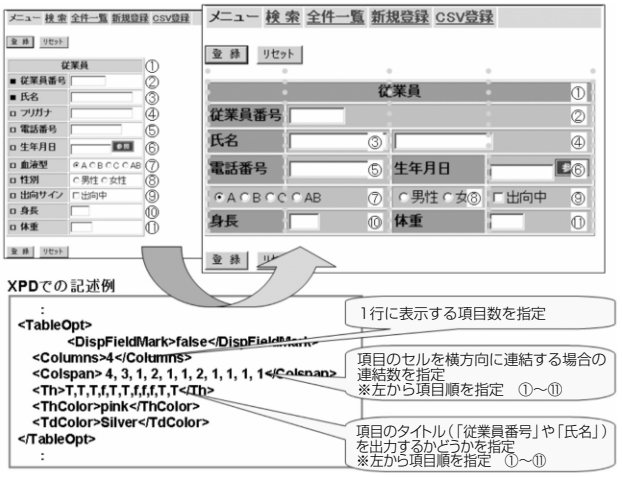

図７　パラメータファイルによるレイアウト設定

(3) 競合他社対策

楽々Framework®Ⅱは社内への最新技術や他社事例の積極的フィードバックのため、社内だけではなく外部への販売を行っている。社外での競争力を高め、製品力向上のためにも競合他社対策は重要である。

その中でもWeb技術の発達に伴い、プラットフォームがWindows限定であればExcel形式での表形式入力の画面ができるなど利用者へのユーザーインターフェースの向上は必須要件になってきている。また、一般的にWebシステムではDBが必須ではないことも多いためSQLに不慣れな開発者への対応も求められた。

3.　Ajaxへの適用対象

３－１　業務画面への適用

(1) 操作性の向上

マウスを利用しないホスト系システムの操作感に近づけるために、画面操作をマウスレスで操作できるようにすることとした。また、表形式入力・一覧表示画面では、Excel形式の操作感に慣れているため、それに近い操作性を実現することとした。Web画面でもExcelのような表計算アプリケーションを埋め込む方式も考えられるが、クライアントの環境が限定されてしまったり端末にインストールされているアプリケーションの種類・バージョンなどにより動作が変わってしまう可能性があるため採用しないこととし

た。具体的な機能として、①項目を固定してスクロール、②項目の表示幅・高さのスライド調節、③項目の入れ替えを実現することとした。

(2) レスポンス向上

データ入力時、利用者が負担に感じるのは、エラーチェックのタイミングである。そのため、従来の、データをすべて入力しユーザーが「送信」ボタンを押してからエラーチェックを行うという操作感を一新し、データを入力した直後にエラーチェックする方式に変更することにより、データ入力そのもののレスポンスを上げることとした。

またデータ入力時、リアルタイムで絞り込まれた候補が表示されることでエラーになることなく、入力補助としても有効であるため、実装することとした。

3－2　開発画面への適用　開発時にもっとも問題なのは、XML形式のパラメータをエディタで直接編集するという直感的でないプログラミングにあるといえる。そのためXMLをプログラマに直接編集させずに、すべてのパラメータをGUIから値を設定できるようにすることを方針とした。

特に不評であった画面設計に関しては、HTMLの画面をWYSWYG的に開発できるように、ドラッグアンドドロップにより直接項目の追加・削除・変更操作できるGUIを開発することとした。

またSQLに関してもAccessによるテーブルの選択・結合を参考に、GUIでテーブルを結び付ければ、自動的にSQL文が生成されるような機能を実現することとした。

4. 基幹情報システムへの適用方針

4－1　QCDの確保　基幹情報システムではQCDの確保が必要不可欠である。一般にAjax技術を適用するためには、JavaScriptやHTMLでのコーディングが必要となる。しかしJavaScriptの場合、Java開発でのEclipseといったような統合開発環境などの強力な開発ツールが十分整備されているとは言えず、コーディングやデバッグが非常にやりづらいため、開発生産性の向上が困難である。JavaScriptは、一般にプログラムの開発生産性向上が可能といわれているオブジェクト指向技術を取り入れてはいるが、Java等のオブジェクト指向言語とは異なり、クラスの雛形（プロトタイプ）をコピーして上書きする方法で実現している。そのため通常のオブジェクト指向とは異なったプログラミング技術が必要となるため、技術習得に余計なコストがかかる。

また、JavaScriptはスクリプト言語らしい柔軟なコーディングができる反面、強固な型チェックが行われないために実行時まで不具合を特定できず、コーディングミスに気付き難いという問題を内在している。そのため、システム全体の品質を確保するのが難しい。このような理由から、Ajaxを利用する場合、プログラマが個々のプログラムでJavaScriptを記述することは基幹情報システム開発では避けるべきである。

