特集

スピード時代のシステム構築基盤

System Construction Platform for High-Speed Age

あらまし

システムの短期構築，開発生産性の向上，信頼性向上の実現には，フレームワークの適用が有効である。とくに，大規模システム開発においては，構築スタイルの統一によるプログラムの可読性向上，作業者に依存しない品質・保守性の向上のために不可欠なものとなる。富士通は，フレームワークをIT基盤「TRIOLE」の一部と位置付け，下位インフラの変化に対するアプリケーション変更の抑制や，運用のための仕組みの共通化を目指している。富士通が提供するフレームワークの体系B².Sframeworkでは，従来からミッションクリティカル対応など，より高機能・広範囲へと拡張してきた。加えて今回，クライアント向け製品Client J Frameworkの追加と，Interstage Application Framework Suiteの各種機能の拡充，ミッションクリティカル対応機能を追加した。

本稿では，これらの追加機能を紹介する。

Abstract

Frameworks have proven effective for reducing the time needed to construct a system and also improving productivity and reliability. In the development of large-scale systems, frameworks have become indispensable for improving program usability by making the construction style uniform and improving quality and maintainability independent of the system developers. Fujitsu has positioned such frameworks as part of the TRIOLE IT platform to reduce application modifications due to changes in the low-level infrastructure and make the mechanism of system operation common throughout. Fujitsu's B².S framework has been further extended with higher features and a wider application range for handling mission-critical applications. We have now added a client product called Client J Framework and extended the functions of Interstage Application Framework Suite. We have also added various mission-critical functions. This paper introduces these added functions.

小川俊雄（おがわ　としお）

金融ソリューションビジネスグループソリューション開発センターFBCソリューション部 所属
現在, Client J Framework の開発に従事。

中山朋治（なかやま　ともはる）

金融ソリューションビジネスグループソリューション開発センターFBCソリューション部 所属
現在, Client J Framework の開発に従事。

元田孝司（もとだ　たかし）

ミドルウェアプラットフォーム事業部第一開発部 所属
現在，フレームワークの開発に従事。

保西義孝（ほにし　よしたか）

ミドルウェアプラットフォーム事業部 所属
現在，フレームワークの開発に従事。

ま え が き

システムの短期構築，開発生産性の向上，信頼性向上の実現には，フレームワークの適用が有効である。とくに，大規模システム開発においては，構築スタイルの統一によるプログラムの可読性の向上，作業者に依存しない品質・保守性の向上のために不可欠なものとなる。

アプリケーションを作成する場合，まずアプリケーションの骨格を決めてから，その骨格に肉付けするようにアプリケーションを開発していく。この骨格のことをアプリケーションフレームワークと呼ぶ。フレームワークを使用したアプリケーションの開発では，業務ロジックや画面などのアプリケーション要素を規定に従った形で作成し，フレームワークにはめ込むことでアプリケーションを作成する。これにより，アプリケーションの骨格が明確となり，保守性や再利用性が高いアプリケーションを作成することができる。

B².Sframeworkは，富士通が提供するIT基盤「TRIOLE」における，システム構築のためのフレームワーク体系である。B².Sframeworkを構成する製品の位置付けを**図-1**に示す。

Client J Frameworkは，今回新たに追加された，リッチクライアントをJavaで開発するためのフレームワーク製品である。

e.Frameは，従来から提供しているサーバサイドの業務寄りの機能を提供するオプション製品である。

Interstage Application Framework Suite（以下，AFS）は，従来はサーバサイドの基盤となる汎用フレームワークの製品であったが，今回クライアントサイドへの拡張と，サーバサイドの各種機能拡充を行い，とくにミッションクリティカルシステムに必要な機能の追加を行っている。

本稿では，今回の追加・拡充部分に焦点を当て，以下の3点について順を追って紹介する。

（1） 今回追加されたClient J Framework
（2） 各種機能拡充を行ったAFS
（3） AFSのミッションクリティカル対応

Client J Framework

近年，システムのWeb化はますます加速しており，最近では基幹システムにおいてもWeb化を導入する企業が増えてきている。このような状況の中，クライアントサーバ型システムからのマイグレーションを実現する手段としてリッチクライアントが注目されている。

