OSS ベースの Java EE アプリケーション・フレームワーク ―― Maia の研究と開発

Java EE Application Framework based on Combination of “OSS” ―― Research and Development of “Maia”

山　口　周　志

要　約　Java EE 関連のオープンソースソフトウェア（以下，OSS）の品質向上や高機能化に伴い，開発工数の削減と開発生産性の向上を目的に，システム開発への OSS 適用が一般化してきている．他方，OSS の重要性が高まり，同様な機能を持つ OSS が多数開発された結果，OSS の機能・特徴を把握することが難しくなってしまった．また OSS の種類が増えたことで，利用知財の蓄積すらも難しいのが現状である．このような状況においては，標準的な OSS 適用指針を策定し，知財の蓄積と活用を促す必要がある．

OSS に関する技術調査の結果，Struts，Spring，Hibernate を，推奨する OSS の組み合わせ「OSS 基本パターン」として選定した．ただ，これらの OSS だけでは一般的にシステムに必要とされる非同期処理・帳票処理等の機能をサポートすることができないため，Java システム標準アーキテクチャを策定し，不足部分の補完機能を開発して，システム全体をサポートすることが重要である．

日本ユニシスグループでは，OSS 基本パターンの不足機能を商用フレームワーク MIDMOST for JavaEE を参考にした独自実装により補完することで，システム全体をサポートするアプリケーション・フレームワーク Maia の研究・開発を進めている．Maia は，OSS と商用ソフトウェアの利点を統合し，また知財の蓄積と活用を促す試みである．

本稿では，標準的な OSS の適用指針である OSS 基本パターン，Java システム標準アーキテクチャ，不足機能の補完実装に関して解説を行い，日本ユニシスグループが研究・開発を進めている Maia について紹介する．

Abstract　With the improvement of the quality and the functionality of Java/Open Source Software (OSS), the usage of application layer OSS have been widespread in the information systems development. However, rapid increase in the number of OSS of similar functions as a result of OSS development race, makes it difficult for us to tell the characteristics and the capability of individual OSS, and to accumulate the knowledge about OSS. To resolve these problems, it is necessary to accumulate and utilize OSS knowledge by standardizing the OSS application guide.

From the result of our recent research on OSS, we selected “Struts”, “Spring” and “Hibernate” as standardized OSS application guide that is “OSS Fundamental Combination”. However, only these combined OSS is not enough to support the Java EE system because it does not support the asynchronous processing and report writing, and so on. Therefore, we focus on developing the new framework for standard architecture for Java EE system by combining the three OSS frameworks mentioned above and supplementary functions to enhance the reliability of Java EE system.

Nihon Unisys Group implements supplementary functions by reference to the proprietary framework for

Java EE system which is “MIDMOST for Java EE.” Thus, we have researched and developed an application framework for Java EE system, which is named “Maia”, supports Java EE systems in terms of development and execution. Maia is an approach to accumulate and utilize OSS knowledge and combine the advantage of OSS and proprietary software.

This paper overviews the standardization of the OSS application guide “OSS Fundamental Combination”, the standard architecture for Java EE system, supplementary functions, and introduces Maia that Nihon Unisys is going to research and develop.

1. はじめに

企業情報システムの重要性が高まる中，今日の情報システムは益々複雑化・大規模化するとともに，システム開発期間の短縮が求められている．そのため Java EE ベースのシステム開発では，商用あるいは独自のアプリケーション・フレームワークを適用することが一般化している．アプリケーション・フレームワークは，複数のシステム間で共通する基本的な構造や機能セットを実装したソフトウェア部品であり，システム開発に適用することで開発生産性や品質の向上を図ることができる．

しかし，商用のフレームワーク製品は一般的に高価であり，特定ソフトウェアベンダーへの依存度が高いという課題がある．また，インターネット等で公開されている利用技術情報量の少なさやこれに起因する習得コストの問題も無視できない．そこで，近年はアプリケーション・フレームワークにオープンソースソフトウェア（以下，OSS）を活用することが選択肢として挙がってきている．

1990 年代に誕生した直後の OSS は，その機能範囲や品質，サポート等が必ずしも十分ではなく，ライセンス許諾範囲が不明確であるというコンプライアンス上の問題を含んでいたこと等により，企業情報システムへの浸透はあまり進まなかった．しかし，各 OSS の開発コミュニティによる継続的な努力により機能と品質が向上し，またソフトウェアベンダーによる有償サポートサービスの提供や啓発活動が行われた結果，OSS は企業情報システム開発に広く適用されるようになってきた[1][2]．オペレーティングシステム，ミドルウェア，アプリケーション，開発支援ツールなどの各領域で OSS の整備が進み，特に近年では，OSS の活用を支持する欧州委員会の報告書[3]や，信頼性・パフォーマンス・セキュリティ・TCO[*1] などの定量的なデータに基づき OSS を推奨する論文[4]，さらには米国政府での OSS 活用を記した報告書[5]などが相次いで発表されており，商用ソフトウェアから OSS への移行が顕著になってきている．

