オープンソースを援用した
部分直接償却債権管理システムの構築について

（株）九州地区農協オンラインセンター

■ 執筆者Ｐｒｏｆｉｌｅ ■

池田　貴康

2007 年　株式会社九州地区農協
オンラインセンター　入社
業務部担当

■ 論文要旨 ■

ゆうちょ銀行誕生を背景とした金融業界の競争激化に伴い、対外的信用度のバロメータとなる不良債権比率の改善を図ることが急務であり、早急の対策として部分直接償却債権の管理を主な目的としてシステム構築を行った。システム構築にあたり、新規案件のため細かい要件定義を行う必要があり、かなりの工数を要すると予測された。しかし、早急のシステム化が求められているため、短期間でのシステム構築を目指した。短期構築かつコスト削減を実現する手段として、オープンソースを援用し、システム開発を効率的に実施し、短期構築かつ低コストを実現した。

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　当社の概要

当社は１９７７年に九州７県のＪＡバンクの信用業務を統合し設立されたシステム開発・保守・運用の会社であり、現在、１２０ＪＡ/２６００店舗を結ぶオンラインネットワーク網を完成させ個人取引先は九州エリア内に約５５０万人、九州全域の金融インフラを支えている。

事業内容：

・計算事務の受託
・情報提供サービス業務の受託
・ソフトウェアの開発・販売
・施設の賃貸並びに受託管理

設立　　：昭和５２年１０月１日
資本金　：８５億円
従業員数：７８名
売上高　：5,３０６,４１５千円（２００８年3月期）
代表者　：代表取締役社長　倉光　一雄
事業所　：福岡市南区

１．２　部分直接償却債権管理システム開発について

ゆうちょ銀行誕生を背景として金融業界の競争激化に伴い、今後の経営見直しが急務となっている。そこで中期経営計画の一環として対外信用度のバロメーターとなる不良債権比率改善を目的とし部分直接償却実施を実現する必要があり、部分直接償却債権の適正な管理をおこなうため新規システムを構築することとなった。

システム構築にあたり、迅速な開発、また、低コストでの実施が求められているため、それらを実現するための開発環境の構築、開発方法の選定を行った。まず、低コストを実現するためにオープンソースの採用を決定した。それに伴いエンタープライズで広く使われており、CGI 技術よりも高パフォーマンスを実現する Java 言語（J2EE、JavaEE）をプログラミング言語として採用した。しかし、当社としては Java 言語におけるシステム開発は今までに実施前例がなくリスクが懸念された。

リスクヘッジとして以下のポイントを勘案し開発プロセスの構築作業を行った。

- 学習コストが低いものを選定する。（工数削減）
- 自動化可能な部分は極力自動化を行う。（品質確保、工数削減）
- かゆいところに手が届く（例外に対処しやすい）ものを選定する。
  （新規開発によるリスク低減）

上記ポイントを踏まえ、システム開発においてのオープンソースの選定過程や工夫した点

を紹介する。

２．部分直接償却債権管理システムについて

２．１　システムの概要

部分直接償却債権管理システムは金融業務における不良債権を管理する目的で構築する。部分直接償却とは、資産の自己査定の結果、「回収不能または無価値」と判定された担保・保証債権について貸出額から担保の評価額、または保証による回収が可能と認められる額を控除した貸出残高を貸倒償却として償却額から減額することであり、この部分直接償却によって一定の不良債権管理を財務諸表上、オンバランスからオフバランスへシフトすることが可能となる。

部分直接償却の管理には一般に直接法（債権元帳から直接減額し、減額した金額を別管理する方法）と間接法（債権元帳は減額せずに、減額分を別管理し、B/S 残高において減ずる方法）とがあるが、利息算出について直接法の場合は債権元帳と部分直接償却額が必要である。一方、間接法の場合は債権元帳のみで算出が可能なことから、このシステムでは間接法を採用することとなった。

