

フレームワークによる開発作業の効率化

Promotion of Development Efficiency by Using Frameworks

あらまし

SDASの開発期間短縮，開発効率・品質向上の中心技術がフレームワークである。富士通では，アプリケーションフレームワーク体系として次の三つのフレームワークから成るB².Sframeworkを提供している。Interstage Application Framework Suite (IAFS)は，Webアプリケーションのフレームワークである。IAFSを使うと，「画面」「業務ロジック」「データ」が分離されるため，開発や保守を効率化できる。また，Webアプリケーションのぜい弱性対策も行える。Interstage Business Application Server (IBAS)は，基幹システムのためのフレームワークである。IBASではオンライン業務向けフレームワークとバッチ業務向けフレームワークの機能を提供することで基幹システムの効率的構築を実現する。Client J Framework (CJF)は，リッチクライアントシステムをJavaで構築するためのフレームワークである。

本稿では，これら三つのフレームワークの最新機能や事例を中心に紹介する。

Abstract

The framework is the central SDAS technology for reducing the time needed to develop a system and improve the development efficiency and quality. Fujitsu provides the B².Sframework, which consists of three frameworks, as the application framework system. The first framework, Interstage Application Framework Suite (IAFS), is a Web application framework that can make development and maintenance more efficient because it separates screens, business logic, and data. It can also improve the security of Web applications. The second, Interstage Business Application Server (IBAS), is a framework for mission-critical systems. IBAS ensures efficient construction of mission-critical systems by providing a multi-language container and batch framework functions. The third framework, Client J Framework (CJF), is used for establishing a rich client system by using Java. This paper focuses on the latest facilities for using these frameworks and some examples of how these frameworks are applied.



車井 登(くるまい のぼる)
ミドルウェアプラットフォーム事業
部第三開発部 所属
現在，フレームワークの開発に
従事。



花森利弥(はなもり としや)
ミドルウェアプラットフォーム事業
部第三開発部 所属
現在，基幹システム向けのフレーム
ワークの開発に従事。



島 隆史(しま たかし)
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。

まえがき

オープンシステムにおいては、ミッションクリティカルな領域に適用するメインフレームのAIM (Advanced Information Manager) のようなモデルウェアは存在しない。そのため、業務アプリケーションにおいて、業務処理に加えミッションクリティカルな制御処理を行う必要がある。これを改善するため、制御処理の共通部品、簡易な制御機能、およびこれらをフレームワークという体系にまとめたものを個々に作成したり、ベンダ製やオープンソースソフトウェアのフレームワークを使ったりしている。

富士通はアプリケーションフレームワーク体系としてB².Sframeworkを提供している。B².Sframeworkは、つぎの三つの製品から成る。

- (1) Interstage Application Framework Suite (以下、IAFS): Webアプリケーションフレームワーク
- (2) Interstage Business Application Server (以下、IBAS): 基幹システムフレームワーク
- (3) Client J Framework (以下、CJF): Javaリッチクライアントシステム構築フレームワーク

これらの製品については、本誌2005年1月号 (TRIOLE特集)⁽¹⁾で既に紹介しているので、本稿では最新の機能や事例を中心に紹介する。

Interstage Application Framework Suite (IAFS)

IAFSはJ2EEに準拠したWebアプリケーションを開発するためのフレームワーク製品である。

IAFSを利用したWebアプリケーションは、データ、画面、業務ロジックの三つの要素から構成され、開発者は、データを規定した後に画面および業務ロジックを設計する。このとき、画面と業務ロジックはデータから分離されるため、一方の修正に対する他方への影響を最小限にすることができる。その結果、拡張性や保守性の高いWebアプリケーションを開発することができる。そのほか、IAFSではEJB/Webサービスのためのフレームワーク、WebサーバとAPサーバ間のログを採取する機能などシステムを構築する上で有用な機能を提供している。

これまでIAFSはWebアプリケーションの開発効

率化や保守性を追求してきた。一方で、Webアプリケーション開発においてはぜい弱性対策への関心が高まっている⁽²⁾。本特集掲載の論文「開発プロセスの標準化とWebアプリケーション開発への対応」の「セキュアWebアプリケーション」の章でも述べられているように、ぜい弱性対策はシステム開発のコスト増大につながるため、事前検証されたセキュアなフレームワークの適用が望まれている。本章では、まずWebアプリケーションのぜい弱性について述べた後、IAFSが提供するセキュリティ機能について説明する。