OSS 普及の背景にある，OSS を適用する上での一般的なメリットは次のとおりである．

1) 製品自体が無償であるためソフトウェア費用を削減できる
2) ソースコードを含む各種情報がインターネット上で広く公開され，各製品を解説する Web サイトや書籍等も豊富なため，情報収集や習得に要するコストを軽減できる
3) 特定ソフトウェアベンダーの製品による囲い込み（ロックイン）を回避し，より優れた製品やサービスを利用者が主体的に選択できる

他方，一般に OSS は各開発コミュニティの有志によって開発が行われており，最先端の技術が取り込まれやすい反面，機能拡張・変更や不具合修正などを含むバージョンアップが頻繁に発生する．また，OSS に関する幾つかの課題が浮き彫りになってきている．

それは次々に新しいOSSが開発され，さらに様々な要求に従い細分化・多機能化しているため，個々のOSSの機能・特徴の把握が難しくなってしまったことである．その結果，OSS導入に際して掛かる技術検証コストが膨らんでいる．またOSSの種類や組み合わせのパターンが増えたことで，不具合情報やパフォーマンスチューニングなどの知財の蓄積と活用に問題をきたしている．さらに，ミドルウェアより上位レイヤのOSSにおいては，企業情報システム構築に必要となるシステム監視やユーザ管理などの機能が欠落しているものが多く，その部分を含めた補完機能開発が必須である．

OSSに伴うこれらの課題は，複数のOSSを組み合わせて利用することになるアプリケーション開発において特に顕著である．そこで，アプリケーションレイヤにおける推奨OSSのパターンを策定し，その利用を促進することで，推奨OSSの利用技術の蓄積と活用を行い，開発生産性や品質レベルの向上を図ることが可能となる．

日本ユニシスグループでは，OSSの有効活用を行いつつ，独自に不足機能を補完する方式を前提とした，Java EE用アプリケーション・フレームワーク「Maia（マイア）」の研究・開発を進めている．「推奨するOSSの組み合わせ」と「独自の補完機能」から構成されるMaiaの適用推進と知財蓄積を図ることが，システム構築プロジェクトにおける開発品質と生産性の向上に寄与するものであると考える．

本稿では，まずOSSに関する技術調査と，その結果選定した推奨するOSSの組み合わせであるOSS基本パターンについて紹介する．次に，Java EEシステム全体の機能範囲を検証するために策定したJavaシステム標準アーキテクチャと，OSS基本パターンだけでは不足する機能の独自実装による補完について述べる．そして最後に，Maiaとその内部構造に関する技術解説を行い，今後の開発予定について論ずる．

2. 推奨するOSSの組み合わせ「OSS基本パターン」

本章では，Java EEアプリケーション開発に利用するOSSの機能・特徴を把握するために実施した技術調査の概要と，その結果を基に策定した，推奨するアプリケーションレイヤのOSSの組み合わせである「OSS基本パターン」について述べる．

2.1 Java EEフレームワークのOSS技術動向

OSSベースのJava EEアプリケーション・フレームワークに関する推奨プロダクトの組み合わせを規定するに当たり，主要なOSSの機能・特徴の把握と整理を目的とする技術調査を実施した．この調査は，アプリケーションレイヤのOSSのみを対象にしており，PostgreSQLやJBoss等のミドルウェアや，Linuxの各種ディストリビューション等は対象外である．

調査に先立ち，対象とするOSSのカテゴリや機能範囲の分類を目的に，Java EEシステムの内部アーキテクチャ・パターンの整理を行い，一般的なWebシステムに必要とされる機能を大きく八つに分類した（図1）．

今回の調査では，図1で示した構成要素のうち，Webアプリケーションの中核であるオンライン処理を構成するプレゼンテーション層，ビジネスロジック層，データアクセス層の三層に注目し，これらの層に属するソフトウェアを調査対象としている．2.1.1項以降で各層毎の調査観点と結果について述べる．