初めにシステム概要を以下（図１　部分直接償却債権管理システム概要図）に示す。

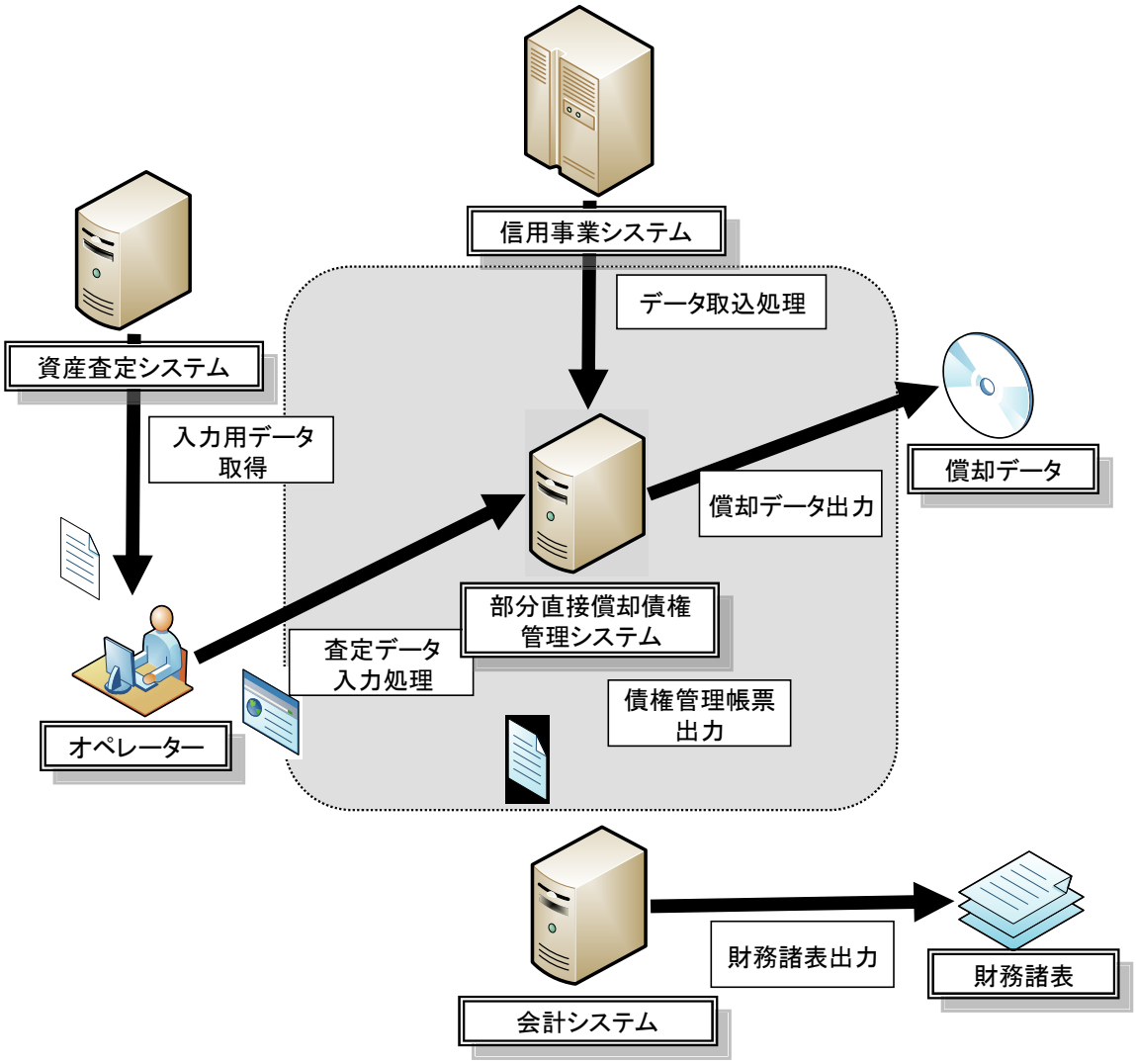

図 1　部分直接償却債権管理システム概要図

運用の流れとして、入力担当者は年度末に資産査定システムを利用し、資産の自己査定を実施する。その後、得られた自己査定結果の出力データを基に部分直接償却債権管理システムに対し、査定結果の非、Ⅱ、Ⅲ、Ⅳ分類額の入力を行う。
部分直接償却債権管理システムには予め、ホストからの債権レコードをバッチ処理においてデータベースに格納してある。これらを用い部分直接償却額の算出を行う。その後、算出したデータを基に各種、帳票を出力する。得られた帳票を基に会計システムへのデータ入力を行い、財務諸表の作成を行う。

部分直接償却債権管理システムが保有する機能は以下のとおりである。

① 部分直接償却額の月次自動更新機能（間接法）
② 部分直接償却レコードの登録・更新・削除機能
③ 部分直接償却レコードの検索機能
④ 部分直接償却レコードの照会機能
⑤ 部分直接償却データ年次自動洗替機能
⑥ 各種部分直接償却額帳票出力機能

⑦　保有データー括出力機能

２．２　システム構成について

本システムの構成を以下（図２ 部分直接償却債権管理システム構成図）に示す。本システムは Web ブラウザをユーザインターフェースとする Web アプリケーションである。