楽々Framework®Ⅱが高い生産性を実現できている理由のひとつには、プログラムの機能単位という大きな粒度の部品を組み合わせて開発する、部品組み立て型の開発手法があげられる。そこで、Ajax技術を部品内部に実装することで、プログラマがJavaScriptを意識せずに機能開発できるようにすることを考えた。

4－2　ブラウザ間の互換性　当社では標準ブラウザとしてInternetExplorerとオープンソースのFirefoxを採用しており、当社で開発する基幹情報システムにはこれらのブラウザへの対応が求められる。HTMLやCSS※2は、①ブラウザの種類によって解釈方法が異なる、②指定するプロパティ名が異なるといった、挙動・仕様の違いが存在する。そのため、同じ画面でもブラウザによって見た目や挙動が異なることが考えられる（クロスブラウザ問題）。また同じ種類のブラウザでもバージョンによって挙動が変わることも多々ある。このような異なるブラウザ・バージョンでも見た目、挙動を同じくするためには、JavaScriptの関数レベルでの互換性を逐一チェックし、場合によってはロジックで分岐するなど工数を掛けて対応する必要があり、個々のシステムで対応するのは基本的に不可能である。さらにブラウザのバージョンアップに従って、システムのロジックを見直していく必要があり、保守メンテンナンスに支障をきたす。

このように、クロスブラウザ対応、バージョン互換対応による問題を解決するための工夫が必要であった。

4－3　パフォーマンス　一般にWebシステムでは、画面表示で利用者が待てる時間は3秒といわれている。しかし、JavaScriptはスクリプト言語（インタープリタ言語）であるため、一般的な特性として実行速度が遅い。

また、Ajax技術では動きがあるページを実現するために、DOM※3を用いてHTMLを動的に変更する。DOM操作は端末のマシンパワーを要求する処理であるために、多用すると実行速度が低下しレスポンスが悪化する。社内では古くから使用しているスペックの低いPCが現在も使われている等、様々な種類のクライアントが存在しており、負荷の高い機能を提供することは極力避けなければならない。

このような観点から、DOM操作は必要最低限の利用にとどめることとし、社内の標準スペックのPCで画面表示が3秒以内に完了することを目標に取り組むことにした。

5. 楽々FrameworkⅡへの実装上の課題と解決方法

5－1　Ajax部品の開発　楽々Framework®Ⅱの部品群は従来のWeb技術をベースに開発されており、ページ全体を送信、全体を書き換えることが前提になっている。Ajax技術をこれらの部品に組み込むためには、入力内容の一部を送信したり、ページの一部を動的に変更したりする

といった必要が出てくるが、現状の仕組みのまま部品化することは困難であった。また、楽々Framework®Ⅱは多くの基幹情報システムで利用されていることから、バージョンアップによる互換性の維持が必要であり、リリース済みの機能への影響を最小限に抑えることが求められた。

そこで、既存部品への修正は一切行わず、ページの一部を送信したことを示すパラメータを追加し、オブジェクト指向の継承機能を活用して既存部品をラッピングするAjax部品を追加することにした。Ajax部品はJavaScriptを出力するとともに、内部でパラメータによって処理内容を切り替えるようにした。

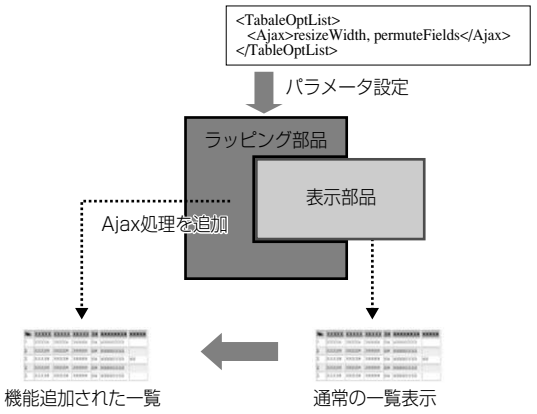

図８　ラッピング部品

Ajax部品の開発により、プログラマは今まで通り部品を選択し、パラメータを設定するだけで、JavaScriptを記述することなく基幹情報システムのプログラムにAjax技術を利用することができるようになった。