Webアプリケーションのぜい弱性

Webアプリケーションのぜい弱性としては未検証の入力値によるものや、クロスサイトスクリプティング (以下、XSS)、セッション管理、クロスサイトリクエストフォージェリ (以下、CSRF) などが挙げられる。

とくに入力値の検証はThe Open Web Application Security Project (OWASP) でも最も重大なぜい弱性として挙げられている⁽³⁾。以下、それぞれのぜい弱性について概要を説明する。

(1) 未検証の入力値

クライアントからの送信パラメータを検証せずに利用することでシステム内部のコンポーネントなどが攻撃されてしまうぜい弱性である。SQLインジェクションはこのぜい弱性に対する攻撃方法である。このぜい弱性を防止するためにはサーバ側ですべての入力値を検証し、不当な値を取り除く必要がある。

(2) XSS

フォームに入力されたデータをそのままWebブラウザに表示した結果、悪意のあるスクリプトなどが実行されてしまうぜい弱性である。これを防止するには、画面に文字列を表示するときに危険な文字を置換・除去する必要がある。この処理をサニタイジングと呼ぶ。

(3) セッション管理

ログイン認証を含むWebアプリケーションの場合、ログイン済みのセッションIDを第三者に知られてしまうと、Webサイトを攻撃される恐れがある。これを防ぐ一つの方法は、ログイン時に新しいセッションIDに付け替えることである。また、ログアウト後に不要なセッションを残すと悪用される恐れがあるため、速やかに破棄する必要がある。

(4) CSRF

CSRFとは、悪意のあるWebサイトを閲覧した第三者が、意図せずに別のサイトに対してリクエストを送信する攻撃である。その結果、知らない間にオンラインショッピングなどで買い物をしているという結果になる。これを防ぐ一つの方法は、画面遷移の正当性を確認するということである。意図しない画面（Webサイト）から送信されたリクエストをアプリケーションがブロックすることで攻撃を防ぐことが可能である。

IAFSのセキュリティ機能

前節で述べた代表的なWebアプリケーションのぜい弱性は、アプリケーションの作り込みによって回避することが可能であるが、システムごとに関係・検証を行うことは開発コストの増大を招くことになる。そこで、IAFSではこれらのぜい弱性に対応するための機能をフレームワークとして提供する。以下で、IAFSが提供するセキュリティ機能について説明する。

(1) 入力値の検証

IAFSではサーバ側で入力値を検証するための機能を提供する。入力値を検証するには、検証ルールをXML形式で記述し、業務ロジックから専用のAPI（Application Program Interface）で呼び出す。このXML形式の検証ルールは入力値の変換（int, long, stringなど）や組み込みの検証ロジックを持っているほか、入力値の相関チェックも行うことができる。つぎに検証ロジックの例を示す。

- ・2値間のチェック（大小比較、一致など）
- ・文字列長のチェック
- ・列挙値との比較
- ・正規表現によるチェック

また、ユーザ定義の検証ロジックをJavaで記述して組み込むことも可能である。

(2) サニタイジング

IAFSではUJIタグと呼ぶJSP拡張タグを提供している。UJIタグは、フレームワークを制御するものと、画面部品として利用するものの2種に大別することができる。画面部品はすべてサニタイジングに対応しており、HTMLで危険とされている五つの文字（&, <, >, ", '）をエスケープして表示する。この結果、画面部品でスクリプトが出力されても機能しなくなる。

画面部品のUJIタグを表-1に示す。

(3) セッションIDの更新

先に説明したようにログイン済みのセッションIDを第三者に知られないようにする一つの方法は、ログイン時に新しいセッションIDに付け替えることである。通常、セッションIDを付け替えるには以下の手順が必要である。

- ・セッションに関連付けられているすべてのオブジェクトを退避
- ・セッションを破棄
- ・新しいセッションを生成
- ・退避しておいたオブジェクトをすべて新しいセッションに再設定

IAFSでは、上記の手順を一括して行うためのAPIを提供する。このAPIを利用することでアプリケーション固有に定義されたセッションスコープのオブジェクトを自動的に新しいセッションに引き継ぐことが可能となる。また、ログアウト時などにセッションオブジェクトを破棄する機能も提供する。この機能を利用することで、セッション終了後の不要なリソースを開放するとともに、第三者がセッションを悪用する危険を抑えることが可能である。