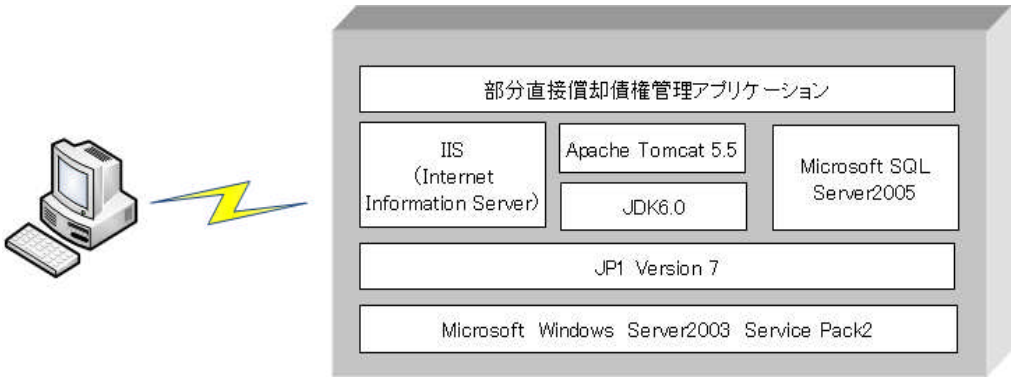

図 ２　部分直接償却債権管理システム構成図

サーバ構成は費用面や、システム規模により１台のサーバにて AP サーバ、DB サーバを兼用することとなった。AP/DB サーバの基本構成では、ハードウェアに富士通株式会社製のサーバ「PRIMERGY RX300 S3」を採用した、サーバ OS（オペレーティングシステム）にはマイクロソフト株式会社の「Microsoft Windows Server 2003 Standard Edition Service Pack2」を採用した。

Web サーバソフトウェアとしてマイクロソフト株式会社のインターネットサーバソフトウェア「Internet Information Server(IIS)」を採用し、アプリケーションサーバソフトウェアに「Apache Tomcat 5.5」*1 を採用した。本システムである部分直接償却債権管理システムの Web アプリケーションプログラムはこの Apache Tomcat5.5 上において動作している。また、Apache Tomcat5.5 は Sun Microsystems 社製の JavaVM（Java バーチャルマシン）上で動作しており、加えて、本システムである部分直接償却債権管理システムのバッチ処理アプリケーションプログラムについても JavaVM 上において動作している。

データベースソフトウェアとしてマイクロソフト株式会社製の「SQLServer2005 Standard Edition」を採用した。また、統合システム運用管理ソフトウェアとして株式会社日立製作所製の「JP1 Version 7」を導入している。

３．システム開発の課題と方策

３．１　システム開発の現状と課題

現在、Java 言語はエンタープライズシステム開発において多くの実績を誇る。当社においても Java 言語による Web システム構築は有用であるとの判断や、業務の幅を持たせることの重要性からも採用を決断した。しかしながら、部分直接償却債権管理システム構築以前において、当社では Java 言語を使用した Web システム開発実績がなく、いかにリスクを低減するかが課題となった。そこでまず、短納期、低コストの要求を満たし、なおかつ新

規開発におけるリスクを補うようなシステム開発の実施が求められた。

３．２　システム開発プロセスの構築

システム開発における開発プロセスは同じ条件の案件でない限り異なるのが通常である。システム開発の最適なプロセス構築こそが、システム開発における重要なリスクヘッジ作業であると位置づけ、短納期、低コスト、スキルカバーを満たす開発プロセスを目指し、システム開発プロセスの策定を行った。

３．２．１　開発プロセスモデルの選定

初めに開発プロセスの構築について、ベースとなる開発モデルの選定を行った。今回のシステム開発においては、当初より要件定義作業がスムーズに行えていたこともあり、手戻りのリスクは少ないと判断し、ウォーターフォール開発モデルをベースモデルとすることを決定した。また、当社でのウォーターフォールモデルでの開発ノウハウの蓄積も採用理由の一つとした。

ただし、一部の要件定義・設計・実装工程においてはプロトタイピングモデルの要素を取り入れた。プロトタイピングモデル要素の導入理由であるが、一つは工数削減を目的としている。また、もう一つの理由として、次の懸念事項があったためである。