５－２　JavaScriptライブラリの開発　Ajax部品の開発にあたり、当初はオープンソースのJavaScriptライブラリである、共通処理の拡張モジュール：prototype.jsおよびユーザーインターフェースモジュール：script.aculo.usを利用することを検討した。まずはプロトタイプを作成して評価を行ったが、いくつかの問題点が判明した。まず、これらのライブラリはJavaScriptの機能を拡張するというコンセプトであるためJavaScript標準のオブジェクトを上書きすることにより実現しており、楽々FrameworkⅡで独自に開発するプログラムと干渉する危険性があった。また、オープンソースであるため将来に亘って期待する機能を継続して利用可能である保証がない点があげられた。特にこの分野は発展途上で、大きな仕様変更などが行われる可能性があり、互換性を維持する上での障害になる。

そこでこれらのライブラリを参考に下記の点に留意して、独自のJavaScriptライブラリを開発することにした。

・Ajax部品に必要な最小限の機能を実装方針とし、できるだけ軽くする
・関数名、変数名など当社独自の接頭辞をつけることにより、標準のスクリプトと異なる名前空間で機能を実現する
・InternetExplorerとFirefoxで可能な限り標準的な機能（関数）のみを利用するようにし、将来に渡って安定した動作を保証する

このようにJavaScriptライブラリも含めて楽々Framework®Ⅱで提供することができるようになり、ブラウザのバージョンアップやブラウザの種類による見た目・挙動の違いに関して、楽々Framework®Ⅱのライブラリ内で吸収する形にできるようになった。これにより、基幹情報システム側でなんらコーディング修正を行うことなく、楽々Framework®Ⅱのバージョンアップだけで Ajax技術が利用できるようになった。

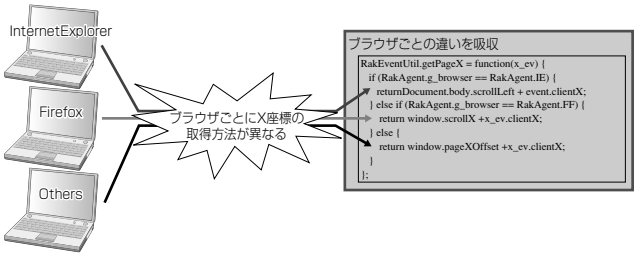

図９　JavaScriptライブラリ留意点

５－３　パフォーマンス・チューニング　JavaScriptの実行速度の遅さは、Ajax部品のプロトタイプ作成において問題になった。当初prototype.jsを利用していたが、10,000行のレコードを画面表示するために60秒を要していた。その原因の一つは前述のJavaScriptライブラリprototype.jsの実装の問題であったため、独自JavaScriptライブラリを作成することとし、10秒程度までの高速化を実現した。さらに調査を進めるうちに、InternetExplorer7ではFirefoxに比べ2.5倍程度のレスポンスがかかることが発覚した。調査によると、InternetExplorer7はFirefoxとの比較で、JavaScriptで2倍、DOMで10倍の処理時間が掛かるという報告があった。どのブラウザでも安定したパフォーマンスを得るために、ブラウザごとにコードレベルのチューニングが必要になった。

目標の3秒の達成のために、JavaScriptで特に遅い操作といわれている、変数へのアクセス回数を減らす、DOM操作をまとめて行う、といったチューニングによって、最終的に2秒を実現した。

```
p_node.style.position = 'absolute'
p_node.style.width = '100px'
p_node.style.height = '100px'
```

.（ドット）演算子を減らす

```
var p_style = p_node.style;
p_style.position = 'absolute'
p_style.width = '100px'
p_style.height = '100px'
```

```
p_node.innerHTML += 'xxxxxxxxx';
p_node.innerHTML += 'xxxxxxxxx';
p_node.innerHTML += 'xxxxxxxxx';
p_node.innerHTML += 'xxxxxxxxx';
p_node.innerHTML += 'xxxxxxxxx';
```

変数へのアクセス回数を減らす

```
var p_text = p_node.inner HTML
        + 'xxxxxxxxx';
        + 'xxxxxxxxx';
        + 'xxxxxxxxx';
        + 'xxxxxxxxx';
        + 'xxxxxxxxx';
p_node.innerHTML = p_text;
```