(4) リクエストの正当性検証

IAFSではリクエストの正当性を検証するために、フレームワークが画面遷移の状態を管理する。不当な画面からリクエストが送信されるとアプリケー

表-1 画面部品のUJIタグ

基本タグ	文字列表示
	リソース
コンポーネントタグ (19種)	文字列入力
	整数入力
	日付入力
	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
	アンカー
	ラベル など
画面部品タグ	テーブル
	ツリー
	リスト

ションに対して例外が通知され、エラー画面への遷移や、ログアウト処理など任意の処理を実行することが可能である。

本節ではIAFSが提供するセキュリティ機能について説明した。IAFSを利用することで基本的なWebアプリケーションのぜい弱性を防ぐことができ、開発者はアプリケーション固有のぜい弱性対策に注力することが可能である。

Interstage Business Application Server (IBAS)

従来、メインフレーム中心であった基幹システムもインターネット技術をはじめとするオープン技術の進展により周辺業務だけでなく、コア業務もオープン化、分散化が進んでいる。また、基幹業務のオープン化の流れは、オンライン処理中心に始まり、それに伴いデータベースがオープン化され、分散したシステムの連携、オンライン処理やデータベースに付随する一括処理のオープン化へと進展している。

このような環境において、情報システム部門では、システム構築・運用に関しては、

- (1) システム開発の効率向上/スピードアップ/安定稼働、
 - (2) 柔軟なシステム拡張/変更、
 - (3) TCO削減
- を最重要課題と認識している。

IBASは、オープンプラットフォームを活用した基幹系システム（オンライン、バッチ業務）構築に

必要な共通技術の提供、基幹システムのアプリケーション構造の共通化と連携技術の提供、および業務運用を支援する機能を提供することで、生産性・堅ろう性の高い、柔軟性・拡張性に富むシステム構築と運用を支援するソフトウェア製品である。

基幹システムフレームワークの概要を図-1に示す。

IBASでは、オンライン業務（同期、非同期）およびバッチ業務構築のために以下の機能を提供している。以下では、代表的な機能について説明する。

オンライン業務向けフレームワーク

(1) 基幹システム構築に必要な共通技術の提供

アプリケーションは、制御ロジックと業務ロジックに大別できる。制御ロジックとは、オンライン処理におけるアプリケーションの実行制御、トランザクションの管理、データベースのアクセス制御など処理形態に依存しない共通制御、および非同期処理のキュー制御やバッチ処理における資源の事前割当てなど処理形態ごとの固有制御に分類され、アプリケーション間の連携や資源管理を行うためのプログラムを意味する。また、業務ロジックとは、顧客情報や注文内容をデータベースに反映するなど業務に特化した制御を行うプログラムを意味する。

しかし、制御ロジックには、高い信頼性と品質が要求され、設計および開発に時間とコストがかかる。

IBASにおいては、制御ロジックと業務ロジックを分離して開発し、また、制御ロジックをIBASが実行環境として提供することで、アプリケーション

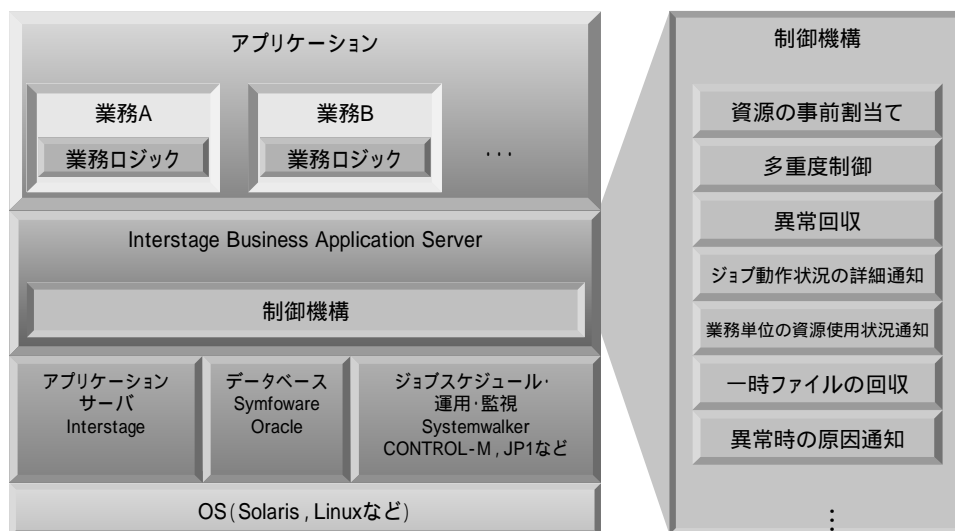


図-1 基幹システムフレームワークの概要
Fig.1-Outline of mission-critical system framework.