本システムの開発においては、迅速なシステム構築が求められており、フレームワークの採用は必須であるとの認識であった、しかし現状では、世の中に多くのフレームワークが生み出されており、どのフレームワークを採用するか、またどのフレームワークと、どのフレームワークを組み合わせるかなどのフレームワークの選定・組み合わせによるコスト増大が懸念され、選定・組み合わせコストの低減を行う目的としてプロトタイピングモデルの部分導入を行うに至った。

３．２．２　開発環境の構築

次に Java 言語を用いた開発を踏まえた開発環境の構築を行った。開発環境構築において採用したソフトウェア・ツールを選定理由を含め次に述べる。

Eclipse:[*2]

Eclipse は Java 言語を用いての開発において最も有名である IDE（Integrated Development Environment） であり、今回の開発環境において標準的な開発環境と定めた、他に NetBeans IDE との比較検討も行ったが、豊富なオープンソースプラグインが用意されていることや、開発用コンピューターのスペックが低いものでも開発が行えるように、消費リソースを比較し、NetBeans IDE と比べて、動作中のコンピューターメモリの消費量が少ないことを選定理由とした。

FindBugs:[*3]

FindBugs は Java ソースコードの静的解析ツールであり、ソースコードの不具合や、推奨されないコーディングを行っていた場合、警告を発する機能を有している。バグの早期発見における開発工数の削減、保守性の向上を理由として採用した。

CSE（Common SQL Environment）:[*4]

データベースの操作には SQL が必須であり、SQL 開発環境ソフトウェア、CSE を標準的な

開発環境と定めた、SQL の編集実行、レコードの新規作成・更新・削除、データベースオブジェクトのブラウズなどの機能を有し、ODBC（Open DataBase Connectivity）データソース対応の RDBMS（Relational DataBase Management System）ならばすべて接続可能となることを選定理由とした。

Microsoft Visual SourceSafe 6.0 ：

ソースコードリポジトリとして VSS（Microsoft Visual SourceSafe 6.0 ）を採用した、Microsoft Visual SourceSafe の採用理由については、既存の別プロジェクトによって、既にソースコードリポジトリを構築済みであり、ライセンス、リポジトリ構築コストの削減と、ノウハウの流用といった点を考慮した結果となった。

Apache Ant ：*5

Java ベースのビルドツールであり、VSS を採用した事由も含め、汎用的なビルド作業として対応が可能な Apache Ant を採用した。

Hudson：*6

Hudson は CI（Continuous Integration）ツールであり、継続的なビルド実行、成果物のバージョン管理機能、成果物のダウンロードが Web ブラウザを用いて行うことができ、ソフトウェア移送・導入コストの低減が見込めることや、構築が容易な点も含めて採用理由とした。

jconsole:

jconsole は JavaSE5 以降 Java のパッケージに標準で梱包されるようになった監視ツールで、サーバにおける JavaVM 上のヒープメモリ使用状況やプログラムの実行単位をあらわすスレッド数、クラスのロード数といった、リソース状態の監視を行う機能を有しているツールである。標準で梱包されているツールとはいえ、必要とする機能が十分実装されているとの判断を行い。パフォーマンスチューニングなど、システムテストの支援ツールとして採用した。

Apache JMeter：*7

Apache JMeter はパフォーマンス測定、サーバ負荷機能を有した、オープンソースとしては高機能なテストツールであり、ストレステストやパフォーマンスチューニングの支援ツールとして導入を決定した。

上記、開発環境ソフトウェア、ツールを開発の各フェーズにて利用することとした。

３．２．３　ソフトウェア基盤の構築

迅速な開発を行うためにはフレームワークの採用が必須であり、本システムにおいてはバッチ処理、Web アプリケーションを構築することを前提にフレームワークの選定・導入、拡張機能の構築を行った。

まずは、JavaEE アプリケーション開発において、アプリケーションレイヤについてのレイヤ定義を行った。本システムにおいてはビュー、画面など UI（ユーザインターフェース）をプレゼンテーション層と定義し、リクエストやプログラムの制御を行うレイヤをコントローラー層、業務処理を行うレイヤとしてサービス層、データベース更新など、データ操作を行うレイヤをデータアクセス層として定義した。定義したレイヤを元に、バッチ処理、Web アプリケーションに対してそれぞれレイヤの適用を行い以下（図３ 部分直接償却債権管理システムアプリケーションレイヤ図）の様な設計モデルとした。