リッチクライアントは，HTMLクライアントで課題であった以下を実現する。

（1） キーボードのみでの迅速なデータ入力

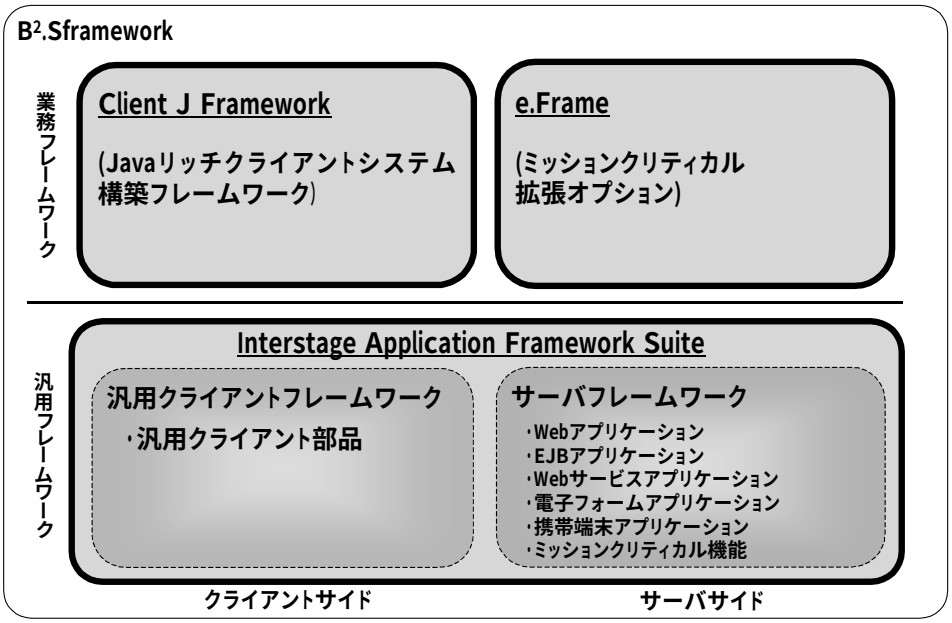

図-1　B².Sframework
Fig.1-B².Sframework.

（2） オフラインでの利用
（3） 頻繁なサーバ通信から解放された応答性の良さ

さらに，Webブラウザをクライアントに用いた場合と同様，配布や管理が容易といったメリットも享受できる。

野村総合研究所が2004年5月に発表したリッチクライアントに関するユーザの意識調査によると，現在利用されているクライアントの中でリッチクライアントの割合は約13%で，2年後には約2倍の28%になるという結果が出ている[1]。

このように今後需要が期待されるリッチクライアントではあるが，開発には高度な専門知識やコーディング技術が必要で，たとえ設計や開発に多くの時間を割いても，システム稼働後のトラブルが少なくないのが実情である。

このような問題を解決するために，富士通ではJavaリッチクライアントのフレームワーク製品であるClient J Framework（以下，CJF）を開発した。

● CJFとは

CJFは，リッチクライアント向け業務アプリケーションを開発するためのフレームワークである。

以下にCJFの特長を述べる。

（1） オープン技術の採用

CJFでは独自のスクリプト言語などを使用せずに，Web環境におけるデファクトスタンダードであるJavaをベースにXMLなどのオープンな技術を採用している。これにより，資産の流用や拡張もスムーズに行える。また，開発エンジニアのスキル確保も容易で，開発生産性の向上も可能である。

（2） MVCモデル(注1)の採用

業務ロジック（Model），画面（View），制御ロジック（Controller：フレームワークで提供）に分離することで，開発者は画面と業務ロジックの開発に専念でき，高品質なシステム開発が可能となる。また，画面と業務ロジックは，並行して開発することができ，開発期間の短縮を図ることができる。

（3） 必要最低限の開発

ロジックの一部を外部定義化していることで，必要最低限の開発のみで希望の動作を実現できる。例えば，画面遷移や画面レイアウトは簡単な外部定義のみで実現でき，これにより開発フェーズの早い段階で実際の画面を用いたユーザ確認ができる。

（4） 容易なイベント処理

検索処理や印刷などの時間のかかる処理はその応答を待たずに次の処理を行い，その応答が返ったときに特定の処理を行うといったケースがある。CJFでは非同期イベントのハンドリングをフレームワークが吸収するため，このような処理も容易に行える。

● 従来の開発手法の問題点とその解決策

従来は，業務画面ごとに画面とロジックを一体で開発していた。このような開発手法では，同じような処理を行う場合であっても，画面ごとに処理を実装しなければならない。その結果，画面数が増加するにつれて開発量も増加し，修正も複数のロジックに手を入れなければならなくなる。また，画面や業務処理以外にも画面遷移・画面構成，イベントハンドリング，データ持ち回りなどの高度なスキルを要する部分では，システムの品質が開発者のスキルに依存する傾向があった。

