特集論文

Webシステム開発フレームワーク

原田雅史* 河村美嗣*
倉持和彦*
石井 洋*

Application Development Framework for Web-based Systems

Masafumi Harada, Kazuhiko Kuramochi, Hiroshi Ishii, Yoshitsugu Kawamura

要 旨

Webシステムが，技術革新によって従来の情報系システムから企業などの基幹系システムの構築に用いられるようになり，顧客から高品質・低コスト・短納期での構築が求められるようになった。これらの要求を満たすために，三菱電機では早期からフレームワークの開発に取り組んでおり，これまで事業本部対応で，“BizFrame”“TRIWALQ”及び“MIWESTA（MDIS Web Development Standard）”の開発を進めてきている。これらは，多数のWebシステム構築に使用され10～30%程度の生産性向上を達成している。

当社のフレームワークは，生産性向上や品質向上を実現するために“実行環境”だけでなく，ソースコード自動生成ツールなどの“開発環境”や“開発プロセス”も合わせて提供している。また，事業特性に合わせてベースとなるフレームワークに対して様々な機能強化を実施した“実行環境”を開発している。

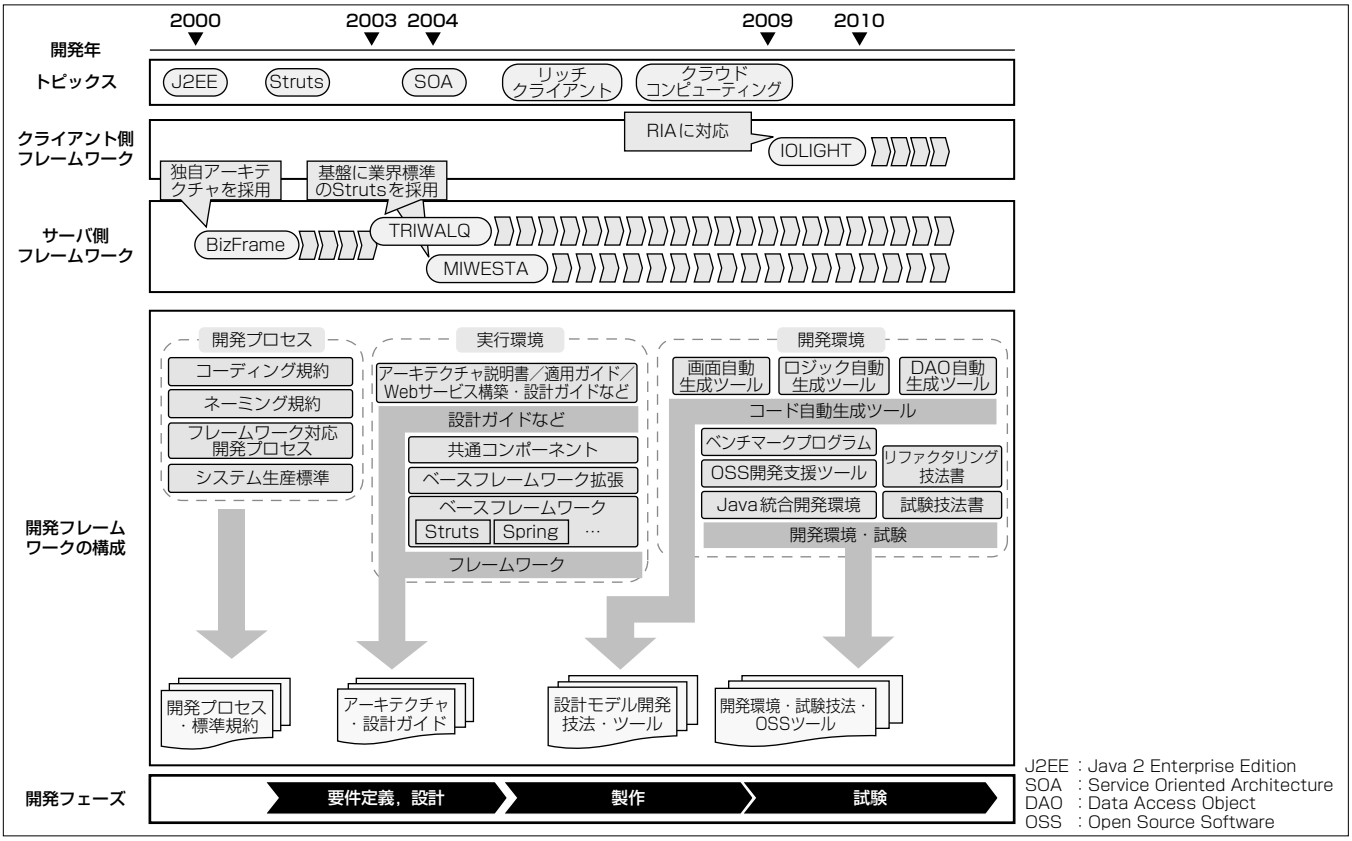
本稿では，次の3つの取組みについて述べる。

(1) CoC（Convention over Configuration）(注1)概念を適用して，設定情報の記述量を削減可能なサーバ側の開発フレームワーク

(2) UML（Unified Modeling Language）(注2)で記述された設計文書を入力とし，生産性向上を実現するソースコード自動生成ツール

(3) RIA（Rich Internet Application）技術を適用して，クライアントサーバシステムと同等の応答性能を実現するクライアント側の開発フレームワーク

（注1） 命名規則などに従うことで，フレームワークに対する設定情報の記述量を削減する概念。
（注2） UMLは，Object Management Group Inc. の登録商標である。



開発フレームワークのロードマップと構成

当社では，Webシステムの黎明（れいめい）期から開発フレームワークに着目し，システム構築時の生産性・品質・保守性を向上するために継続的に開発に取り組んでいる。当初は独自アーキテクチャであったが，業界標準フレームワークであるStrutsを取り入れオープン化を実現した。フレームワークを含む“実行環境”，開発支援ツールなどを含む“開発環境”及びフレームワークに対応した“開発プロセス”を提供することで，各開発フェーズでシステム開発を強力にサポートしている。