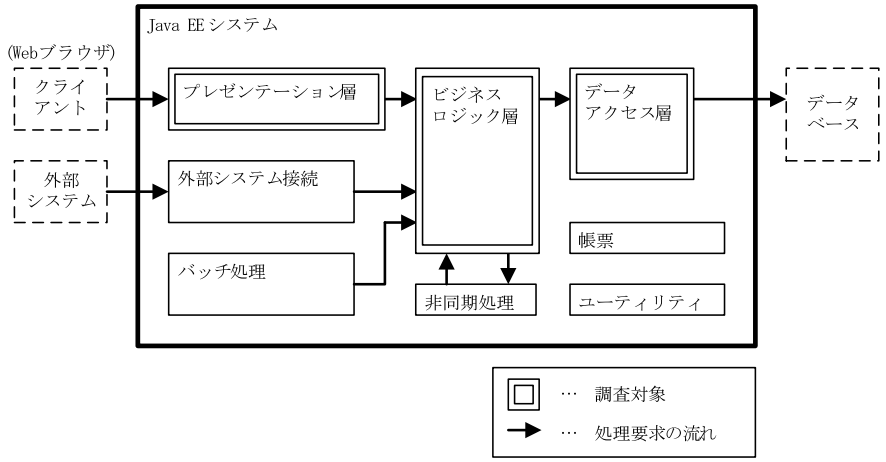

図 1　Web システム概要図

2.1.1　プレゼンテーション層

プレゼンテーション層の主たる機能は，クライアント（Web ブラウザ）画面の入出力処理のサポートであり，この層に属するソフトウェアでは，画面系の開発作業効率を向上させるための諸機能や，画面設計方式などが重要となる．具体的には，JavaEE 5 の標準仕様である Java ServerFaces（JSF）[*2] への準拠，プレビュー機能の有無，レイアウトテンプレート機能[*3] の有無，カスタムタグコンポーネントの機能範囲，実装・設定難易度や開発ツールの有無，入手可能な技術情報の質と量などである．調査対象としたソフトウェアは，MyFaces[*4]，Facelets[*5]，Shale[*6]，Tapestry[*7]，Mayaa[*8]，Click[*9]，Struts[*10] である．

調査の結果，今後の標準技術としては JSF が期待され，その動向を注視する必要はあるものの，これに準拠したソフトウェアである Shale や Facelets などは適用事例がまだ少なく，未成熟な面もあることから，現時点においては，実績が豊富でかつデファクトスタンダードと称される Struts の価値は依然として高いと判断した．

2.1.2　ビジネスロジック層

ビジネスロジック層は，プレゼンテーション層やデータアクセス層と連携して業務処理を実行するレイヤであるが，プログラムのテスト・保守・設定・再利用の容易化と，これに伴う開発生産性や品質向上の観点からは，「DI コンテナ[*11]」機能を有するフレームワーク部品を採用することが望ましい．

ビジネスロジック層の DI コンテナ機能を有するソフトウェアについては，他の層を構成する OSS との連携機能の豊富さ，設定・実装の容易性，サポートする OSS 数の多さに代表される汎用性が求められる．そのため基本機能はもとより，設定ファイルの記述にかかる負荷，関連プロダクト，サポートする OSS とその連携方法，実装・設定難易度の高さや開発ツールの有無，また入手可能な技術情報の質と量などが重要な要素となる．この層で調査対象とした OSS は，Spring[*12] と Seaser2[*13] の二つである．

調査の結果，Spring は他の OSS との連携が容易であり，かつ連携可能な OSS の種類も多いことが，また，Seaser2 は連携可能な OSS の種類は少ないものの，独自のソフトウェア群による一貫性のあるアプローチを採用していることが判明した．ビジネスロジック層の推奨ソ

フトウェアとしては，多くの OSS との連携をサポートしているという意味で汎用性が高く，連携に関する制限が少ないことから，Spring のほうが有利であると判断した．

2. 1. 3　データアクセス層

データアクセス層では，ビジネスロジック層からの要求に従い，データベースにアクセスし処理を行う．この層には「O/R マッピングツール」がしばしば利用される．データベースとアプリケーションとの対応付けを担う O/R マッピングツールの評価においては，データベースへの処理機能の豊富さやアプリケーション設計・実装の容易性が重要となる．

具体的な製品比較の観点は，前提とするデータベース設計アプローチ，O/R マッピングのための Java 標準 API である Java Persistence API（JPA）への準拠，対応するデータベース管理ソフトウェアの種類とバージョン，設定ファイル等の作成負荷，クラスタリング対応，SQL 自動生成，SQL 発行タイミング，任意 SQL の実行の可否，排他制御方式，実装・設定難易度や開発支援ツールの有無，入手可能な技術情報の質と量などである．

この層で調査対象とした OSS は，Hibernate[*14]，iBatis[*15]，Cayenne[*16] の三つである．

調査の結果，これらの OSS はそれぞれが異なるアプローチ・機能を採用していることが判明したが，2007 年 9 月現在で標準仕様 JPA に正式対応しているのは Hibernate のみであった．