これに対してCJFでは，開発効率を高めるために，業務画面ごとではなくイベントごとに処理を開発する手法を採る。そのための仕組みとして画面とロジックを分離し，さらにロジックを制御ロジック，業務ロジック，定義情報に分離する（**図**-2）。また，制御ロジックでは次のような機能を提供する。

（1） 様々なイベントを必要とするものだけキューに格納する。
（2） 定義情報を参照して，キューの先頭イベントに対応する業務ロジックを呼び出す。

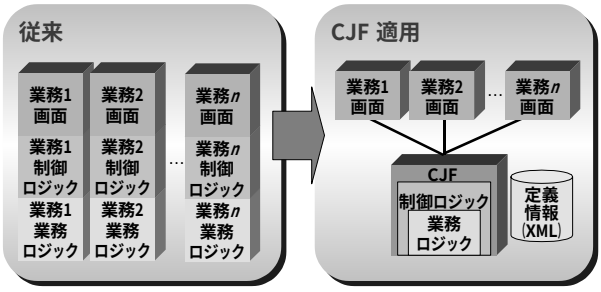

図-2 従来とCJF適用時の開発手法
Fig.2-Conventional-development and CJF-applied methods.

（注1） M（Model：業務ロジックを記述する部分），V（View：画面のデザインを記述する部分），C（Controller：業務処理の連携制御を記述する部分）の三つのプログラミングのデザインパターンであり，それぞれの役割に応じたプログラムを分けて作成する方法である。効率的な開発を可能にし，設計変更への対応も容易となる。

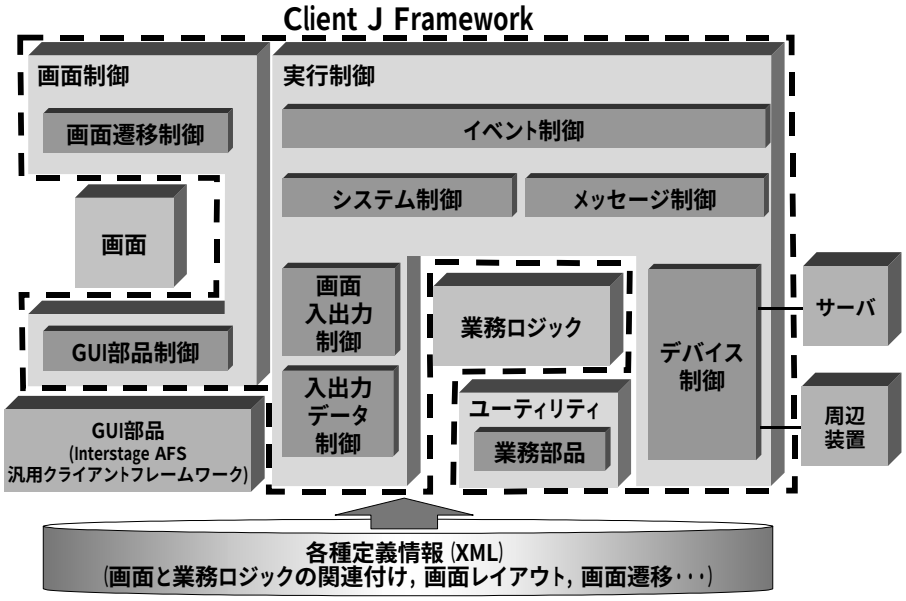

図-3　CJFの構成
Fig.3-Composition of CJF.

表-1　CJFの機能概要

機能	説明
システム制御	・開始・終了手続きや業務ロジック起動のための初期化・終了処理を行う。 ・不慮の例外をハンドリングする。
イベント制御	・画面やデバイスなどから通知されるイベントの一括制御を行う。 ・画面イベントと業務ロジックの対応付けを行う。
画面制御	・定義情報によりレイアウト構成や画面遷移を決定する。 ・画面間でのデータの引継ぎを行う。
データ領域管理	・入出力データ領域の獲得・解放を行い，開発者は意識する必要がない。
デバイス制御	・サーバ通信制御や周辺機器制御の個々のインタフェースを隠蔽し，統一したインタフェースを提供する。
GUI部品データ	・GUI部品のインタフェースを隠蔽し，統一したインタフェースを提供する。
メッセージ制御	・メッセージ内容を業務ロジックから独立管理。メッセージ内容のメンテナンスが容易となる。 ・メッセージポップアップ画面の自動表示・消去を可能とする。
ユーティリティ	・ログ出力や画面イメージ出力などのユーティリティ機能を提供する。