*情報技術総合研究所

1. まえがき

開発フレームワークを使用してWebシステムを構築する際に，更なる生産性向上や品質向上を実現するためにフレームワークに代表される“実行環境”だけでなく，それと密接に連携する“開発環境”及び“開発プロセス”も合わせて提供している。これらには，設計・開発ガイドや各種技法書などのドキュメント類も完備しており，はじめてフレームワークを使用するプロジェクトに対して導入しやすくしている。また，事業特性に合わせてベースとなるフレームワークに対して様々な機能強化を実施した“実行環境”を開発している。

本稿では，次の3つの取組みについて述べる。

(1) CoC概念を適用し，肥大化する設定ファイルの記述量を削減したサーバ側の開発フレームワーク

(2) 上流からのフロントローディングによって生産性及び品質向上を実現するために，業界標準の設計言語であるUMLで記述された文書を入力として，Java(注3)ソースコードを自動生成するツール

(3) RIA技術を適用したクライアント側の開発フレームワーク

(注3) Javaは，Sun Microsystem, Inc. の登録商標である。

2. CoC概念を適用したフレームワーク

2.1 Apache Struts設定情報の課題

Apache Struts(以下“Struts”という。)は，Java言語を用いたWebシステムを構築する際に用いるデファクトスタンダードなオープンソースのアプリケーションフレームワークである。**図1**に示すように，Strutsは，ユーザーのボタン押下などによるWebブラウザからのリクエスト(URL)を，Strutsが提供するActionServletが受け付ける。ActionServletは，Struts設定ファイルを参照し，リクエストに対応するActionクラスの呼出し及びリクエストパラメータを格納したFormクラスのインスタンスを作成する。Actionの実行が終わるとActionが指定した論理的なとび先に対応するJSP(Java Server Pages)にフォワードする。

このように設定ファイルにはURLとActionクラスの対応，ActionクラスとFormクラスの対応，JSPへのフォワード情報を保持する。そのため，画面数や画面上に配置されるボタン数が増加すると，それに対応するActionも増え設定ファイルも比例して大きくなる。近年，大規模な業務システムにもStrutsが利用されるようになり，大規模開発におけるStrutsの設定情報が肥大化することで，作成や管理・保守作業が困難になり，問題視されている。

2.2 Struts設定情報削減機能の提供

Struts設定情報削減機能は，設定情報の肥大化の原因となっているAction定義，Form定義，フォワード定義の削減を目的としている。提供する機能は次の4機能からなる。