の生産性を向上し、信頼性のあるシステムを早期構築できるようにしている。

・アプリケーションの実行制御

アプリケーションが起動された場合に、そのアプリケーション内に存在する業務ロジックを一括してローディングする機能（プレロード機能と呼ぶ）と、業務ロジックへの処理要求時に、該当の業務ロジックをローディングし、処理終了時にアンロードする機能を提供する（要求時ロード機能と呼ぶ）。これらにより、高速なレスポンスの保証やメモリ資源の有効活用などアプリケーションの利用形態に応じてシステム構築が行える。

・データベース処理の簡易化制御

データベースを利用したアプリケーション向けに、データベースコネクションの事前接続やコネクションが切断された場合に再接続する機能により、データベースへのアクセスの高速化やコネクション異常時の処理の簡易化が可能となる。

また、業務ロジックの処理結果（終了コード）により、データベースに対してコミットやロールバックを行うトランザクション制御機能により、アプリケーションで、開始や終了といったトランザクションの考慮が不要となる。

・非同期処理でのキューとデータベースの一貫性保証

アプリケーション間で、非同期処理により連携する場合は、送り手がキューにメッセージを書き込み、受け手がキューから取り出す形態になっている。また、アプリケーションがデータベースにアクセスを行う場合、キューからのメッセージの取出しとデータベースへのアクセスに対して、一貫性を保証する必要がある。一貫性を保証しないと、データベースへの更新にエラーが発生したにもかかわらず、キューには処理要求のメッセージが削除されるような問題が発生する。また、これらの処理の開発には時間とコストがかかる。IBASでは、すでに説明したようにキューにアクセスするような処理については、制御ロジックとして提供している。また、キューのメッセージをデータベースに格納する機能を提供することで、トランザクション制御機能により、キューへのアクセスとデータベースへのアクセスの一貫性を保証した。

・非同期処理でのエラー処理

キューからメッセージを取り出しアプリケーションで処理を行うが、アプリケーションの様々な処理異常により、当該メッセージがキューに滞留し続け、後続の業務処理に影響を与えることがある。

IBASでは、異常メッセージの退避機能により、自動的に異常メッセージを別のキューに退避させ、問題の局所化を行うことができる。

(2) アプリケーション構造の共通化と連携技術

企業を取り巻く市場や需要の急激な変化や多様な顧客ニーズに即応するためには、柔軟性および拡張性の高いシステムを構築する必要がある。

IBASでは、制御ロジックと分離した業務ロジックの構造を規定し、プラットフォーム、ミドルウェアおよびオンライン形態（同期処理/非同期処理）に依存しない業務部品として汎用性と再利用性を高め、また、異なる開発言語（C言語、COBOL、Java）でも、同じ設計方法で業務ロジックの開発を可能とした。

さらに、オンラインの同期処理に対しては、クライアントとサーバのアプリケーション間の文字コード変換やデータ型変換機能を提供し、異なる言語でのクライアントとサーバのアプリケーションの開発を容易にした。

非同期処理に対しては、業務アプリケーションの呼出し順序、受け渡すメッセージ、および異常が発生した場合のシーケンスを定義するフローツールを提供した。これにより、業務ロジックの柔軟な組合せにより新サービスを簡単に構築することを可能とした。

(3) 運用支援機能の提供

オープンシステムは、Webサーバ、APサーバおよびDBサーバの機能分担が進み、また、これらの機能をサーバに分散してシステムの構築が行われている。業務量拡大への柔軟な対応や信頼性向上のメリットはあるものの、複数サーバや業務管理の煩雑さから管理者の運用コストが増大してきている。

IBASでは、アプリケーションの呼出しから応答までの時間、パラメタおよびエラー内容などの稼働状況を利用者の識別情報を自動的に付加して各サーバ単位に出力するロギング機能を提供する。

出力したログは、各種のツールにより容易に分析が可能である。

また、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APIを提供する。このAPIを使用することにより、ユーザログを簡単、かつ確実に取得することが可能となる。