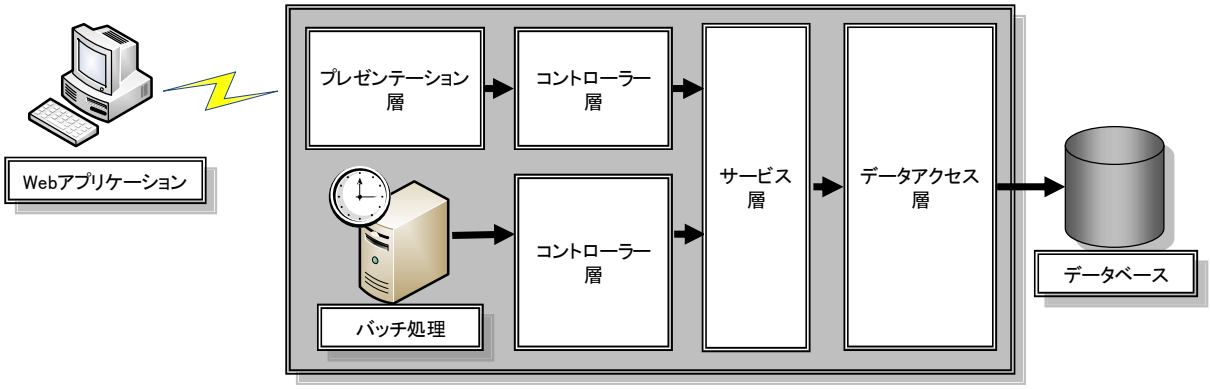

図 ３　部分直接償却債権管理システムアプリケーションレイヤ図

Web アプリケーション：

プレゼンテーション層、コントローラー層、サービス層、データアクセス層のレイヤ構成とした。

バッチ処理：

コントローラー層、サービス層、データアクセス層のレイヤ構成とすることとした。

アプリケーションレイヤの定義確定を受けてフレームワークの選定作業に移行した。

まず、バッチ処理、Web アプリケーションに共通的なレイヤにおいてのフレームワーク選定を行った。共通的なレイヤとしてサービス層とデータアクセス層がそれに当たるが、機能単位で比較した場合、実装時にサービス層のプログラム実装はあるがデータアクセス層のプログラム実装はないパターンが考えられ、データアクセス層よりもサービス層における実装頻度が高いと判断し、フレームワークの選定をサービス層から優先的に行うこととした。

まず、プログラムのテストや保守性、再利用性を考慮し、DI コンテナの採用が望ましいとし、DI コンテナ機能を有するフレームワークをサービス層の選定対象とした。DI コンテナとはコンポーネント間の依存関係を取り除き、開発容易性の向上を実現したソフトウェアである。調査対象とした DI コンテナ機能を有する OSS フレームワークは Spring[*8] と Seasar2[*9] の二つのソフトウェアとした。Spring においてはシステム導入実績事例が多く、また、他のレイヤにおけるフレームワークとの連携機能が豊富であり、DI コンテナとして現在、世界で最も有名なソフトウェアである。Seasar2 は国産の OSS フレームワークであ

り、フレームワークに対する日本語情報が豊富である。しかし、他のレイヤとの連携可能なフレームワークは多くはなく、Seasar2 を各レイヤに適用した独自のフレームワーク群を同プロジェクトにおいて提供している。

結果的に、導入実績が豊富な点、他のレイヤとのフレームワークとの連携容易さを考慮し、Spring を DI コンテナ機能を有するフレームワークとしてサービス層に採用することを決定した、日本語情報については Spring についての日本語書籍も発売されていることを加味し、プロジェクトに影響を及ぼすデメリットにはならないと判断した。

次に、共通的なレイヤにおけるデータアクセス層のフレームワークの採用を検討した、このデータアクセス層には O/R マッピングフレームワークがしばしば採用される、そこでまずは O/R マッピングフレームワークに対する調査を行った。調査対象としたのは Hibernate[*10] と iBatis[*11] である、Hibernate は O/R マッピングフレームワークの代表格として有名なフレームワークであり、加えて導入実績が多数あり、SQL を記述しない代わりに、HQL (Hibernate Query Language) による柔軟な検索を行うことができる機能を有している。iBatis は Apache プロジェクトで開発されている O/R マッピングフレームワークである。

調査の結果、O/R マッピングフレームワークの使い方として基本的にデータベースマッピングファイルを記述しなければならず、これは SQL の記述量よりも冗長であると判断した。また、開発メンバーには SQL に対するスキルセットを持ち合わせたメンバーが多くおり、O/R マッピングフレームワークの学習コストをかけるよりも SQL を用いた開発を行うほうが効率的であると判断した。しかしながら、JDBC をそのまま採用した開発においても、コストが増大するため、Spring フレームワークとの連携が容易な JDBC フレームワークである SpringJDBC を採用することとした。SpringJDBC は Spring フレームワークと連携することでデータベーストランザクション管理機能を有し、データアクセス層のプログラムにおいてデータベーストランザクションを意識しない開発が可能となる。また、O/R マッピング機能基盤も有している。本プロジェクトにおいては更に、SpringJDBC フレームワークを補完する機能を追加した。