このような仕組みにより，別々の画面で同じ処理を行いたい場合，イベントに対して同じロジックを呼び出すように定義設定をしておくことで，一つの業務ロジックの記述のみで済み，画面の追加や変更が発生しても，業務ロジックへの影響を抑えることが可能となる。また，画面と業務ロジックを分離したことで，開発者のスキルに合った作業を並行して行え，開発期間の短縮も図れる。

● 豊富なCJFの機能

CJFでは開発者を支援するための多くの機能をフレームワークとして提供している。またCJFで提供しているGUI部品に関してはAFSの汎用クライアントフレームワークのGUI部品をベースにしている。CJFの構成を**図**-3に，機能概要を**表**-1に示す。

● 使いやすい開発環境

開発環境は，開発エンジニアのスキル確保や開発生産性に大きな影響を与える。CJFではJava統合開発環境のデファクトスタンダードであるEclipse(注2)をベースとしたJava統合開発環境Interstage Apworks（以下，Apworks）(注3)を使用するため，開発者が使い慣れた環境で開発することができる。

画面の開発はApworksのグラフィカルエディタで実際のイメージを確認しながら行える。制御ロ

（注2） オープンソースの統合ソフトウェア開発環境（IDE）の一つ。Java開発者を中心に急速に普及しており，ソフトウェア開発の共通プラットフォームの標準になると予想されている。
（注3） 効率的なシステム構築を可能とする，富士通のコンポーネント指向Java統合開発環境。

ジックはフレームワークで提供されるので，ロジックを記述して開発するのは業務ロジックのみとなる。どのように画面を遷移させるか定義する画面遷移定義やイベントと業務ロジックとの紐付けを定義するアクションマッピング定義などの定義情報は汎用ツール（Excelなど）で設定し，XMLファイルに変換することにより容易に定義できる。

● 標準システム構成

CJFはクライアント側で動作するため，IIOP（Internet Inter-ORB Protocol）通信などでAP（Application）サーバと直接連携することが可能であり，またHTTP通信でWebサーバと連携することも可能である。このようにクライアントをCJFに移行しても，サーバサイドの資産は生かせるので，システム移行に伴う変更は少なくて済む。

サーバと連携する部分には統一されたアダプタインタフェースを提供する。

CJFを適用した場合の標準システム構成を**図-4**に示す。

Interstage Application Framework Suite

AFSは，アプリケーションフレームワークとして他社に先駆けて2001年より順次次の強化を行っている。

（1） J2EEに準拠したJSP（JavaServer Pages）/Servletを使用したWebアプリケーション

（2） EJB/Webサービス

（3） 電子フォームアプリケーション

（4） 携帯端末アプリケーション

また，新たにクライアントアプリケーション向けの開発からミッションクリティカルな業務アプリケーションの開発までをカバーする製品のエンハンスと基本機能の拡充により，さらなる高信頼・高生産性を実現している。ここでは，下記の構成にしたがって紹介していく。

（1） AFSを使用した開発概要

（2） 各種拡充部分

・汎用クライアントフレームワーク

・基本機能（JavaServer Faces[2] Webサーバ-APサーバ間の横串のログ採取機能）

（3） ミッションクリティカル対応機能

● AFSを使用した開発概要

まえがきで述べたようにアプリケーションを作成する場合，まずアプリケーションの骨格を決めてから，その骨格に肉付けするようにアプリケーションを開発していく。例えば，AFSのWebアプリケーションフレームワークを利用した業務アプリケーションの開発では，データを中心に各種の画面構成要素を整理し，業務ロジックと処理結果を表示するための画面をそれぞれ独立に開発する。これにより業務ロジック変更時の影響範囲を限定することが可能となり，アプリケーションの保守性・拡張性が高