(1) Action名やForm名の命名規約定義

ユーザーが指定したパッケージ配下に存在するActionクラス及びFormクラスに対して，自動でアクション名やフォーム名を命名する機能である。

(2) Action，FormやJSPの自動マッピング

CoC概念を適用して，Actionに対応するFormやJSPのマッピングを自動で行う機能である。自動マッピングに対応する場所にActionクラス，Formクラス，JSPを配置することによって，設定なしで動作させることが可能になる。

(3) アノテーションによるアクション定義

Action，Formの自動マッピング機能でマッピングしたForm以外のFormを利用したい場合に，Actionクラスのソースファイルに対してアノテーション(注釈)を付加することで，Actionに対応するForm名を指定する。

(4) 物理パスによる呼出し

Actionクラスが次遷移先の論理名を指定する代わりに，JSPなどのURLを直接指定する機能である。

2.3 Struts設定情報削減機能の有効性確認

画面数10，ボタン数15のサンプルアプリケーションを題材にしてStruts設定情報削減機能を用いたものと用いていないものを作成し，コード量を比較した。その結果を**表1**に示す。

Javaコードがアノテーション定義の増加によって4.3%増加したが，設定ファイルは70.5%削減できた。さらに，設定ファイルの詳細は，**表2**のとおりとなった。

Struts設定ファイルであるstruts-config.xmlの記述量が230行から22行と，削減率90.4%に達しており，設定情報削減機能が有効に機能していることが確認できた。

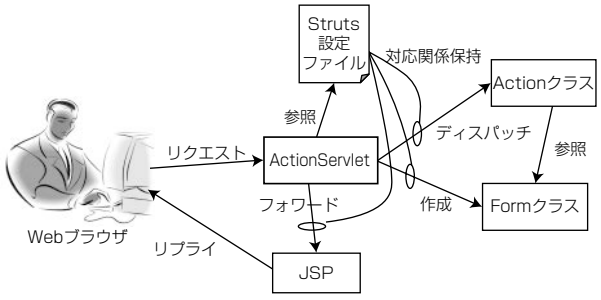


図1．Strutsのアーキテクチャ

表1．アプリケーション作成コード量

種類	非利用(行)	利用(行)	削減率(%)
Java	1,692	1,764	−4.26
JSP	480	418	12.92
設定ファイル	295	87	70.51

表2．設定ファイルコード量比較

ファイル名	非利用(行)	利用(行)
/WEB-INF/web.xml	65	65
/WEB-INF/struts-config.xml	230	22

3. UMLを入力とするソースコード自動生成ツール

3.1 ソースコード自動生成の課題

Webアプリケーションの開発では，オープンソースのアプリケーションフレームワークを利用してアプリケーション構造を，画面を担当するプレゼンテーション層，業務ロジックを担当するビジネスロジック層，データベース接続を担当するデータアクセス層の3層に階層化することが一般的となっている。

当社はこれまで，階層化された各層に対してソースコード自動生成ツールを適用することで，開発効率の向上を図ってきた。プレゼンテーション層・データアクセス層では，定型処理が多いため，設計情報（画面・データベース）から，サンプルアプリケーションの例で内部処理を9割程度生成可能という高い自動生成率を実現していた[(2)]。一方，ビジネスロジック層では，アプリケーションごとに異なる処理が多く，内部処理の自動生成が困難であるという課題があった。

3.2 UML図を入力とするソースコード自動生成機能の提供

3.1節で述べた課題を解決するため，UML図を入力としてソースコード自動生成を行うツールを開発した。UML図とは，ソフトウェア仕様を図式化する統一された手法であり，クラス図やシーケンス図からなっている。自動生成ツールの動作を**図2**に示す。自動生成ツールは，まずクラス図を解析することで，クラスやメソッドの定義情報を読み取る。次にシーケンス図を解析することで，メソッドの内部ロジックを生成する。

3.2.1 クラス図の解析機能

クラス図には，クラスの構成情報が含まれている。自動生成ツールは，クラスの構成情報からクラス名，属性情報（名前・型など），メソッド情報（名前・戻り値の型・引数名・引数の型など）といったクラスのひな型を生成するために必要な情報を取得する。これらの情報からシーケンス図を読み込む準備とソースコードのひな型の生成を行う。