初めに業務要件としてデータベース保有データの CSV ファイル出力の要求があった。CSV ファイルはカンマ区切りの表であらわされるファイル形式であり、また、データベースにおいても本システムにて RDB を採用しており、こちらも表であらわされる。しかし、O/R マッピング機能を採用した場合、データベース保有のデータを CSV ファイルへ出力するためには表からオブジェクト、その後、オブジェクトから表への変換を行う必要があり、効率が悪い。そこで補完機能として、通常の表データとしてデータベース保有データを取得できる機能を追加した。その他以下のような機能を追加した。

SQL カタログ機能：

保守性、再利用性の観点より SQL を外部ファイルに保持し利用可能とする機能を追加した、この SQL ファイルの形式は単独で SQL 実行ツールにて実行可能であり、また、補完機能にて条件によりパラメータの変更、データバインディングを行うことができる。また、複数の SQL ファイルを XML ファイルにまとめて、一元管理する機能も追加した。

SQL 自動生成機能：

データベース定義書より、表に対する、DROP、CREATE の DDL、CRUD の DML を生成するユーティリティーツールを作成した、合わせてデータアクセス部分の Java ソースと Entity 部分の Java ソースも自動生成する機能を構築した。

ソース自動生成機能：

複雑なデータ抽出に対応するために作成した SQL を基にデータアクセス部分の Java ソースと Entity 部分の Java ソースや、後述する Excel テンプレート用貼り付けタグを自動生成する機能を構築した。

次にプレゼンテーション層のフレームワークの採用を検討した、現在、主流な Web フレームワークの殆どがプレゼンテーション層とコントローラー層の両層セットになったフレームワークを提供しており、今回は両層を合わせて検討した。調査対象としたのは Struts[*12] と Click Framework[*13] である。

Struts は Java 言語における Web フレームワークのデファクトスタンダードで、導入実績が豊富なこともあり、当初は Struts の採用を重点的に検討した。一方、Click Framework は学習コストが低くなるように設計されており、また、フレームワークのソースコード量も Struts と比べて少なく、詳細なエラーリポート機能を備え、問題に対するトラブルシューティングが行いやすい点、更に標準機能として Spring フレームワークとの連携機能を備えている点、加えて Struts よりも豊富なコンポーネント部品群、Struts の struts_config.xml に見られる、Web フレームワークに頻出の設定ファイルを記述しないでよい点、HTML のレンダリング・テンプレートエンジンとして学習が容易な Velocity[*14] を採用している点などを評価した。

総合的に判断して、今回のプロジェクトにおいては、学習コストの低さ、Spring フレームワークとの連携部分の構築コストを節約できる点や、設定ファイルレスである点が工数削減につながると判断し、Click Framework を採用することを決定した。本プロジェクトにおいては更に Click Framework の補完機能を追加した。Click Framework はコンポーネントベースのフレームワークであり、Java 言語ベースでのコンポーネント単位の拡張が容易であり、以下のような拡張を行った。

バリデーション機能：

SQL インジェクション対策や、セキュリティ対策のために入力用コンポーネントに対しバリデーション拡張を行った。合わせて、クライアントサイドに JavaScript でのバリデーション機能も追加した。

ファイルダウンロード機能：

CSV ファイルや Excel ファイルを動的に生成し、ダウンロード可能とするユーティリティー機能を追加した。

各レイヤに適用するフレームワークを決定した後、その他の補完機能を以下のとおり追加した。

メッセージカタログ機能：

保守性の観点よりメッセージをメッセージファイルとして外部ファイルに保持し、システム内部で取得する機能、また、メッセージファイルを XML ファイルとして一元管理する機能を実装した。

メッセージ出力機能：

ユーザビリティの観点より、JavaScript を使用しダイアログメッセージを出力する機能を追加した。

ログ出力機能：

デファクトスタンダードなログ出力ライブラリの一つである Apache Log4J[*15] を導入し、加えて Spring フレームワークの AOP (Aspect Oriented Programming) 機能を用い、パフォーマンス計測用ログ、実行 SQL トレースログ、アプリケーションログを出力する機能を追加した。

