解説 5

# IBMサービス事業でのOSSへの取り組み
## ― サービス主導によるOSSの利用 ―

**オープンソース・ソフトウェア（以下、OSS）は、自由で自律的な進化を続け、先進的な技術分野をリードするだけではなく、ミッション・クリティカルな分野でも積極的に利用されています。しかし、サービスの現場で OSS の効果を最大限に得るには、サービス対象である「情報システム」が進化する方向性に合わせて OSS の利用法を最適化する必要があります。IBM では、Linux® などソフトウェアとしての OSS だけでなく、サービスの一環として、フレームワークや開発支援ツール、OSS の組み合わせの検証などに取り組んでいます。そして、OSS 活用のベスト・プラクティスを確立するとともに、システムの開発者とシステムの保有者であるユーザー企業との間に生じがちな、OSS に対する認識の溝を埋めることを目指しています。**

Article 5

## Deploying OSS in Business Services
### - Using OSS in Service-centric Views -

OSS has become the technical driver of enterprise applications, and not only the new glowing technologies. But deploying OSS in business services still has some difficulties and risks. Finding the best combination and optimum usage of OSS are required for system innovation, which is the goal of the business service. For business services, we continue to make efforts to test the combination of the framework, development tools, and OSS which are used widely and practically in many business services in IBM. Our goal is to establish a best practice and mediate the gap in the stance towards OSS between developers and system owners.

## ❶ 進化を続ける OSS

例えば OS では Linux、ミドルウェアでは HTTP サーバーの Apache、DBMS（Database Management System）の MySQL/PostgreSQL などがその良い例です。また、OSS はアプリケーション開発サービス（以下、サービス）の現場にも大きな影響を与えています。Java™ 言語を例にとると、フレームワークである Struts（画面制御）/Spring Framework（DI/AOP コンテナー）/Hibernate（DB アクセス）、開発支援ツールである JUnit（単体テスト）/Eclipse（IDE：統合開発環境）などは世界中で利用されています。近年では、OSS がある対象領域を席巻し、その領域に以前より存在した標準仕様にまで影響を与える現象もしばしば見られるようになっています。さらに、Web 2.0 をはじめとした先進的な分野では、商用ソフトウェアをしのぐスピードで、新たな OSS が次々と生まれています。

このような OSS の普及と進化の原動力は、開発者主導のオープンな技術コミュニティーです。例えば、ある開発者が先進的な技術を実装して公開することで、ほかの開発者はそれを安価または無料で利用することができます。もし不足している機能があれば、コミュニティーに対応を要求するか、公開されているソース・コードを利用者自らが拡張して目的にあった機能を実装し、必要があれば公開することになります。このように、OSS は複数の開発者により頻繁に改良され、優れた改良があれば次のリリースに反映されていきます。特定のベンダーに縛られず、自由で自律的な分、OSS は商用ソフトウェアよりも活発に進化するということができるでしょう。

一方、サービスの現場で OSS の効果を最大限に得るには、その理念や前提環境、開発方針など、幾つかのポイントを踏まえ、サービスの対象である「情報システム」が進化する方向性に合わせて利用法を最適化する必要があります。方向性のそろった OSS を長期間使用することでノウハウが蓄積し、繰り返し利用するたびに生産性は向上します（**図 1**）。逆に、方向性の異なる OSS を利用しながらシステム全体の方向性を一貫させるには、サービスの各場面において相応の工夫が必要となります。

## ❷ サービスの場面と OSS

ここではサービスの各場面において、OSS の利用にあたって検討・注意すべきポイントを解説します。

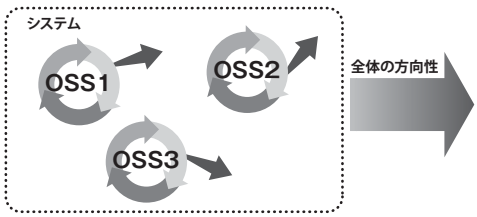


図 1. OSS とシステムの方向性

**【方針・計画／要件分析】**

OSS の中には、その利用者にも著作権の放棄を要求する（コピーレフト）ライセンス形態を持つものがあります。これらのライセンスは、著作権の独占を避けることでソフトウェアの進化を促すことを目的としており、商用ソフトウェアとは進化に対する考え方がそもそも異なります。システムの保護に必要な権利を確保できるかどうかを、遅くとも利用を開始するまでには把握しておく必要があります。

