

# オブジェクト指向技術で変化に対応する ALIVE Solutionシリーズ 就業システム

田中隆治  
大和田政嗣  
加藤嘉之  
安川武史

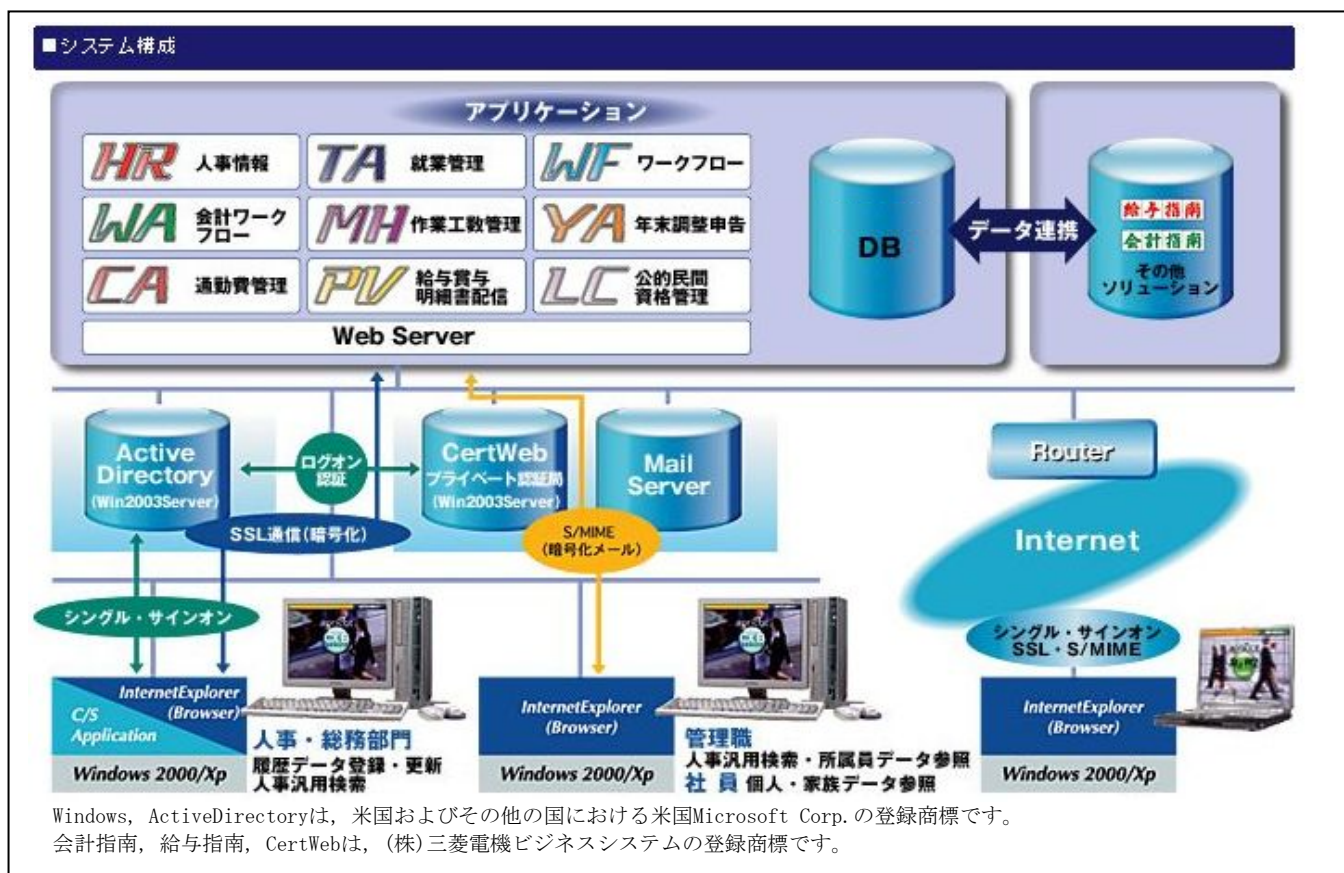
## 要旨

(株)三菱電機ビジネスシステム (MB) では、就業(TA)、ワークフロー(WF)などから成る Web システム”総務・人事・経理トータルシステム ALIVE Solution シリーズ”を開発・販売している。