これらの機能により、業務の可視化や性能のボトルネックの早期検出が可能となり、性能チューニングやキャパシティプランニングなどの運用コストを削減できる。

バッチ業務向けフレームワーク

(1) 基幹システム構築に必要な共通技術の提供

ジョブが使用する資源を事前割当てし、処理遅延要因をジョブ開始時に集約することで、以下のような事象を抑え、ターンアラウンドタイムを安定化する。

- ・ジョブ実行途中の資源待ちの完了遅延
- ・ジョブ後半に資源不足によるエラー終了
- ・デッドロック発生によるジョブ停滞

また、CKRM（Class-based Kernel Resource Management）などの資源管理ソフトウェアと連携し、以下の機能をサポートすることで単一サーバであっても大量のジョブを安定的に走行させることが可能となる。

- ・ジョブの特性による分類
- ・実行クラスごとの多重度制御
- ・実行クラスごとの資源割当て

(2) アプリケーション構造の共通化と連携技術

ファイル割当てや異常回収処理などの制御はフレームワークが行うため、業務プログラムと使用する資源を定義するだけで簡単にバッチジョブが作成でき、制御アプリケーションの作成は不要となる。

そのため、バッチフレームワークを適用すると、シェル/定義ステップと比較して開発生産性が約5～10倍向上する。

さらに、異なる言語（COBOL、C言語、Java、そのほかユーティリティ）のプログラムも使用でき、既存資産やノウハウを有効活用できる。

(3) 運用支援機能の提供

従来、運行管理のミドルウェアを使用しているジョブ内の処理ステップの動作状況や資源の利用状況を把握することはできなかったが、バッチフレームワークを適用することで以下の情報をリアルタイムに取得することが可能となる。

- ・処理ステップごとの動作状況（Elapse timeなど）
- ・処理ステップごとのファイル利用状況

また、業務の稼働分析・キャパシティプランニングを行う際、従来、資源使用状況をプロセスやサーバ単位に集計できても業務とプロセスの対応を別途管理する必要があるため、業務単位に集計することは困難であった。バッチフレームワークでは、以下の機能を提供することで本課題を解決する。

- ・ジョブ単位の資源の使用状況ログの取得
- ・資源使用状況ログの業務単位の集計およびビジュアルな分析ツール（Systemwalker連携）

また、ジョブが異常終了した場合、どこまで処理を実行したのか、異常の原因は何かを特定するのに時間がかかり、ジョブの再実行までに時間を要していた。

この問題を解決するために、ジョブ単位の詳細な処理結果を通知し、リカバリが完了するまでの時間を短縮するための以下の機能を提供する。

- ・実行したステップ、使用した資源、ジョブの異常終了の原因を出力するジョブログ
- ・異常が発生したステップからの再開機能

また、従来のバッチ処理でアプリケーションやシェルが異常終了した場合、未回収の資源の累積によるトラブルを避けるため、定期的に未回収資源の削除をシステム運用として実施する必要があった。

バッチフレームワークでは、ファイルの要・不要の定義に基づき、不要資源を完全に削除する機能を提供する。このため、異常終了に対するシステム運用の負担を軽減することが可能となる。

Client J Framework (CJF)

近年、社内基幹系業務システムの構築にリッチクライアントを導入するニーズが高まっている。野村総合研究所の顧客アンケート⁽⁴⁾では、リッチクライアントの割合は、2004年の約13%から2年後には約2倍の28%になるという調査結果が出ている。また現行ファットクライアントからリッチクライアントに移行したいシステムのうち、社内基幹系業務システムが33%を占めている。このような背景のもと、ソフトウェア製品市場においては、リッチクライアントに関連した製品や技術が、数多く登場してきている。

また、今後採用したい技術としては、Javaで構

築を検討しているユーザが多い。2004年度と2005年度（9月末）までに、富士通の対応したシステムインテグレーション商談約1,200件を調査したところ、Web系システムを対象としているユーザは、2005年度で52%（前年度比8%増）に上る。図-2に示すようにこのうち27%（前年度比2%増）がクライアント技術にJava（AppletまたはJavaアプリケーション）を採用することを検討している。

このようにクライアントシステムの構築においてニーズの高いJavaであるが、現状では設計手法が未成熟である。とくに基幹系業務システムを構築する場合、非同期制御の実装や性能確保のための高い専門知識や技術が必要という課題がある。

このような課題を解決するために、富士通ではリッチクライアントソリューションとしてJavaをベースとしたフレームワーク、CJFを提供している。CJFを適用してリッチクライアントシステムを構築すると、開発規模・開発工数を大幅に削減できる。