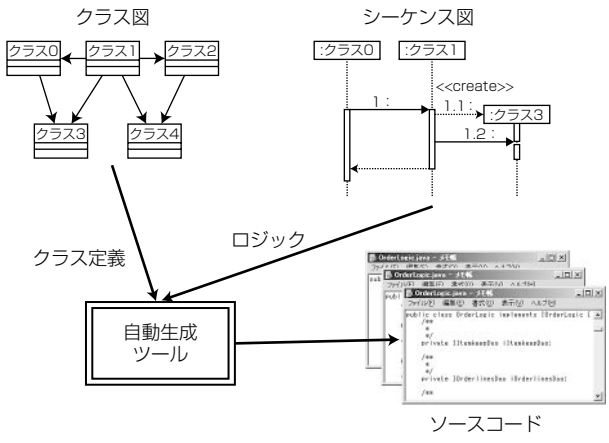


図2．自動生成ツールの動作

3.2.2 シーケンス図の解析機能

シーケンス図には，オブジェクト間のメッセージの流れが時系列順に表現されている。自動生成ツールは，シーケンス図に含まれる全メッセージから，送信元がビジネスロジッククラスであるメッセージを取得し，取得したメッセージをオブジェクト間のメソッド呼出しに変換する。また，通常のメッセージだけでなく，createメッセージやreturnメッセージは，それぞれnew文，return文に変換する。また，シーケンス図は処理の分岐・繰り返しなども表現することができるため，それぞれif文・while文に変換する。このようにして，シーケンス図からメソッド内部ロジックを生成することができる。シーケンス図から生成したロジックを，クラス図から生成したソースコードのひな型と組み合わせることで，設計情報を反映したソースコードを自動生成する。

3.3 ソースコード自動生成機能の有効性確認

ビジネスロジッククラス数3のサンプルアプリケーションを対象として，ビジネスロジック層のソースコードを自動生成し，出力したファイルの行数を比較することで評価した。コード全体に対する自動生成ツールが生成したコードの割合を**表3**に示す。

自動生成ツールが生成したコードは279行であり，全体の93%であった。このように，UML図を入力とすることで，コードのひな型だけでなくメソッド内ロジックも自動生成が可能となり，コードの大部分が自動生成できることが確認できた。これによって，設計情報を実装に反映でき，ソフトウェアの開発生産性と品質の向上への貢献が期待できる。

4 RIAクライアント開発フレームワーク

4.1 WebシステムにおけるWebブラウザの課題

初期のWebシステムでは，データの入力効率が低いマウス操作と，操作ごとに発生する画面の書換えを伴うサーバとの通信による通信データ量の増加が課題となっていたため，クライアントサーバシステム（以下“C／Sシステム”という。）からの移行は限定的なものであった。近年Ajax（Asynchronous JavaScript[注4]＋XML（eXtensible Markup Language））の登場によって，課題となっていた入力効率の改善と画面の書換えを伴わないサーバとの非同期通信による通信データ量の削減が可能になったことで，Webシステムへの移行が進んでいる。

（注4） JavaScriptは，Sun Microsystem, Inc. の登録商標である。

表3．ソースコード自動生成率

	行数（行）	割合（%）
自動生成ツール	279	93
開発者	20	7
合計	299	100

しかし，C／Sシステムは，通信プロトコル上の制約から社内LAN(Local Area Network)での利用が前提であったのに対して，WebシステムではHTTP(Hypertext Transfer Protocol)を用い，社内LANと比べて低速な広域網(インターネット)を利用して構築するケースが大半である。このため，C／Sシステムでは問題とならなかった大量のデータ通信を伴う業務システムでの応答性能の確保が課題となる。

4.2 RIA技術を適用したデータバッファリング機能の提供

RIA技術は，クライアントでの音声や動画といったリッチなコンテンツに注目が集まりがちであるが，業務システムでは処理ロジックを実装できる点が重要である。このことによって，これまでWebブラウザだけでは実現できなかったことが可能になった。

今回，RIA製品の一つであるSilverlight(注5)を利用し，インターネット上のクライアントとサーバとの間で大量のデータ通信を伴う業務システムに対して，応答性能を確保するためのデータバッファリング機能を試作し，その有効性を検証した。