この度、この ALIVE Solution シリーズを刷新するに際し、最新プラットフォーム対応、対応業種の拡大、保守性・拡張性の向上のため、Java 言語とオブジェクト指向技術を積極的に取り入れた。特に、コンポーネント化とフレームワーク、DI コンテナといった技術を積極的に導入し、基盤のフレームワークとして、オープンソースソフトウェアの Spring Framework と MB が開発した Java フレームワーク “radish” を採用して、柔軟なシステム化を可能にした。

就業システム(TA)の開発では、メニュー、ログイン・認証、ログ出力などを、ALIVE Solution シリーズで共通して利用できるように設計すると共に、就業システム(TA)に特化する機能に関しても、カスタマイズによる機能拡張が行い易い設計を心掛けた。

これらの改善により、顧客から寄せられた様々な要望にこれまで以上に素早く応え、効率よくシステムを提供できると確信している。今後は、ワークフロー(WF)なども順次バージョンアップして、更なる顧客の業務改善、TCO 削減に貢献したい。今後もこれらの改善を継続して、継続的に保守性・拡張性を向上させることが、顧客である企業の発展に寄与するものと考えている。



## ALIVE Solutionシリーズのシステム構成図

ALIVE Solution シリーズは、Web システムの各サブシステムが連携して、人事・総務部門のTCO削減に貢献している。また、MB製品であるCertWebなどと連携したセキュアな環境構築や、業務パッケージの指南シリーズと連携して、更に効率的に業務を行うことが出来るシステムである。

## 1. ま え が き

ALIVE Solutionシリーズは、Webアプリケーションを中心としたパッケージシステムで、2000年にMBの業務システムとして誕生し、2002年から広く一般向けに販売を開始した。販売開始後も、社内や顧客のニーズを反映して、様々な機能追加やサブシステムの拡充を図ってきた。

しかし、パッケージシステムとして更に広範囲な業種・業態のサポートと顧客対応の機能追加・変更（カスタマイズ）を容易に行うため、オブジェクト指向技術を利用してシステムの再構築を行うことになった。

本稿では、ALIVE Solutionシリーズの中核パッケージである就業システム(TA)の開発について述べる。

## 2. ALIVE Solutionシリーズと就業システム

### 2.1 ALIVE Solutionとは

ALIVE Solutionシリーズは、図1に示すように、総務・人事・経理の業務をトータルにサポートするシステムである。施策立案や経営の目標設定・評価などの戦略企画機能を支援するとともに、ファジィで非定型な総務・人事系業務の効率化も最新Webアプリケーション技術の活用により実現した。サブシステムである就業(TA)、ワークフロー(WF)、Web会計(WA)、通勤費管理(CA)、年末調整申告(YA)、作業工程管理(MH)などが相互に連動して効果的な業務改革でTCO削減に貢献する。



図1. ALIVE Solutionシリーズの構成

### 2.2 製品の特長

ALIVE Solutionシリーズの製品には、以下の特長がある。

- ①実際の業務に精通した人事・総務部門の要求を最大限に反映させた、使い勝手を考慮したシステム
- ②社員への積極的情報開示
- ③セキュリティ機能による安全な情報公開とプライバシーの保護

- ④企業グループ内の複数企業での利用を考慮し、シェアードサービスを実現

## 2.3 就業システムの特長

日々の就業実績をWebブラウザから入力することで、就業情報をリアルタイムに管理でき、人事・総務部門の負担を軽減するWebアプリケーションである。Webブラウザを使った容易な操作に加え、入力ミスや計算ミスのチェック機能も備え、入力作業者の負担を軽減する。更に、『給与指南』をはじめとする「給与計算システム」との連動で、給与計算担当者の負担も大幅に軽減される。