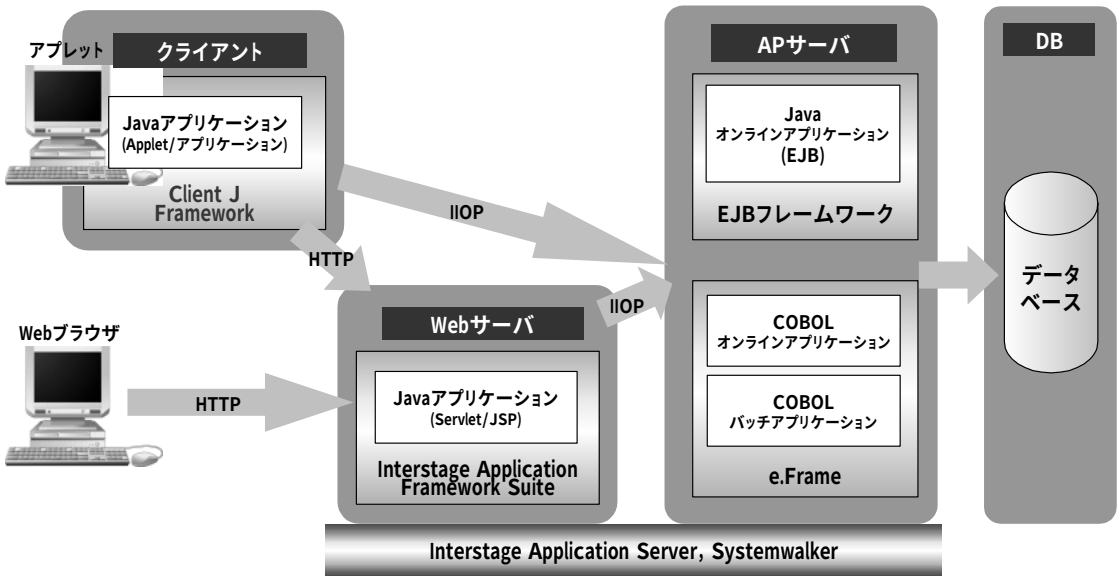

図-4　CJF適用時の標準システム
Fig.4-Standard system by CJF application.

まる。EJB/Webサービス，電子フォームアプリケーション，携帯端末アプリケーションの開発もWebアプリケーションと同じスタイルでの開発を可能としているため，一つの形を習得すれば様々なアプリケーションを同じマナーで記述でき生産性の向上に寄与できる。

また，AFSでは業務依存の仕様変更に迅速に対応できる機構として，画面遷移や処理フロー，項目チェック仕様（単項目チェック，項目相関チェック）を，XMLを用いてアプリケーションの外に記述する機能を提供している。これにより，データ項目のチェック内容などの仕様変更に対して，ソースコードの修正，リコンパイルすることなく対応が可能となる。また，再利用/部品化の促進のため，コマンドスクリプティング機能を提供している。業務ロジックをJavaで記述する代わりに処理フローをXML形式で記述できるようにしており，Javaの記述を必要な部分のみに限定できる。この機能を使用することにより以下のメリットが得られる。

（1） 処理フローが明確になる。
（2） Java処理を細分化することで再利用が容易になる。
（3） 修正時の再構築が不要となり変更が容易になる。

AFSではアプリケーションの骨格のみでなく，アプリケーションサーバが提供する機能をより扱いやすい制御部品として提供することにより，アプリケーションの制御ロジック開発量の削減を実現している。提供しているアプリケーションサーバ用の制御部品の機能には下記がある。

（1） EJBタイムアウト検出時の再実行処理
（2） EJBコンテナの開始・終了時の，業務単位の前処理・後処理・業務停止時の処理

● 汎用クライアントフレームワーク

汎用クライアントフレームワークは，B²-SFrameworkのClient J Frameworkが提供する機能のベースとなる。Webブラウザの普及に伴い，基幹業務のシステム形態がクライアントサーバ型から，Webブラウザ（HTML）ベースへと移行してきた。それに伴い，従来のクライアントサーバ型のシステムで当たり前のように実現できていた下記の機能を実現することが困難で，多大の作り込みが必要になるという切実な問題が発生している。

（1） きめの細かい操作性
（2） 高速な画面遷移
（3） クライアント既存資産の活用

これらの機能を実現するために，長年の各種業務システム構築により培ってきたノウハウをもとにWebベースの業務システム構築を短期間で開発できるための最適なリッチUI（User Interface）向け汎用クライアント部品を提供している。**表-2**に示す画面部品と，その画面部品が持つ各種チェック機構を使用して開発することにより，ミッションクリティカル分野での利用に耐え得るクライアントの開発が可能となっている。

● JavaServer Faces対応

J2EEに準拠したJSP/ServletベースのWebフロントアプリケーションには，現在各社から複数のフレームワークが提供されており，作成したアプリケーションの可搬性に課題がある。