2. 2　OSS 基本パターンの策定と利用推進

OSS ベースのフレームワークに関する知財蓄積・活用の促進や，これらを通じた開発品質と生産性の向上に向けては，プロジェクト毎に異なる製品を採用するのではなく，推奨する OSS の組み合わせを規定し，標準化を図る必要がある．この標準を「OSS 基本パターン」と名づけ，2.1 節の調査結果を基に Struts，Spring，Hibernate を選定した．

ただし，この OSS 基本パターンは今回選定した OSS（Struts, Spring, Hiberante）に永続的に固定化すべきではない．なぜなら，OSS には頻繁に新しい機能や仕様が追加されるため，たとえ現時点でデファクトスタンダードな OSS であっても，すぐに別のソフトウェアに取って代わられる可能性があるからである．そのため OSS の技術調査を継続的に実施し，実績や安定性，豊富な機能を兼ね備えた OSS が新たにデファクトスタンダードとなった場合には，その OSS を検証した上で OSS 基本パターンに適時取り込み，組み合わせのバリエーションを増やしていく必要がある．なお，OSS 基本パターンの追加選定においては，選定 OSS の数が増えすぎることで利用技術の蓄積が妨げられることのないように注意する必要がある．

2. 3　Java システム標準アーキテクチャから見た評価

OSS 基本パターンとして，Struts，Spring，Hibernate を推奨する OSS の組み合わせと規定したが，このパターンだけでは非同期処理やバッチ処理などの機能が不足しているため，これらの不足機能について独自に補完実装を行うことで，Java EE システム開発に必要な機能を提供する必要がある．

そこで，Java EE システム全体に要求される機能の可視化を目的として「Java システム標準アーキテクチャ」を作成した（図 2）．Java システム標準アーキテクチャは，図 1 をさらに詳細な機能構成まで分類したものであり，システムの構成要素とその関係を，フレームワークにより共通化可能な部分と，業務固有な部分の二つの領域で表している．このアーキテクチャ

により，独自補完機能が必要となる領域が明確になる．

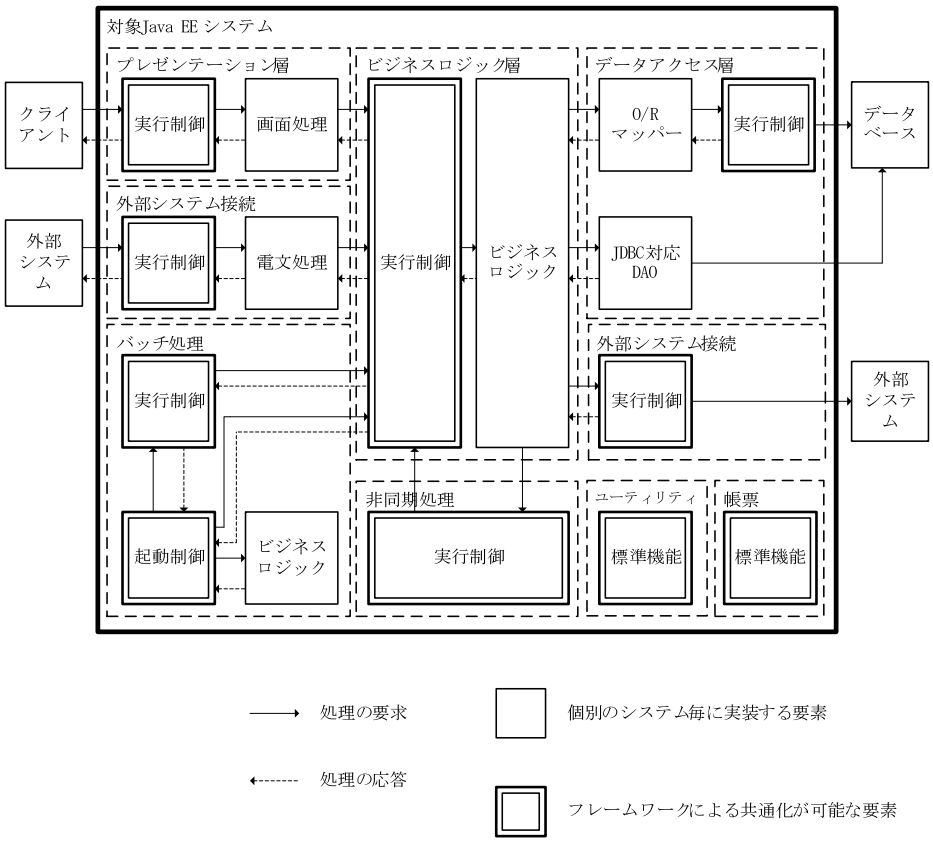

図 2　Java システム標準アーキテクチャ

Java システム標準アーキテクチャには，オンライン処理を担うプレゼンテーション層，ビジネスロジック層，データアクセス層の三層の他にも，一般的にシステムに必要とされる外部システムとの接続やバッチ処理，非同期処理，帳票処理などの詳細構成が含まれている．