## 3. 従来システムの課題

### 3.1 機能追加・変更の煩雑さ

就業システムでは、顧客のニーズに応じたカスタマイズが頻繁に行われている。このようなニーズに素早く柔軟に対応するために、既存のカスタマイズ事例を活用できる仕組みが望まれている。

例えば、A社で指紋認証のカスタマイズを、B社で一括承認のカスタマイズがあり、C社では両方のカスタマイズが必要な場合、従来はモジュールの依存関係が大きいいため、A社とB社のソースコードを差し替えるだけでは正常に動作しないことが多かった。このため、こうした制約を回避できる仕組みが望まれている。

### 3.2 保守の困難さ

パッケージシステムは、機能追加や不具合対応のため常にプログラム改訂を行っている。カスタマイズしたシステムには、担当したSEがこれらの改訂情報を個別に手作業で適用しており、煩雑な作業になっている。このため、カスタマイズされたシステムに対しても改訂情報を適切に反映する仕組みが望まれている。

また、前述のように機能の差し替えが簡単にできれば、パッケージの中核は基本機能だけを提供し、オプション機能は必要に応じて組み合わせが可能になるので、プログラムがシンプルになって保守性の向上が期待できる。

### 3.3 新たな機能要件の対応

過去に顧客から挙げられた要望事項や労働基準法改正に対応するため、次のような機能追加も求められている。

- ・多彩な勤怠データ入力方式に対応
- ・各勤務形態（通常勤務者・交替勤務者・フレックス勤務者・パート/アルバイト）に対応
- ・各種勤怠申請・命令機能
- ・36協定（労働基準法第36条：時間外・休日労働に関する労使協定届）管理機能
- ・シフト勤務管理機能
- ・小売・流通業対応
- ・作業工数管理機能
- ・休日・時間外計算機能

## 4. 新システムのアーキテクチャ

3章で述べた課題を解決するために、コンポーネント化、フレームワークといったオブジェクト指向技術を導入して開発を行った。

### 4.1 コンポーネント化とDI技術の導入

多様なシステム要件に対応するため、柔軟なカスタマイズへの対応が要求される。このため、主にオブジェクト指向技術のフレームワークとコンポーネント化を取り入れることとした。

コンポーネント連携には、コンポーネント間の独立性をより高め再利用を容易にする、DI(Dependency Injection)技術<sup>(1)</sup>を導入した。また、このDI技術を実現する基盤のフレームワークには、オープンソースソフトウェア(OSS)の“Spring Framework”を採用した。

以上の技術を用いて、次のような効果を狙っている。

- ① コンポーネント単位で機能の追加・変更ができる
- ② 不具合の影響を局所化し易い
- ③ コンポーネントがシンプルで開発し易い
- ④ コンポーネントの差し替えにより保守し易い
- ⑤ 開発段階ではクラス単体テストが可能になる

### 4.2 システム構成

就業(TA)システムにおける、コンポーネントとフレームワークの関係を図2に示す。

コンポーネント化されたアプリケーションと役割別に階層化されたフレームワークで構成される。

### 4.3 フレームワークの階層化

Spring Frameworkを基盤とし、Webシステムに特有な処理やシステムの共通処理を実現するため、役割別にフレー

ムワークを階層化して処理の最適化を図った。

#### 4.3.1 Spring Framework

DIコンテナの実装には様々な製品があるが、DI提唱者ロッド・ジョンソンが実装したSpring Frameworkは、最も早くから普及し、世界中で広く利用されている。また、実行時に機能を挿入するAOP(Aspect Oriented Programming)もサポートされており、認証チェック、アプリケーションのログ出力で利用している。