この状況を解決するためにWebフロントの標準アプリケーションフレームワークとしてJavaServer Faces（以下，JSF）の1.0版が2004年3月に策定され，5月に1.1版に改訂された。JSFはJSR127としてJCP（Java Community Process）により策定された仕様であり，J2EE5.0で標準として採用される予定である。

表-2　リッチUI向け汎用クライアント部品（抜粋）

画面部品（36種）	文字列フィールド
	整数フィールド
	日付/時間フィールド
	整数フィールド
	チョイス
	トグルボタン
	イメージボタン
	リスト
フォーカス自動脱出	指定した入力けた数を満たすと自動的に次の入力項目に入力遷移する機能
アラーム鳴動	入力時でエラーが発生したときにアラームを鳴動しオペレータへ再入力を促す機能
各種入力制限	指定されたけた数を入力しないとエラーとなる機能（全けた必須入力）
必須入力	省略できない項目で入力が省略されたときにエラーとなる機能
日本語入力制御	日本語入力フィールドに移動した時点で自動的に日本語入力状態となる機能
高速な画面遷移切替え	高速な画面遷移を実現する画面制御機構

【JSFの特徴】

JSFの特徴を以下に示す。

(1) 再利用可能なUIコンポーネントを使用して画面を作成できる。

(2) クライアントで発生したイベントに対してサーバサイドのプログラムを記述できる。

(3) 利用者がJSFの規約に沿ったUIコンポーネントを作成できる。

今回のJSFへの対応では，最新のJSF実行環境を提供するとともに，標準JSFでは提供されない業務向けの高機能タグを提供する。文字列入力，テーブル，テキストエリアなどを使用することにより，今までと同じ業務画面を今までと同じようにより簡単に構築することが可能となる。高機能タグの一覧表を**表**-3に示す。

また，JSFで作成したWebフロントからは，バックエンドのEJB/Webサービスを簡単に呼び出せるような統一されたインタフェースを提供しており，バックエンドとの連携部分に関し他社と比較し開発効率が向上している。

【JSFを用いたアプリケーションの開発】

JSFを用いたアプリケーションの開発では，Apworksを使用することにより実際に表示される画面イメージを確認しながら開発することが可能である。**図**-5に示すように，画面部品パレットから運用時の画面をドラッグ&ドロップで配置することにより，実際の画面を確認しながら開発することが可能である。これにより，JSP/Servletの開発が従来のクライアント画面の開発と同等の開発スタイル（画面部品をドラッグ&ドロップ）によって開発が可能となり，JSP/Servletの知識を十分に有しない開発者でも開発が容易となる。つまり，Javaの知識を有していれば，Apworksの各種開発支援機能（画面設計，ウィザード・各種定型処理，デバッグ機能，サーバへの資産配備）を利用することによっ

表-3 高機能タグの一覧（抜粋）

コンポーネントタグ（21種）	文字列入力
	整数入力
	日付・カレンダ入力
	十進，十進小数点入力
	チェックボックス
	ラジオボタン
	プッシュボタン
	コンボボックス
	リスト
画面部品タグ	テーブル
	ツリー
	リスト
クライアントスクリプトタグ	入力フォーム
	インプット
	テキストエリア
	複数選択