また、OSS は要員計画にも影響を及ぼします。商用ソフトウェアにも同様のことがいえますが、利用するソフトウェアに対する開発者の熟練度は、サービスの生産性と品質に大きな影響を与えるため、要員の手配ができるだけ容易な OSS を事前に選別しておくことが望まれます。

**【設計】**

設計の場面では、OSS 間および商用ソフトウェア間の組み合わせ（アーキテクチャー）について詳細な検討が必要となります。可能であれば成功事例のある組み合わせを選ぶことが理想ですが、それがない場合は独力で検証をしなくてはなりません。事例の調査自体が難しいこともあります。

また、全社的な情報システム標準（エンタープライズ／テクノロジー・アーキテクチャー）あるいは社内にほかのシステム事例がある場合、それに沿った形で OSS を選定・配置することで、システム間の親和性が向上し、再利用によって開発効率も向上します。

さらに、後続の実装／テストに向けた開発者用ガイドの整備にあたっては、OSS が提供しているドキュメントの種類、数、言語などがガイドの優劣（開発効率）を左右します。

**【実装／テスト】**

実装／テストの場面では、OSS の品質やサポートのリスクが顕在化することがあります。実際に、ある OSS の JavaScript™ ライブラリーによる画面操作が遅延する現象が多発し、サポートも受けられなかったため、実装／テストの終盤で別のライブラリーに変更した例が報告されています。このような事態を避けるためには、OSS の選定と検証に十分な開発資源を投入しておかなければなりません。

**【保守】**

保守の場面では、OSS の仕様変更への対応が求められることがあります。先進的な技術分野における OSS は、その変化の範囲も頻度も大きくなる傾向にあります。場合によってはその OSS が淘汰（とうた）されてしまうこともあります。OSS 選定の際に、その技術分野の動向や成熟度合いを考慮しなければなりません。

表 1. サービスの場面と検討対象

| 方針・計画／要件分析 | 設計 | 実装／テスト | 保守 |
|---|---|---|---|
| ライセンス<br>要員計画 | 組み合わせ<br>標準化<br>開発ガイド | サポート<br>品質 | 仕様変更 |

## ❸ サービス主導による OSS の利用

サービスを起点に考えると、これまで述べてきたリスクに対応するには、システムの進化に役立つ OSS を見極め、それらを組み合わせて評価・検証した上でサービスに取り込むことと、取り込まれた OSS をサービスに従事する開発者がどのような形で使用すべきなのかということの二つの観点が必要となります（**図 2**）。

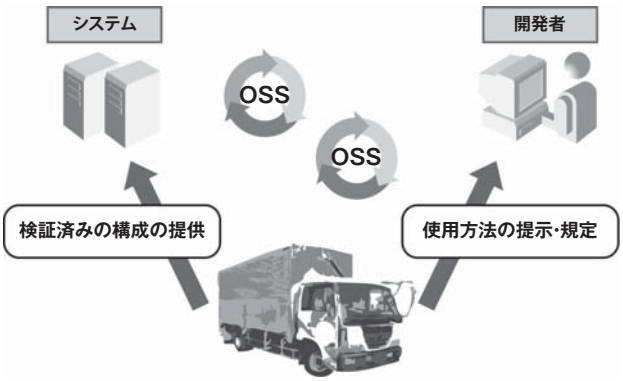


図 2. サービスを中心にとらえた OSS の利用

本稿では、これらの二つの観点から IBM のサービスにおいて OSS を利用する際のリスクにどう対応しているのかを、Web application components の場合を例としてご紹介します。Web application components は、Web アプリケーションの開発基盤を提供する IBM の知的資産であり、IBM のサービスを通じて 250 近くに及ぶシステムで再利用されている実績を持ちます。

**【OSS の選定】**

Web application components の構築要素はライブラリーと開発ツールとに大別されます。ライブラリーの機能としては、同様の領域をカバーするデファクト・スタンダードの OSS である Struts と重複するものも多く、両者はしばしば比較の対象となってきました。そこで、Web application components のアーキテクチャーを見直し、Struts に対する追加機能に相当する部分を、Struts と組み合わせた形でも同様に利用できるように改良しました。