DI技術には、共通処理の隠蔽とプログラム構造の統一を促す効果もある。その結果、コンポーネントがシンプルでカスタマイズし易いプログラム構造になる。

また、コンポーネントの追加や差し替えに関しても、XML形式の定義ファイルを変更するだけで、簡単に行うことができる。

#### 4.3.2 radishフレームワーク

Spring Frameworkと同様に、新システムのコアになるのはMBで開発したJavaフレームワーク“radish”である。

ユーザーインターフェイス層にある画面は、通常のHTMLをタグ拡張したファイルで構成され、これをradishの“Web UIエンジン”が実行時にHTMLとJavaScriptに変換する。Javaで標準的なJSP(JavaServer Pages)とは異なり、画面デザインとプログラムコードが分離されるので、簡単な画面レイアウトの変更はHTMLの知識だけで可能になっている。

データアクセス層では、SQL文を直接プログラム内に記述しない。単純なデータベースアクセスは、DAOハンドラが動的にSQL文を生成して実行し、複雑なSQL文は外部ファイルに記述しておきSQLハンドラが実行する。従って、開発したアプリケーションは特定のデータベースに依存しない。

#### 4.3.3 ALIVEフレームワーク

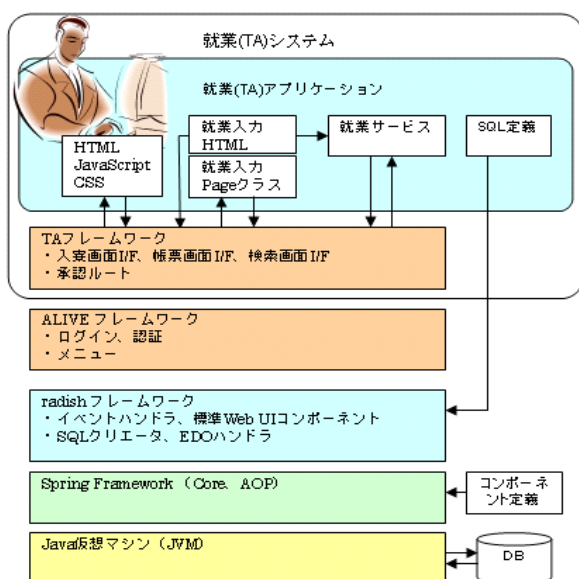
radish上に、ALIVE Solutionシリーズで共通に利用できる機能を“ALIVEフレームワーク”として実装した。

ALIVEフレームワークはサブシステム間を連携させるために重要な役割を担っており、メニュー画面、ログイン、サブシステムや画面の実行権限といった共通処理を実現している。

#### 4.3.4 TAフレームワーク

就業システム(TA)に特化した共通機能は、ALIVEフレームワーク上に“TAフレームワーク”として実装した。

就業システムで共通に使用するオブジェクトの定義や、承認ルートを実現する抽象化された機能が実装されている。個別の入力画面や帳票は、このフレームワーク上で動作することで、共通処理の隠蔽とプログラム構造の統一が図られ、カスタマイズし易い作りになっている。



ALIVE Solutionは(株)三洋電機ビジネスシステムの登録商標です。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。

図2. システム構成の概要

## 5. オブジェクト指向技術の導入効果

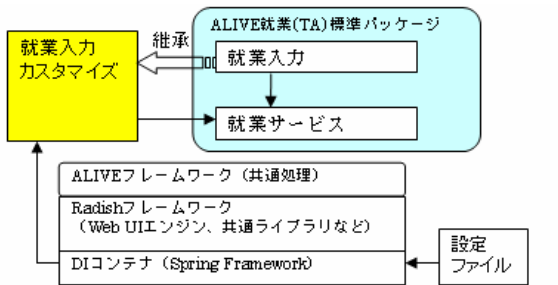
### 5.1 コンポーネント化の効果

コンポーネント化とDIコンテナの導入で、コンポーネント同士の依存関係が小さいシステムを構築することができた。その結果、コンポーネントの差し替えも容易で、不具合があった箇所の差し替えも簡単になったと考えている。