3.　OSS ベースの Java EE アプリケーション・フレームワーク「Maia」

本章では，Maia の概要や特徴の紹介と，内部構造の技術解説を行う．

3. 1　Maia の概要とその特徴

OSS 基本パターンと Java システム標準アーキテクチャを基に，OSS 基本パターンとして選定したソフトウェアだけでは不足している機能を独自に補完する形で，研究開発を進めているのが，アプリケーション・フレームワーク「Maia」である．

2007 年 9 月現在において，Maia は図 1 で示した Web システムを構成する八つの機能要素のうち，オンライン処理基盤である三層（プレゼンテーション層・ビジネスロジック層・データアクセス層）と主要なユーティリティ機能を実装済みである．（図 3）

Maia の特徴とそれによる利点は以下の通りである．

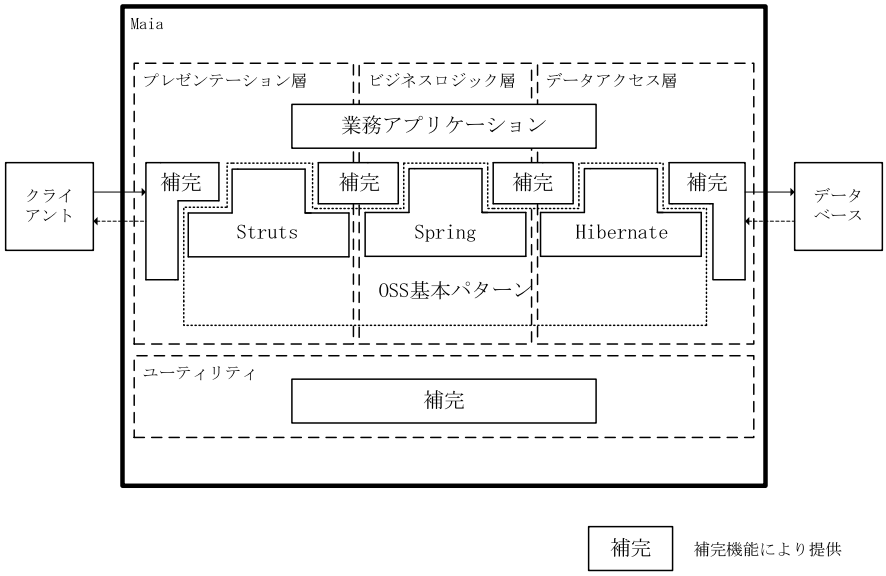

図 3　Maia の構成概要図

1）　デファクトスタンダード OSS の活用

OSS 基本パターンとして実績の豊富な OSS が利用されるため，開発者の技術習得コストの軽減や品質の確保，開発生産性の向上などの，一般的な OSS 適用のメリットを享受することができる．

2）　Java システム標準アーキテクチャ策定

Java システム標準アーキテクチャでは，Java EE システムをモデル化し，各製品・各層の機能範囲を明確に定義しており，将来的に大規模ミッションクリティカルシステムを目指すことを視野に入れている．

3）　OSS の不足機能に対する独自の補完

OSS だけではカバーできない不足機能の補完については，大規模システム開発案件の実績を多数有する統合開発フレームワーク MIDMOST for Java EE[5]から，機能やノウハウの継承を行い，実現している．

また，補完機能では OSS の連携について設定・利用を簡略化する機能・指針も提供しており，複雑な実装や設定を回避することで，開発生産性の向上を図っている．

3. 2　Maia の内部構造

図 3 で示す通り，現時点で Maia はプレゼンテーション層，ビジネスロジック層，データアクセス層，そしてユーティリティによって構成される．ユーティリティが提供する機能は，必要に応じて他の層から利用することができる．

Maia では基本 OSS パターンとして Spring を採用しているため，アスペクト指向プログラミング[*17] と DI コンテナ機能を利用することができる．DI コンテナ機能により，各層の相互依存性を低く抑えることができるため，保守性の向上や単体テストの効率化が期待できる．

3.2.1 プレゼンテーション層

プレゼンテーション層は，クライアント（Web ブラウザ）からの処理要求を受けて，ビジネスロジック層の業務処理を実行し，その結果を Web ブラウザに表示する．Maia はプレゼンテーション層に，Struts と Spring を採用している（図 4）．