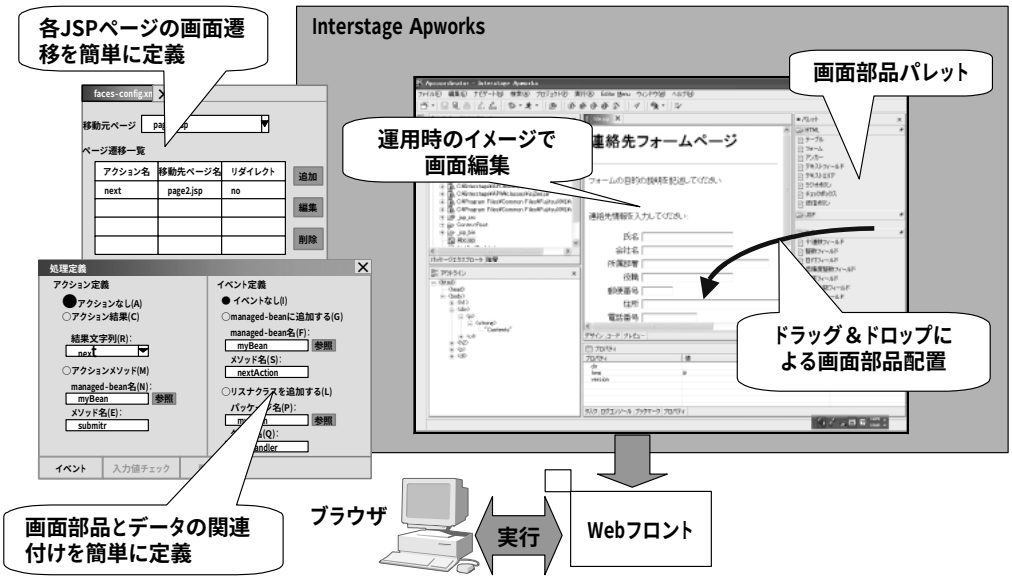

図-5 Interstage Apworksを使用してのJSPの編集
Fig.5-JSP editing JSP by use of Interstage Apworks.

て，JSP/Servletの開発知識がなくても画面の開発とビジネスロジックの開発に集中することができ，Javaを使用したサーバサイドの開発効率の向上が見込まれる。

● **Webサーバ-APサーバ間のログ採取機能**

J2EEを使用したシステムでは，そのスケーラビリティを確保するためにWebサーバとAPサーバとを分離した構成を採用している（図-4）。

この構成においてWebアプリケーションの性能を改善するには，クライアントからのリクエストに対する，WebサーバからAPサーバまでの振る舞いを追跡する必要がある。通常，このために，各業務アプリケーションの要所でログを採取しているが，採取するログがWebサーバとAPサーバそれぞれで独立しているため，WebサーバからAPサーバのリクエストを関連付けるには多くの困難を伴っており，調査に多大な時間を要する一つの要因となっている。

この問題を解決するために，WebサーバとAPサーバとで一意となるID（コンテキストID）をフレームワークによりログに付加する機能を提供する。この機能により，WebサーバとAPサーバの双方のログの対応付けがIDにより可能となり，簡単に対応付けができるようになる。また，このときアプリケーションの変更は不要であり，フレームワークの動作を規定する定義ファイルを変更するのみとなる。

ミッションクリティカル対応

システムライフサイクルの短期化に伴い，基幹業務システムを，システム規模や形態にかかわらず，効率的かつ短期に開発し，安定稼働させる重要性が高まっている。

AFSではこれらの課題を解決するために，富士通がこれまで培った基幹系システムの構築技術を実行基盤に取り込み，フレームワークとして提供する。これにより，分散システムをベースとした基幹業務システムのメインフレームレベルの業務安定性，堅牢（ろう）性を実現し，基幹システムのオープン化における，短期開発，全体最適化が可能となる。アプリケーション連携フレームワークが提供する機能を**図**-6に示す。

以降では，オンラインアプリケーションの非同期業務を短期間で構築するために，アプリケーション連携フレームワークが提供する五つの機能について述べる。

● **オンライン非同期処理実行基盤**

従来非同期のアプリケーションプログラムで記述していた，メッセージの受信（メッセージキューからメッセージの取出し），メッセージの送信（メッセージキューからメッセージの書出し処理）などの非同期特有の処理を実行基盤として提供する。アプリケーションの開発者は，業務ロジックの開発に集中でき，開発の効率化が促進される。