カスタマイズに関しても、オブジェクト指向技術である“継承”により標準機能との差分開発が可能となり、カスタマイズ後のシステム保守も改善された。

また、機能追加や差し替えが容易になるという特長を活かし、新たな機能要件の対応も容易に行うことができた。

例えば、就業入力機能を入力画面の“就業入力”と、入力データをチェックして更新する“就業サービス”に分けて開発したので、Web入力以外のタイムレコーダやタッチパネルへの対応で、同じ“就業サービス”を使うことができ、生産性と信頼性が向上した（図3）。



- (1) カスタマイズは、標準コンポーネントを継承して差分をプログラミング
- (2) DIコンテナの設定ファイルでカスタマイズ機能を指定。
- (3) 実行時、カスタマイズ差分と標準機能が連動する。

図3. コンポーネントの差し替え

システムを構成する業務コンポーネントは、必要に応じてオブジェクト指向技術の“多態”で実装されている。

例えば、“認証コンポーネント”は、一般的なログインID・パスワード方式の他に、Active Directoryによるシングル・サインオンなどに対応するが、機能が抽象化されているので上位のコンポーネントで認証方式を区別する必要がない。このため、認証コンポーネントを差し替えてもアプリケーションは変更せずに対応した（図4）。

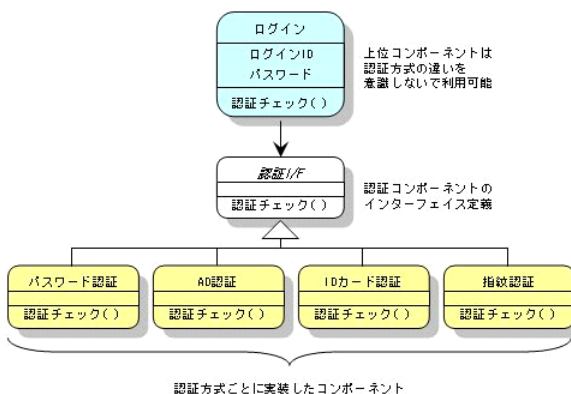


図4. 認証コンポーネントの実装例

### 5.2 UMLの採用

要件定義ではUML(Unified Modeling Language)のユースケース図、詳細設計の一部でクラス図、シーケンス図を採用した。今回はプログラム開発をオフショア開発で行ったが、海外でも通用し言葉に頼らないUMLのような国際標準は有用だった。

ただし、UMLの定義は細かく、過度に採用するとドキュメントが予想外に大量になって、生産性を落とすことになるので、どこまで導入するか留意が必要だった。

## 6. むすび

我々ITベンダーは、顧客から寄せられる様々な要件に素早く対応していくため、従来型のシステム開発では難しい難問を克服する必要が出てきた。この問題解決のため、オブジェクト指向技術をあらゆる企業に取り入れるようになってきた。MBでも、Java言語への取り組みと合わせて、オブジェクト指向技術に取り組んできたが、今回のALIVE SolutionシリーズのJava化は、こうしたMBの取り組みの象徴とも言える。

今回開発したフレームワークをベースに、ALIVE Solutionシリーズのサブシステム開発を行うと共に、様々な企業に対応する過程で蓄積されるノウハウ、カスタマイズされたコンポーネントを充実することで、さらに開発効率を向上させたい。

更なるコンポーネントの充実とフレームワークのブラッシュアップを継続して行い、変化に強い情報システムの構築を可能にしてシステムの付加価値を上げることが、21世紀を生き抜く企業の情報システムへの貢献に繋がると考えている。

## 参考文献

- (1) ロッド・ジョンソン：実践J2EEシステムデザイン，ソフトバンククリエイティブ（2003）