プレゼンテーション層の Maia 補完機能としては，文字コード処理，ログ処理，例外処理や Struts・Spring 連携の実装や設定を簡略化する機能・指針等がある．

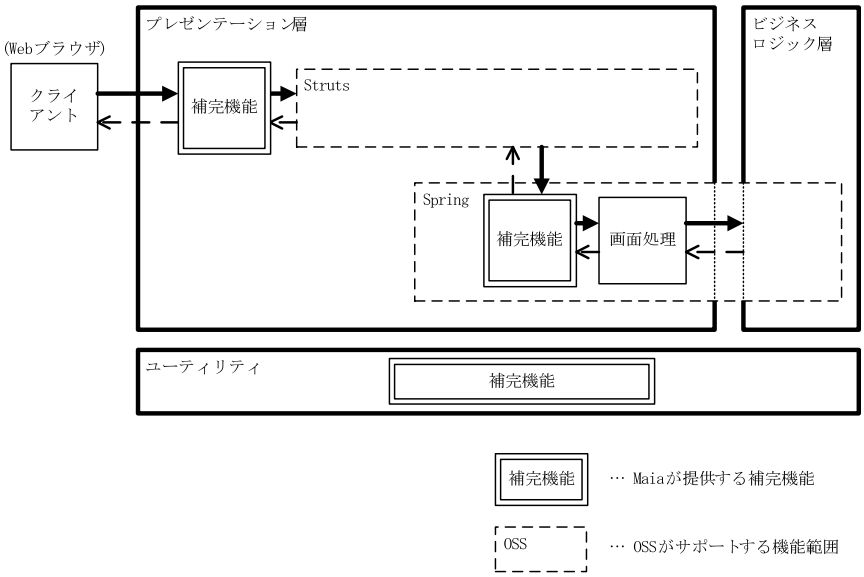

図 4　Maia プレゼンテーション層の構成

3.2.2 ビジネスロジック層

ビジネスロジック層では，データアクセス層を通じデータベースへの処理を行いながら，業務処理を実行する．Maia はビジネスロジック層に Spring を採用している（図 5）．

ビジネスロジック層の内部は，大きな括りで業務処理を実行する「ファサード」と，詳細な業務処理を実行する「ビジネス」の二階層に分けて定義している．ビジネスロジック層の二階層化により，業務処理の粒度に基づいて機能を分けることができ，また補完機能によって，この単位でトランザクションを分割することが可能である．

ビジネスロジック層における主要な補完機能としては，トランザクション制御機能，ログ処理，例外処理などがある．

3.2.3 データアクセス層

データアクセス層は，ビジネスロジック層からの要求に従い，DAO（データ・アクセス・オブジェクト）を通じて，データベースへアクセスし処理を行う．Maia はデータアクセス層に Hibernate と Spring を採用している（図 6）．

データアクセス層における補完機能としては，ログ処理や例外処理などがある．

データアクセス層で採用している Hibernate は，プレゼンテーション層やビジネスロジック層などの OSS に比べて習得コストが高く，またデータベース設計アプローチにも影響するこ

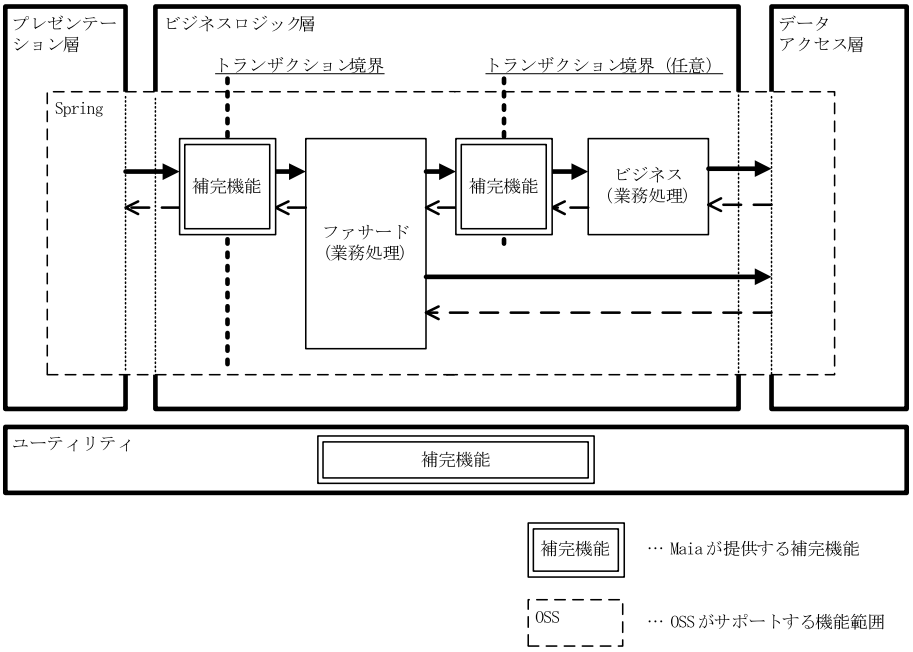

図5　Maiaビジネスロジック層の構成

とから，Maia ではHibernate のシステム開発への採用が適切でない場合の選択肢として，JDBC（Java Database Connectivity）によるアクセス方式も提供している．これら二つの方式は，開発するシステムの特性に合わせて決定する必要がある．