図10　速度向上への取り組み例

表1　10,000行レコードの画面表示レスポンス比較

	Prototype.js 利用時	JavaScript ライブラリ	チューニング ライブラリ
実　測　値	60秒	10秒	2秒
速度向上度*	−	6倍	30倍

＊速度向上度＝実測値／Prototype.js 利用時の実測値

6.　実現したAjax利用機能

6－1　利用者向け機能

(1) マウスレス操作

一覧形式の画面でも、キーボードのみで項目間のカーソル移動やウインドウを開閉・データの選択等の操作ができるようになった。キー割り当てはシステムでカスタマイズ

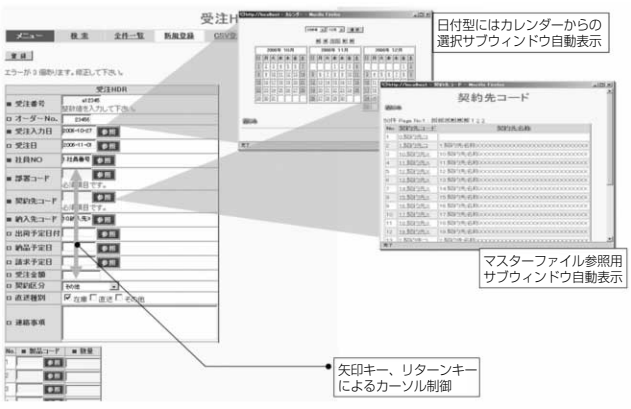

図11　マウスレス操作

できるようになっているため、ホストとほとんど変わらない操作性を実現できた。

(2) ヘッダ固定機能

一覧のヘッダ部分を固定して画面内でスクロールすることで、クライアントアプリケーションと同様の操作を実現できた。

図12　画面内スクロール

(3) 項目の幅と位置の変更

不要な項目の幅を縮めたり、重要な項目を先頭に移動できるようにした。これにより、利用ユーザー自身が各機能を使いやすいようにカスタマイズすることでマイメニュー・マイ機能ともいうべき機能を実現できるようになった。

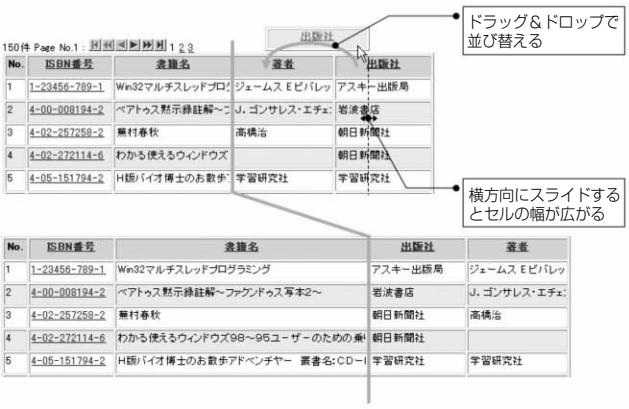

図13　項目幅スライド調整・項目入替

(4) ノンストップスクロール

Webシステムでは、ブラウザ側のリソースに限りがあるため大量データを一度に画面表示できない。通常は一度に

表示する件数を制限してページングを行うことになるが、この機能ではスクロールバーの動きに合わせて動的にデータを取得するのでページングが不要になり、データが続く限りいつまでもスクロールすることができる。これにより一覧性を重視した機能を実現した。

図14　ノンストップスクロール

(5) インクリメンタルサーチ

テキストボックスに入力した文字を、前方一致検索した候補が項目の下部に表示され、入力を進めていくに従い絞り込まれていく機能を実現した。これによりデータの検索性をあげると同時に、入力補助を実現した。

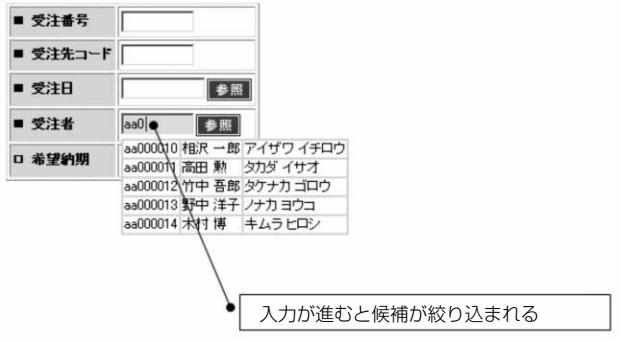

図15　インクリメンタルサーチ

(6) クライアントサイドのエラーチェック

データ入力しカーソルのフォーカスが外れた瞬間にクライアントサイドでエラーチェックすることにより、データを入力したタイミングで入力項目ごとに訂正を可能にした。この機能は利用者の操作性向上だけでなく、サーバー負荷の軽減も同時に実現した。

図16　クライアントサイドのエラーチェック

6－2　開発者向け機能　XMLをGUI編集するエディタとして、RakDesignerを開発した。RakDesigerは、全面的にAjax技術を採用しており、別途クライアント側にアプリケーションは必要なく、ブラウザだけで動作させることができ、開発生産性の向上とともにTCO※4の削減も見据えた機能として実現した。