設定ファイル自動生成機能：

Spring フレームワークや DI コンテナ機能を有する各種フレームワークは依存性を分離するために、依存性を外部ファイルとして定義することが多い。しかし、システム規模が大きくなるにつれて定義ファイルの記述作業は、煩雑化する傾向があるためメンテナンスコストが増大する。そこで、ソースコードを作成して、定義ファイルを記述するという製造プロセスを前提とし、更にソースコードに対し、一定の命名規則をプロジェクト内部で定義することによりソースコードから定義ファイルを自動生成する機能を追加した。

システムメッセージ通知機能：

内部エラーや処理状況など、アプリケーションレベルのメッセージを運用監視ソフトウェアへ通知する機能を追加した。

Excel テンプレート機能：

本プロジェクトにおいては、利便性や、データ処理の効率化を理由として Excel をベースとした帳票を出力することとなった。そこで Excel のテンプレートを用いて帳票を生成する機能を追加した。

URL 書き換え機能：

セキュリティ向上の観点より、リクエストされた URL を書き換え、実際の URL を隠ぺいする機能を追加した。また、実際の URL よりも短い URL にてシステムを利用可能となり、ユーザビリティの向上が期待できる。

データ移送・交換機能：

アプリケーションレイヤを分割定義した場合、各アプリケーションレイヤごとにインプット、アウトプットのためのオブジェクトやデータの移送が発生する、そこで命名規則を元にデータを自動的に移し替える機能を追加した。

上記、機能を拡張を行ったものを本プロジェクトでのソフトウェア開発基盤とした。

３．３　システム開発における施策と工夫点

今回のプロジェクトにおいて、開発プロセス構築の際の施策や工夫点を以下に述べる。

システム開発においては、通常、業務により要件が異なるため、プログラミング作業すべてを自動化することは難しいと考える。逆に言うと業務要件に対する学習コストと業務要件をプログラミング・実装する作業は、システム開発を遂行する上で人的作業の代表的なものであると考える。そこで、業務要件以外の作業に対し自動化・簡略化の可能性を探り、本プロジェクトのシステム開発プロセス構築を行った。

まず、プログラミング作業において業務要件以外のコード記述を簡略化、削減するアプローチの一環として、フレームワークに設定ファイルの自動生成機能を追加し、設定ファイル記述における、記述作業量の削減や、記述内容理解などの学習コストの低減を図った。

一方では、リソース管理コストが低減されるメリットがあるため設定ファイルレス機能を追加する方策も考えられたが、設定ファイルを利用し、システム設定の把握のしやすさ、例外パターンが発生した場合に設定ファイルを書き換えることで対応可能であるといった、保守性、変更容易性の観点から。設定ファイルレスではなく設定ファイルの自動生成という位置づけで留めた。

また、プログラミングに必要なスキルセットとして、コアスキルを SQL、Java 言語、Javascript の３つの言語のみの理解で対応可能であるといった、所要スキルを限定化し、学習コストの削減、開発容易性の向上を図った。

他に、開発に必要なスキルとしてテンプレートの記述能力を要すが、これは学習が容易な Velocity を採用し、Velocity の記述様式の理解のみで Excel テンプレート、Web ページの HTML テンプレート共に同様の記述様式で対応可能といった、テンプレート記述における所要スキルの共有化を図った。更に一部のテンプレート用タグの自動生成といった記述作業削減の機能も盛り込んだ。

また、現在の Web システムの入力インターフェースは決してユーザビリティーが高いとは言えない。特に、大量のデータ入力が要求される機能では入力負荷が高くなると考えられる。そこで、大量データ入力の場合は入力インターフェースを Web ページより使い慣れた Excel ファイルへシフトさせ、Excel ファイルをアップロードすることによって大量データ入力を行うといった機能とした。この結果、広範囲へのデータの貼り付け、入力途中の保存、Excel 特有の関数を使用可能といったメリットが享受でき、かつ、万が一 Web システムにて入力後のエラーが発生したとしても再度入力することなく、ファイルを修正若しくはそのままアップロードする対応のみでよいため、効率的であると考えた。

４．システム開発後の評価

方策の効果としてまず、オープンソースを全面的に採用したことにより、プログラム製造の際のソフトウェアコストがほとんどかからなかったことが挙げられる。これは資源の効率化（コスト削減）を実現したことが言える。

次に時間の効率化であるがフレームワーク並びに自動生成ツールの構築によるコード記述量削減が挙げられる。今回の部分直接償却債権管理システムにおいて記述したコード量を下記に示す（表１ 部分直接償却債権管理システム　コード記述量削減率）