また、OSS の DI/AOP コンテナー（アプリケーション内の部品間の依存関係を解決するライブラリー）である Spring Framework の機能により、Web application components を使用して開発したアプリケーションの構成をより柔軟にし、開発および維持の効率を向上する試みを続けています。

ライブラリーについては、開発の完了後も、サーバー上でアプリケーションとミドルウェアの間に位置する形で使用され続けます。従って、OSS を使用する場合にも、長く使われることを考慮した評価が必要です。具体的には、以下のような点を考慮しています。

### ■ ライセンスの考慮

IBM では、OSS のライセンス形態から発生するリスクに対し、OSS の使用方法をリスクのレベルに応じた形で詳細に規定することで対応しています。Struts については、すでに WebSphere® Application Server の管理コンソールに使用されており、製品に同梱されている実績があることから、Web application components においても Struts を同梱することで承認を得て使用しています。

### ■ 仕様のオープン性の考慮

ソース・コードが公開される OSS では、それ自体の仕様はオープンなものとなりますが、OSS の前提環境の構成要素にはブラック・ボックスとなるものが含まれる場合があります。環境の前提をオープンな規格により説明でき、動作検証がより多くの環境で実施されている OSS を選定することで、仕様に関して正確な情報を得ることが容易になり、ほかのライブラリーとの組み合わせによる使用時のリスクも軽減されます。

OSS の場合は、維持管理をしている団体が情報発信に積極的であるかどうか、情報発信を続けるだけの体力を維持できているかどうかも重要な要素となります。Struts の Web サイト [1] や Spring Framework の Web サイト [2] からは、いわゆるドキュメンテーション以外にもさまざま情報が発信されています。

### ■ デファクト・スタンダードの考慮

習熟度が高い開発者をより多く集める上では、情報の入手が容易であることに加え、実際によく利用されていることを選定基準に含めることが有利に働きます。

デファクト・スタンダードの OSS に対しては、多くの利用者の注目が集まるため、開発ツールによるサポートの充実や、プロダクト改良の高速化といった効果も期待されます。

特に、グローバルな視点から最適化された開発を実施する上では、日本国内での実績だけではなく、特に IT 産業の発展が見込まれる国での利用実績を考慮することが必要となります。Struts と Spring Framework のどちらについても、それぞれが対象とする領域の OSS として、海外でも高い知名度を持っています。

### 【OSS の組み合わせの事前検証】

Web application components の前提となる規格は、エンタープライズ Java の標準である Java EE™ です。また、継続的な動作検証の環境には WebSphere Application Server を使用しています。このために、WebSphere Application Server との親和性の高い OSS を優先的に使用することが、組み合わせて使用する際のリスクを低減することにつながっています。

Struts については WebSphere Application Server の管理コンソールに使用されていますし、Spring Framework では、WebSphere Application Server を実行環境として認定していますので、リスクの低い組み合わせを選択していることになります。

開発が活発に進められている OSS では、改良の迅速さが期待される反面、急速な変化によりバージョン間の互換性の維持が難しくなる場合があるため注意が必要です。また、Web application components の開発において実施しているように、実際に組み合わせた際の動作検証をリリース準備手順に含めることも必要になります。

### 【OSS を含めた標準化をサポート】

事前に動作を検証されたデファクト・スタンダードの OSS であっても、開発現場で使い方を誤ってしまえばトラブルの種になることはいうまでもありません。特に大規模な開発プロジェクトにおいて、OSS などを含めた外部ライブラリーの使用個所についての標準化を怠った場合には、開発の手戻りや運用時のトラブルにつながる危険性が高まります。具体的には、以下のようなケースが想定されます。

- 使用しているクラスや API などが開発者の趣向、スキル・レベルによって異なるために機能差異やパフォーマンスなどの品質差異が発生してしまう。
- 単体テストに必要なスタブ・データが保守できておらずテスト・ケースを使い捨ててしまう。
- システムごとに違う OSS を使用しているために開発、保守要員の横展開ができない。

OSS は大規模プロジェクトにおける開発の標準化までは面倒を見てくれませんし、ほとんどの OSS は実行ライブラリーの

みを提供し、開発支援ツールや仕様書のテンプレートを提供していません。そこで、Web application components では、システム共通機能のコンポーネント化や仕様書からコードを自動生成する開発ツールの提供などの標準化手法を活用し、アプリケーションの作り込み個所を削減することで、OSS の不適切な利用というリスクを抑えています。さらには、システム横断的にエンタープライズ・アーキテクチャーを標準化することで開発、保守要員の横展開を促進、IT ガバナンスを強化しています（**図 3**）。