(1) 画面デザイン機能

開発者向けには、項目をドラッグ&ドロップの直感的な操作で配置することでレイアウトを作成するツールを提供した。

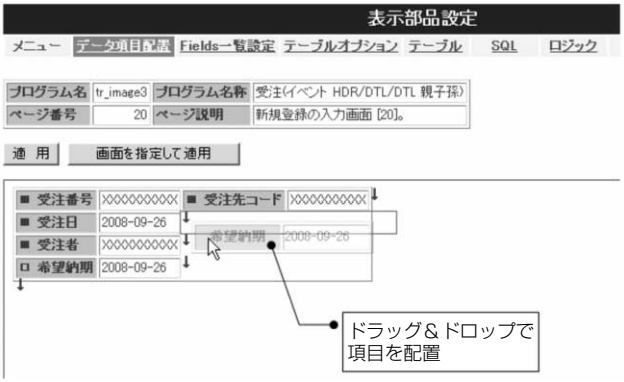

図17　RakDesigner：画面レイアウト

(2) SQL編集機能

GUIでSQLを設定する機能を実現した。テーブルは四角で表現し、テーブルの結合は線で表現した。テーブルを新たに結合する場合には、追加したいテーブルと結合関係に

ある既存テーブルを選択・右クリックし、追加したいテーブルを選択することで自動的に追加される。

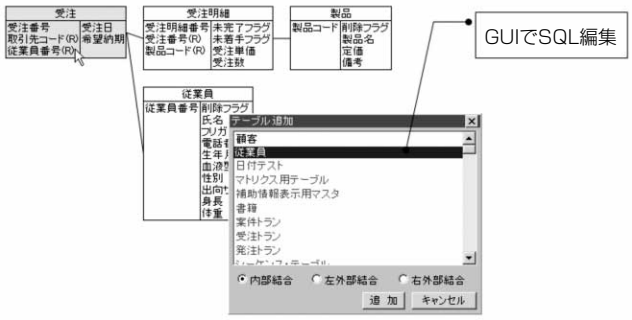

図18　RakDesigner：SQL編集画面

7. 結　　言

当社基幹情報システムのWeb開発基盤である楽々Framework®Ⅱに Ajax技術を導入することにより、基幹情報システムのユーザーインターフェースを大幅に向上する素地ができた。社内では電子申請システム（楽々Workflow®Ⅱ）をはじめ、新規あるいは最近改修されているシステムに関しては順次適用を進めている。さらに、ガントチャートなど新たな機能開発を進め社内ポータルサイトなど情報系システムへの適用や、QuickSolution®※5 とのコラボレーションによるエンタープライズサーチへの展開も進めている。

また住友電工情報システムでは、社内で培ってきた技術をパッケージングし外部へ販売することで、顧客から得られた有益な意見を製品にフィードバックし、当社情報システムへレスポンスよくダイレクトに反映させることができる。今後もこのような環境を積極的に活用し、世の中の動向を注視しながら、製品力向上ひいては当社情報システムの更なる利便性の向上につなげていきたい。

用語集

※1　QCD

製造業における Quality, Cost, Delivery（品質、価格、納期）の略。

※2　CSS

Cascading Style Sheets の略。HTML、XML の表示レイアウトなどの修飾を指定するのための仕様。W3C による勧告の一つ。

※3　DOM

Document Object Model の略。W3C から勧告されている HTML 文書や XML 文書をアプリケーションから利用するための API。

※4　TCO

Total Cost of Ownership の略。「総保有コスト」のことで、ある設備などの資産に関する購入から廃棄までに必要な時間と支出の総計。この文脈では、端末アプリケーションの、インストールからバージョンアップまで含めた総コストをさしている。

※5　QuickSolution

当社が開発・販売しているエンタープライズサーチ・エンジン。特に、大容量・高速検索、辞書不要、多国語対応などが特長。

参 考 文 献

(1) 池田和壽、「DOAによるモデル駆動型のシステム開発－楽々FrameworkⅡの開発－」、SEIテクニカルレビュー第165号（2004年9月）

執 筆 者

山下　哲郎*：住友電工情報システム㈱
ビジネスソリューション開発部
課長　部門スペシャリスト
楽々FrameworkⅡ／楽々WorkflowⅡ
等パッケージ開発・保守業務に従事

黒江　　剛：住友電工情報システム㈱　ビジネスソリューション開発部
池田　和壽：住友電工情報システム㈱　ビジネスソリューション開発部

＊主執筆者