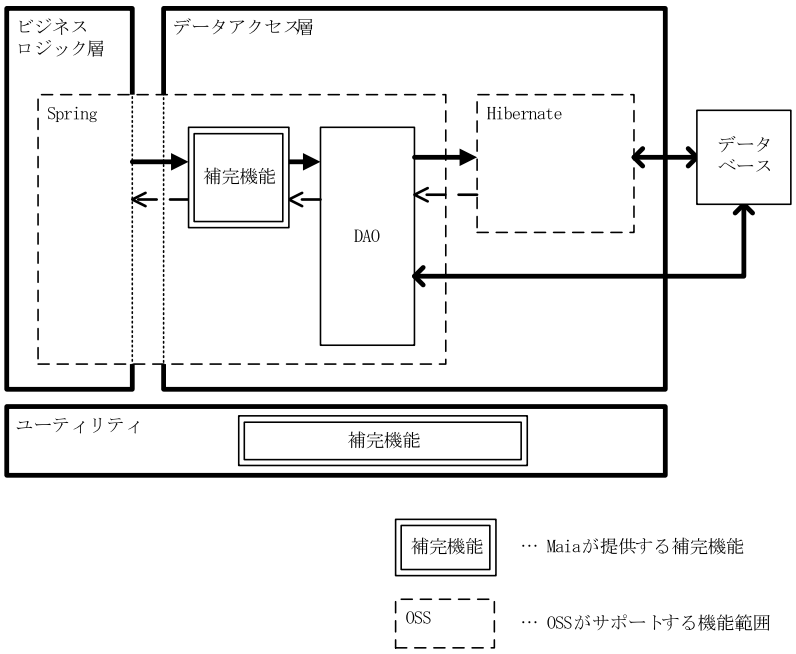

図6　Maiaデータアクセス層の構成

3.2.4 ユーティリティ

Maia では，多くのシステムに共通する機能をユーティリティとして提供している．これらの機能の多くは MIDMOST for JavaEE を参考に実現している．具体的な各機能は以下の通りである．

1) ログ出力機能

運用監視ログ，アプリケーションログ，パフォーマンス測定用ログ，発行 SQL トレースログの四種類のログ出力機能を提供する．また各ログ出力機能では，出力する内容やフォーマットなどを任意に設定することができる．

2) メッセージカタログ機能

アプリケーションがログや画面などに出力するメッセージを，外部ファイルに保持し管理することができる．この機能によって，外部ファイルからメッセージを取得し，またメッセージの置換や変換などを行うことが可能となっている．

3) 例外処理

深刻度と種別に応じて，計七種類の例外を定義しており，各例外発生時の処理を別々に設定・実装し，例外をハンドリングすることができる．

また，Hibernate 等のソフトウェアでは，内部で独自の例外を発生させることがあるため，これらの独自例外を Maia 準拠の例外に変換する機能を提供している．例外の変換機能は任意に設定することができる．

4) スレッドコンテキスト

ユーザ情報などのシステム全体で使用する情報を格納する領域を，スレッドコンテキストとして提供する．スレッドコンテキストには，クライアントからのリクエスト単位で初期化されるタイプと，セッション単位で初期化されるタイプの二種類を提供する．

5) イニシャライザ

イニシャライザを実装・設定することで，フレームワーク起動時（アプリケーションサーバ起動時）に任意の処理を実行することができる．なお Maia では，ログ出力機能とメッセージカタログ機能に対する初期化処理が，標準イニシャライザによって実行される．

4. お わ り に

本稿では，OSS 適用に関する課題から，Maia の開発背景，その概要と内部構造について紹介した．Maia は，「検証済みのプロダクトセット」，「適用方法論」，「サービス」から構成される基盤プラットフォームサービス AtlasBase のアプリケーション・フレームワークの一つとしてシステム開発へ適用される予定である．

独立行政法人情報処理推進機構（IPA）などでも OSS 技術者の育成が重要視されている昨今[7]，OSS の有効活用はシステム開発において無視することができなくなっている．また今後の OSS の成長に関しても，経済学の観点からその成長を考察した興味深い論文[8]が発表され