Web application components が採用している具体的な手法を以下に列挙します。

- Struts の ActionForm、Commons Validator［3］のバリデーター定義ファイル、および JSP™ のツールによる自動生成
- セッション管理機能、画面遷移履歴管理機能、認証機能などを Struts にプラグ・インできる形で共通コンポーネント化
- Spring Framework によるデータ・ソース管理や宣言的トランザクションに対応したデータ・アクセス・オブジェクトのツールによる自動生成
- Dojo Toolkit［4］などの Ajax フレームワークとの高い親和性を持つ Comment Filtered JSON 形式へのレンダリング機能の共通コンポーネント化

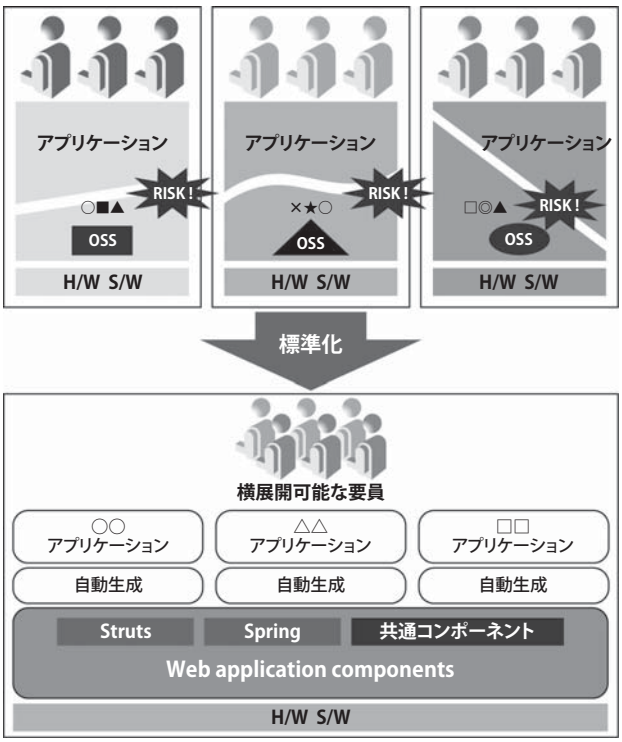


図 3. 標準化による OSS 利用のリスク軽減

- Log4j［5］や Commons Logging［6］の API の差異を隠ぺいした共通インターフェース層の提供
- JUnit の TestCase クラスとそのスタブ・データのツールによる自動生成
- FindBugs［7］と CheckStyle［8］のユーザー・インターフェースおよびルール定義を統合した Java コード解析ツールの提供
- Eclipse［9］が規定するプラグ・インの配布およびバージョン管理に準拠した形態の開発ツール群の提供

開発者のスキルにまったく依存しない開発を実現することは困難ですが、OSS の利用個所を含め、手作業による開発部分を削減し、開発者へのスキル要求を低減することは可能です。このことは、特に大規模な開発プロジェクトでは品質面で大きな威力を発揮します。OSS そのものの安全性や実績を確認するプロセスを実施するだけでなく、標準化されたアプリケーション基盤、開発環境、ガイド類を提供し、開発者へのアフター・フォローを充実させることが、サービスにおける OSS の利用の重要なポイントなのです。

## ❹ まとめ

OSS はシステム開発で幅広く利用されるようになってきており、開発者を中心とするコミュニティーから多くの支持を集めています。しかし、サポートがない点やライセンスが複雑などの問題から、OSS の使用に不安を持つ担当者の方が少なくないのもまた事実です。このことは、OSS の適用にあたってリスクを減らし、開発者とユーザー企業との間にある溝を埋める役割を誰かが担う必要があることを示唆しています。

開発者とお客様の間に立つ IT ベンダーは、本来この役割を担うべきだといえます。IBM ではこの考えに基づき、これまでに説明してきたように、Linux などのソフトウェアとしての OSS だけでなく、SI サービスの一環として、フレームワークや開発支援ツール、OSS の組み合わせの検証などに取り組んでいます。IBM は数多くの経験と実績に基づいて OSS 活用のベスト・プラクティスの確立を目指しています。その目的は、OSS の持つリスクを最小限化するとともに、OSS の持つ魅力を最大限に発揮できる組み合わせ、拡張を提供することにあります。こうしたベスト・プラクティスをシステム開発に適用することで、お客様にとってはシステムの品質、コスト、納期においてメリットがあり、IBM にとってはより実績を積むことができるという Win-Win の関係をもたらします（**図 4**）。