以下、CJFの概要と適用業務事例を紹介する。

CJFとは

CJFは、Javaでリッチクライアント向け業務アプリケーションを構築するためのフレームワークである。CJFでは、一からシステムを開発する場合、難度の高い制御機能を中心に、開発を容易にする機能を提供している。CJFは、画面、業務ロジックおよび制御ロジックを明確に分離した開発手法を採用している。利用者は画面、イベントに応じた業務ロジック、および外部定義情報を作成する。以下にCJFの主要な機能分類を示す。

- (1) システム制御機能
- (2) イベント制御機能

- (3) 画面制御機能
- (4) データ領域管理機能
- (5) デバイス制御機能
- (6) GUI（Graphical User Interface）部品データ機能
- (7) メッセージ制御機能
- (8) ユーティリティ機能

CJFは2004年6月にリリース後、複数ユーザのシステム構築に適用されている。その後、プロジェクト適用での要望機能などを取り込み、機能強化版のV1L30を2005年4月にリリースした。V1L30では、マルチウィンドウ機能の追加、画面遷移機能の強化、フォーカス遷移機能の強化を実施した。このほかに、MDA（Model Driven Architecture）の技術を採用し、画面遷移定義を自動生成する機能の追加を予定している。

適用業務事例

CJFは、業種を問わずエン트리系の基幹業務の開発に幅広く適用されている。また、リッチクライアントの利点を活用するソリューションビジネスにも採用されている。現在のシステム形態における主なニーズと、CJFでの実際の適用事例の関係を図-3に示す。CJFは、エン트리系の業務でスピーディな操作を必要とする社内基幹業務システムに適用するケースが多い。

(1) 金融系での適用事例

金融系では、営業店の店舗システムにおいて高機能GUI、多様な画面遷移（図-4）を必要とする業務にCJFを適用している。

ある金融系システム開発の生産性データについて

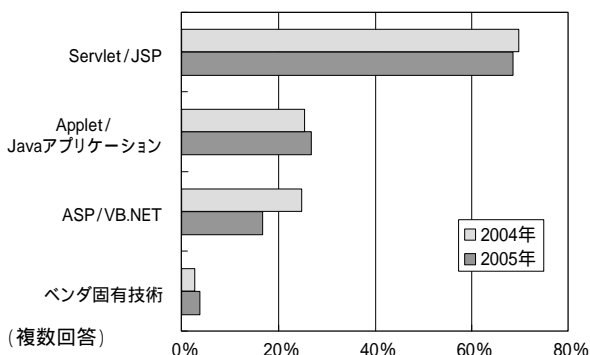


図-2 Web系システムのクライアント技術の候補
Fig.2-Candidate of client technology in Web system.

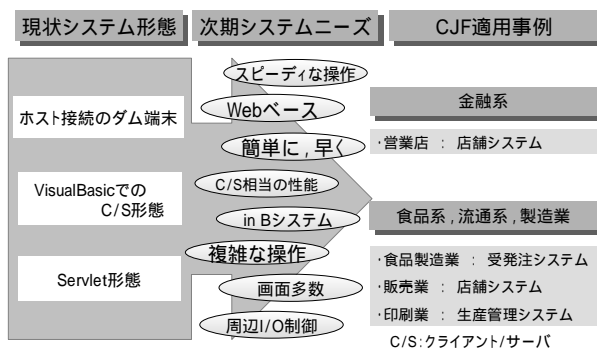


図-3 現行システム形態と適用事例
Fig.3-Architecture of present system and application experience.

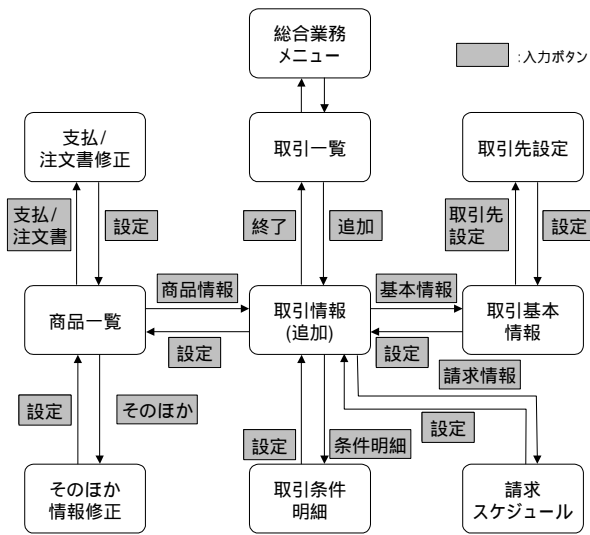


図-4 画面遷移の例
Fig.4-Example of screen transition.