ており，OSS のさらなる発展が期待される．

OSS ベースのフレームワークである Maia については，今後のさらなる発展を目指して機能拡張・開発を行っていく予定である．機能拡張は，実システム適用事例からのフィードバックを基に，より実践的な機能を目指し継続的に実施していく．また機能開発では，未実装の四層（外部システム接続，非同期処理，帳票，バッチ処理）について，OSS の技術調査を進め，特に開発要求・依頼の多い外部システム接続（Web サービス）から開発していく予定である．

また機能拡張・開発と同様に重要なのが，システム開発現場への Maia 適用と Maia 技術者の育成である．なぜなら Maia は知財の蓄積と活用を大きな目的の一つとしており，開発現場への適用と現場からのフィードバックによりその完成度を高めていくからである．Maia の開発と技術者の育成を行いつつ，利用技術の蓄積と活用を効率的に実施することで，その完成度を高めていきたいと考えている．

本稿が Maia の理解と普及推進の一助となれば幸いである．

＊ 1　Total Cost of Ownership : 資産の購入から廃棄までに必要な時間と支出の総計
＊ 2　JavaServer Faces Technology, http://java.sun.com/javaee/javaserverfaces/
＊ 3　複数のページ間などで標準的なレイアウトと概観を規定する機能
＊ 4　MyFaces, http://myfaces.apache.org/
＊ 5　Facelets, http://facelets.dev.java.net/
＊ 6　Shale, http://shale.apache.org/
＊ 7　Tapestry, http://tapestry.apache.org/
＊ 8　Mayaa, http://mayaa.seasar.org/
＊ 9　Click, http://click.sourceforge.net/
＊10　Struts, http://struts.apache.org/
＊11　インターフェース中心の設計により，クラス間の依存関係をソースコード上に直接的に記述するのではなく，外部設定ファイル等により制御できるようにしたソフトウェア
＊12　Spring, http://www.springframework.org/
＊13　Seaser2, http://www.seasar.org/
＊14　Hibernate, http://www.hibernate.org/
＊15　iBatis, http://ibatis.apache.org/
＊16　Cayenne, http://cayenne.apache.org/index.html
＊17　ソフトウェアの特定の振る舞いを「アスペクト」として分離し，機能を分離化するプログラミング技法．オブジェクト指向プログラミングの問題点を補う．

参考文献　［1］　ソフトウェア情報センター（SOFTIC），「オープンソース・ソフトウェアの現状と今後の課題について」，情報処理振興事業協会（IPA），2003.8，http://www.ipa.go.jp/archive/NBP/14nendo/14cho1/030815opensoft.pdf
［2］　CSAJ/OSS 普及推進研究会 ,「オープンソースソフトウェアに関する意識調査」，社団法人コンピュータソフトウェア協会，2006.6，http://www.csaj.jp/committee/oss/h17_oss_report.pdf
［3］　マーストリヒト国連大学 MERIT，Economic impact of open source software on innovation and the competitiveness of EU ICT sector，欧州委員会，2006.11，http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf
［4］　David A.Wheeler, Why OpenSourceSoftware/FreeSoftware? Look at the Numbers!，2007.4，http://www.dwheeler.com/oss_fs_why.html
［5］　渡辺弘美 ,「米国政府におけるオープンソース・ソフトウェア導入を巡る動き」，JETRO/IPA NY，2005.5，http://www.jif.org/column/pdf2005/200505.pdf
［6］　新井敦，「日本ユニシスにおける開発標準の策定と適用への取り組み」，ユニシス技報，日本ユニシス，Vol.27 No.2 通巻 93 号，2007 年 8 月
［7］　IPA OSS センター，「OSS 技術教育のためのモデルカリキュラムに関する調査」，2006.5
http://www.ipa.go.jp/software/open/ossc/seika_0605.html

［8］ Giampaolo Garzarelli, Open Source Software and Economic Growth, 2007.6, http://opensource.mit.edu/papers/0SSGROWTHMIT.pdf
［9］ 吉野良成,「システム構築プロジェクトの現状と課題」, ユニシス技報, 日本ユニシス, vol.27 No.2 通巻 93 号, 2007 年 8 月

執筆者紹介 山 口 周 志（Shuji Yamaguchi）
2004 年 北海道大学大学院理学研究科修士課程修了. 同年 日本ユニシス(株)入社. 金融案件を中心に各業種のシステム開発プロジェクトに対して, 提案支援・技術支援を手掛ける. 現在, 総合技術研究所 OSS センターに所属し, アプリケーション・フレームワーク「Maia」の研究・開発と適用, 開発プロジェクトへの技術支援を担当.