近年ではシステム開発において海外拠点での開発化が進んでいますが、今後もますますグローバル化が推進さ

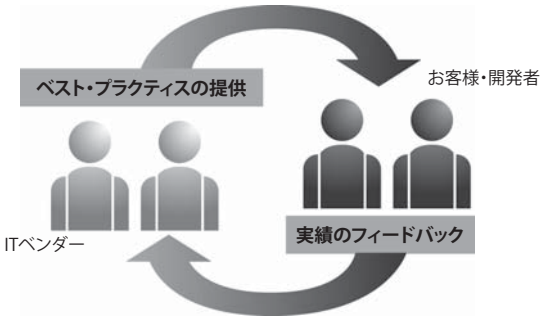


**図 4. お客様と IT ベンダーのあるべき姿**

れていくことは避けられません。こうした状況においても、OSS からは、そのオープン性による情報の入手しやすさ、開発ノウハウの共有、要員確保の容易性など数多くのメリットを得ることができます。

もし、OSS に対して、サポートがない、ライセンスの問題の懸念が捨てきれない、などのネガティブなイメージを現在お持ちの方がいらっしゃいましたら、そういう方にこそ OSS を利用した開発をお勧めします。IBM などが提供する適切なサポートを受けることで、それらのイメージが間違いであったということがお分かりになると思います。今後のシステム開発において、OSS を活用したビジネスをご検討対象に入れてみてはいかがでしょうか。

**[参考文献]**

[1] Struts, Apache Software Foundation, http://struts.apache.org/
[2] SpringSource, SpringSource Inc., http://www.springsource.com/
[3] Commons Validator, Apache Software Foundation, http://commons.apache.org/validator/
[4] Dojo Toolkit, Dojo Foundation, http://dojotoolkit.org/
[5] Log4j, Apache Software Foundation, http://logging.apache.org/log4j/
[6] Commons Logging, Apache Software Foundation, http://commons.apache.org/logging/
[7] Find Bugs, http://findbugs.sourceforge.net/index.html
[8] Check Style, http://checkstyle.sourceforge.net/
[9] Eclipse, The Eclipse Foundation, http://www.eclipse.org/

**日本アイ・ビー・エム株式会社**
**GBS ソリューション&アセット**
**アドバイザリー IT スペシャリスト**

白須 英治 Eiji Shirasu

**[プロフィール]**
2000 年日本 IBM 入社。入社以来、Java を使用した Web システムの提案サポート、設計、開発などに従事。「Web application components」に関しては主にフレームワーク部分の設計／開発を担当。

**日本アイ・ビー・エム株式会社**
**GBS ソリューション&アセット**
**アソシエイト IT アーキテクト**

木村 迅 Jin Kimura

**[プロフィール]**
2001 年日本 IBM 入社。入社以来、Java フレームワーク・開発支援ツール「Web application components」の設計／開発を担当。同フレームワーク・ツールの現場展開を通じて知的資産の収集と適用を推進すると同時に、某都市銀行様の Web 系システムにおけるアプリケーション・アーキテクチャーの標準化や構築を手掛けている。

**日本アイ・ビー・エム株式会社**
**GBS ソリューション&アセット**
**アドバイザリー IT スペシャリスト**

本多 一行 Kazuyuki Honda

**[プロフィール]**
2004 年日本 IBM 入社。入社後、Web システムに関して業種をまたいで、設計、開発、運用などさまざまな局面で幅広くプロジェクトに参画。現在は社内 Java EE フレームワークである「Web application components」の設計／開発／サポートを担当。

**日本アイ・ビー・エム株式会社**
**GBS ソリューション&アセット**
**IT スペシャリスト**

水島 壮太 Sota Mizushima

**[プロフィール]**
2005 年日本 IBM 入社。入社後、Web 基盤構築アセット群「Web application components」の設計／開発を担当。現在は金融業界のプロジェクトを中心に Java 関連の技術支援、および「Web application components」の Ajax 対応や提案サポートに従事。