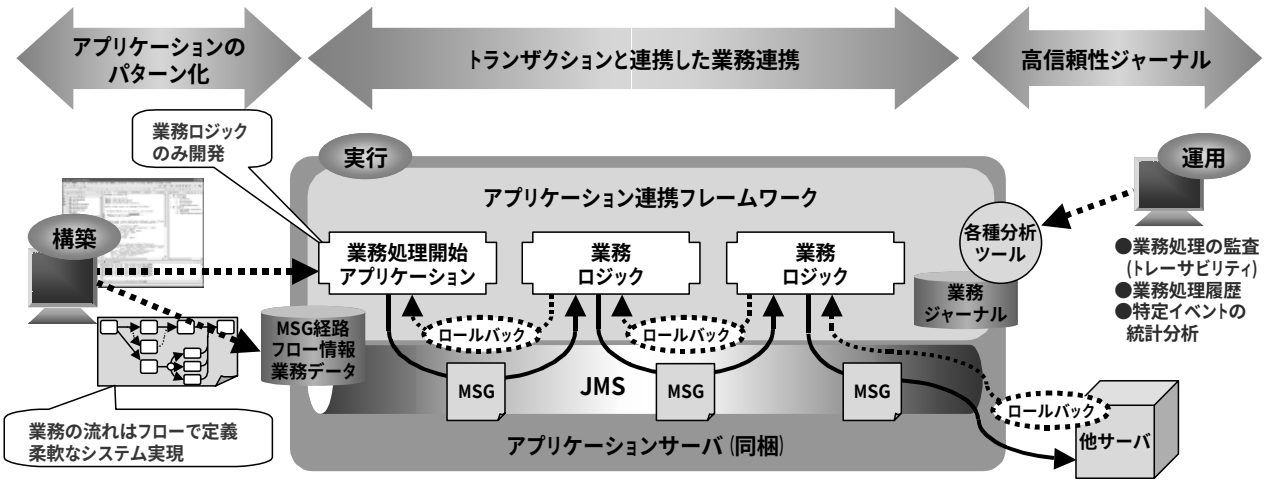

図-6　アプリケーション連携フレームワークの提供機能
Fig.6-Functions about Application Connecting Framework.

● アプリケーションのルーティング制御

処理フローを対話的に定義するツールの提供によりあらかじめ定義された順序に従ってメッセージをルーティングし，自動的にアプリケーションを呼び出す機能を提供する。複数アプリケーションから成る業務（例えば，貸付業務の与信チェック，貸付）を開発する場合に特に有効となる。また，アプリケーションの異常発生時に実行した処理の逆順でリカバリのための処理業務を定義しておくことにより，リカバリ処理を簡単に開発することができる。このため，リカバリ手順に変更が発生しても定義のみの変更で対応が可能となり対応コストの削減に効果がある。

● メッセージの一貫性保証，順序性保証

アプリケーションへの処理要求や処理結果を送信するメッセージと，アプリケーション内部でアクセスする業務データとの一貫性を保証する機能を提供している。アプリケーションが正常終了したときには，送信メッセージや業務データはコミットされ，異常終了したときには，ロールバックされる。また，送信されたメッセージが送信された順序で確実にアプリケーションに渡される機能を提供している。異常発生時のデータの欠落，データ矛盾および処理結果矛盾が発生しないように実行環境が制御するため信頼性の高い業務アプリケーションを低コストで開発することが可能となる。

● メッセージキューの稼働監視，流量制御

業務アプリケーションの異常や処理遅延などの異常を早期に検出するために，メッセージキューの稼働監視機能を提供する。また，異常時には，別のメッセージキューに自動的に振り向ける機能や，メッセージキューに格納されているメッセージの滞留率をもとにメッセージの付加分散を制御する機能を提供する。これらの機能を使用することにより，システム高負荷時の業務停止や業務の遅延を事前に回避することができ，信頼性の高いシステムの構築が可能となる。

● システム運用コストの低減

課金や障害時のデータ追跡などのために，アプリケーションの処理状況（ユーザログ）を記録するためのJavaおよびCのAPI（Application Program Interface）を提供する。アプリケーションがこのAPIを使用することによりユーザログを簡単に，かつ確実に取得することが可能となる。APIでの指定によりユーザログと業務データとの一貫性をとることも可能である。複数アプリケーションの呼出しから応答までの時間などの性能統計情報を業務単位で採取，管理する機能を提供する。この機能を活用することによりボトルネックを早期に検出し，業務システムの改善に利用することができる。

む　す　び

本稿では，システムのWeb化によって注目され始めたリッチクライアントのためのフレームワークであるCJFと，ミッションクリティカル対応を主幹に様々な機能拡充を行ったAFSについて紹介した。

今後，CJFはほかのフレームワーク製品との連携や開発環境の強化を図っていく。またMDA（Model Driven Architecture）などの新技術の取込みも積極的に行い，時代のニーズを満たす製品として更なる適用拡大を目指す。

基幹系業務の構築では，今後SOA（Service Oriented Architecture）を使用した手法へと推移していくが，今回提供するAFSのミッションクリティカル機能はこのSOAのインフラ機構と位置付けられ，今後SOAの各サービスは本機能上に提供されていく予定である。

参 考 文 献

(1) 株式会社野村総合研究所：NEWS RELEASE. 次世代の企業システムクライアントが検討段階から導入段階へ～リッチ・クライアントに関するユーザ意識調査結果で判明～. 2004年5月27日.

(2) C. McClanahan et al.：JavaServer™ Faces Specification Version 1.0, Revision 01. Sun Microsystems, Inc., February 2004.