	総コード行数	記述削減行数	記述コード行数	記述量削減率
Webアプリケーション	8279	3284	4995	39.7%
バッチアプリケーション	2457	1493	964	60.8%
フレームワーク(補完分)	6489			
合計(フレームワーク除く)	17225	4777	12448	28%
合計	17225	11266	5959	65%

表 １　部分直接償却債権管理システム　コード記述量削減率

本プロジェクトの成果物としてシステムを実現する総コード行数（コメント除く）は 17225 行であり、内訳としてフレームワーク（補完分）が 6489 行、Web アプリケーション、バッチアプリケーションがそれぞれ 8279 行、2475 行となった。Web アプリケーションにおいては自動生成ツールを用い、コードの自動生成を行ったことから、3284 行のコードを自動生成し、結果 39.7%の記述量削減率となった、また同様にバッチアプリケーションにおいては 1497 行を自動生成し、結果 60.8%の記述量削減率となった。バッチアプリケーションにおいてはプレゼンテーション層の考慮が不要なため自動生成におけるコードの割合が高い値を示した。

今回のプロジェクトとしてはフレームワークの構築作業が発生したため実記述コードの行数は 12448 行となり全体の 72%を示し、プログラム製造においては約 30%の生産性向上という結果になった。しかしながら今後の開発においてはこのフレームワーク分のコードが再利用可能となり、仮に今回の案件と同等であった場合 11266 行におよぶコードの記述削減が実現でき、記述量削減率は 65%という数値が見込まれる。

５．おわりに

方策の効果として、3 ヶ月の開発期間であったが、遅延することなく開発を終了することができた。更に稼働後、半年近く経過しているが、特に障害も発生せず安定稼動している状況である。今回のシステム開発においてはフレームワークの選定や、補完機能・その他ツールの構築など、開発準備期間に時間を要したが、今後のシステム開発案件においては、この開発準備期間の短縮を図ることが可能となり、さらなるメリットが期待できる。

今回構築したフレームワークは、まだまだ、汎用的なフレームワークであり、種々の業務に対してのシステム構築に対応可能な反面、特定業務に依存し、かつ業務内では汎用的な部分の作りこみが毎回発生してしまう。これは OSS のフレームワークという位置づけで

あれば当然の理屈である。しかし、通常はフレームワークを用いて、特定の業種のシステムを構築するのであるから、構築対象の業種がある程度わかっている場合、フレームワーク自体が構築対象の業種に特化しておくと、より効率的であると考える。そこで今後の課題として、今回構築したフレームワークに対し、業務ノウハウからの蓄積を行い、業種特化（特に信用業務）したフレームワークの構築を行う予定である。

本稿では、部分直接償却債権システムを構築するにあたり、短納期、低コストを実現するために実施した方策を紹介した。OSS はシステム開発をより良く行うためのさまざまなツール・ソフトウェアを提供している、もちろん OSS は未成熟であったり、不完全のものも多数存在する。それらを如何に利用し、また不足の機能をどのように補完していくかがシステム開発を改善するための重要な手段であり、それが仕事をする楽しさ・喜びの一つになり、更にはそれが OSS 提供者の喜びであると思う。まだまだ、改善の余地はあるものの、本実績を今後のシステム開発に活かしていただければ幸いである。

参考文献

［1］岡本 隆史 , 金子 崇之 , 吉田 英嗣 , 権藤 夏男 :“Light Weight Java―JSF/Hibernate/SpringによるフレームワークでWebアプリケーションの開発効率向上”, 毎日コミュニケーションズ

［2］[thinkit], 時代は今「DIxAOPコンテナ」:http://www.thinkit.co.jp/free/compare/15/1/1.html

*1 Apache Tomcat, http://tomcat.apache.org/
*2 eclipse, http://www.eclipse.org/
*3 FindBugs, http://findbugs.sourceforge.net/
*4 CSE, http://www.hi-ho.ne.jp/tsumiki/
*5 Apache Ant, http://ant.apache.org/
*6 Hudson, https://hudson.dev.java.net/
*7 Apache JMeter, http://jakarta.apache.org/jmeter/
*8 Spring, http://www.springframework.org/
*9 Seaser2, http://www.seasar.org/
*10 Hibernate, http://www.hibernate.org/
*11 iBatis, http://ibatis.apache.org/
*12 Struts, http://struts.apache.org/
*13 Click Framework, http://click.sourceforge.net/
*14 velocity, http://velocity.apache.org/
*15 Apache log4j, http://logging.apache.org/log4j/