データバッファリング機能とは，非同期通信機能を利用して，ユーザーが入力操作中にバックグラウンドでクライアントがサーバと通信を行い，業務データを先読みしクライアントのメモリ上にデータをバッファリングするものである。

(注５)　Silverlightは，Microsoft Corp. の登録商標である。

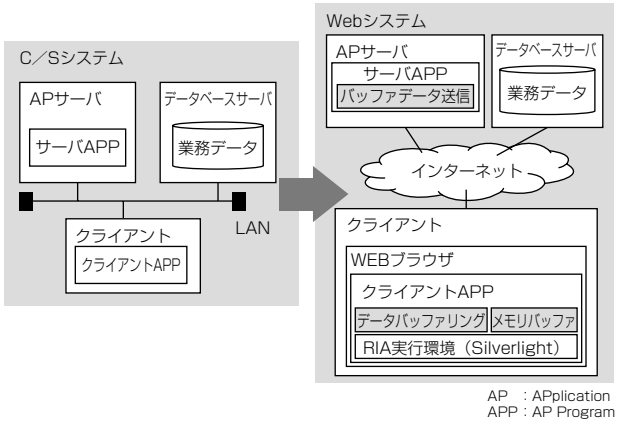
4.3 データバッファリング機能の有効性確認

試作したデータバッファリング機能の有効性を確認するための検証システム(**図３**)を構築し，C／Sシステムと同等の応答性能を達成できることが確認できた。この結果に基づき，実際の業務システムを模したクライアント50台によるサーバ上のデータ2,000件を繰り返し処理するシミュレーションを実施した(**表４**)。シミュレーションの結果，C／Sシステムでは全体の処理時間が313秒(**表４①**)であったのに対し，データバッファリング機能を有効にしたWebシステムでは294秒(**表４③**)となり，データバッファリング機能によって運用形態でもC／Sシステムと同等の応答性能を達成可能なことが確認できた。

今後は，RIAクライアントの開発フレームワーク化に向け，データバッファリング機能など共通機能のライブラリ化，既存のC／Sシステムからの移行ガイドラインなどのドキュメント整備を行う。また，事業側からのニーズに基づいたライブラリや開発支援ツール類の充実化など，フレームワーク整備に向けた取組みを実施する予定である。

5. むすび

当社では，Webシステムが基幹系システムに用いられるようになってからフレームワークの開発に取り組んでおり，これまでに社内外の様々なシステム構築に適用し，生産性向上，品質向上及び保守性の向上を実現してきた。現在，クラウドコンピューティングの台頭によって，情報システムは新しいパラダイムシフトを迎えようとしており，クラウド環境上でのフレームワーク構築にも取り組んでいる(4)。引き続き，最新のシステム構築技術を取り入れた“実行環境”“開発環境”及び“開発プロセス”を三位一体で提供することによって，高い生産性や品質を目指していく。



AP　：APplication
APP：AP Program

図３．RIA技術を適用したデータ検証システムの構成

表４．バッファリング機能による処理時間の比較

	C/Sシステム ①	Webシステム バッファリングなし②	Webシステム バッファリングあり③
データ１件当たりのダウンロード時間	0.3秒	3秒	
データ１件当たりの処理時間	5～10秒		
データバッファ数	0		3
データ：2,000件 端末　：50台 処理時間	313秒	411秒	294秒

参考文献

(1) 原田雅史，ほか： MVCアーキテクチャを実現するアプリケーションフレームワーク“BizFrame”と適用事例，三菱電機技報，**77**，No.7，459～462（2003）
(2) 川口正高，ほか：オープン環境のシステム構築を高品質・短納期で実現するWebシステム開発標準“MIWESTA”，三菱電機技報，**81**，No.7，489～492（2007）
(3) 倉持和彦，ほか：Apache Struts設定削減機能の試作評価，情報処理学会，第71回全国大会，6A-6（2009）
(4) 倉持和彦，ほか：Strutsアプリケーションのパブリッククラウド(Google App Engine)への移行に対する考察，情報処理学会，第72回全国大会，6B-6（2010）
(5) 河村美嗣，ほか：UMLを入力とするソースコード自動生成ツールの開発，情報処理学会，第72回全国大会，6B-3（2010）