図-5 画面構成の例
Fig.5-Example of screen composition.

表-2 適用効果

比較項目	CJF未適用	CJF適用	削減効果
開発規模	70.0 kstep	54.1 kstep	- 23%
開発工数	128人月	79人月	- 38%
開発期間	18箇月	15箇月	- 17%
総バグ数	960件	693件	- 28%

分析した結果、CJFを適用した場合の開発工数について38%の削減効果があった(表-2)。CJFを適用した際の生産性データは、実際の開発でのデータを採取した。CJFを適用しなかった場合の生産性データは、過去の同様の開発実績をもとに算出した。適用効果の要因は、難度が高く性能の配慮を必要とする制御ロジックをCJFが用意していることで、煩雑なイベント処理などから解放され、業務ロジックの実装に専念できたことによる。

今回の比較対象は、ある業種内で共通に使用できる共通基盤部分の開発である。方式設計・構造設計の難度が、一般の業務ロジックに比べて高いにもかかわらず、CJF適用による削減効果があった。一般の業務ロジック部分の開発では、さらに削減効果が期待できる。

(2) 製造業・食品系での適用事例

製造業・食品系では、受発注・生産管理・店舗システムにおいて複雑な画面構成(図-5)、PFキーの多用や周辺機の制御を必要とする業務にCJFを適用している。

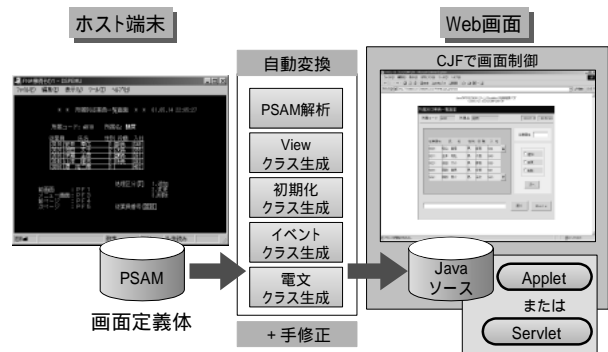


図-6 TransMigrationサービスでのCJF適用例
Fig.6-Example of applying CJF in TransMigration service.

現在、流通系・官公庁においてもリッチクライアントのニーズがあり、CJFの適用を検討している。

(3) ソリューションビジネスでの適用事例

各種ソリューションビジネスにおいてもCJFの適用事例が増加している。本特集で紹介している「TransMigrationサービス」では、ホスト端末の画面をWeb画面に移行する際に、PSAM (Presentation Service Access Method) などの画面定義体からJavaクラスを自動生成する機能を提供している。そのWeb画面の制御層にCJFを適用した例を図-6に示す。また、イメージエントリのパッケージ製品では、多彩なイメージ表示制御を実現するためにCJFを適用している。

む す び

本稿では、Webベースのシステム構築に必須なアプリケーションフレームワークであるIAFS、IBAS、CJFについて紹介した。

フレームワークを適用した開発は、最初の導入に習熟時間が必要であるが、2回・3回と繰り返し使うことで、学習による生産性向上や部品群の整備などのノウハウが蓄積され、開発生産性が必ず向上するものである。したがって、フレームワークを使っていない開発者には、是非使ってほしい技術である。

富士通は、これらのフレームワーク製品を自社のシステム構築サービスで使い、その結果を随時フィードバックしていくとともに、最新の技術やWebアプリケーションのぜい弱性に対応していくことで、一層の高品質で高生産性のシステム開発の基

盤となるフレームワークに仕上げていく予定である。

参 考 文 献

- (1) 小川俊雄ほか：スピード時代のシステム構築基盤．*FUJITSU*，Vol.56，No.1，p.22-30（2005）．
- (2) 高木浩光：Webアプリケーションにおける脆弱性．*情報処理*，Vol46，No.6，通巻484号，p.636-642（2005）．
- (3) OWASP：Top Ten Most Critical Web Application Security Vulnerabilities．
<http://www.owasp.org/documentation/topten.html>
- (4) 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．
<http://www.nri.co.jp/news/2004/040527.html>

