軽量型Webフレームワークの標準策定と実用化

テクニカルコンピテンシー部
開発・品質技術チーム
ITスペシャリスト

広戸　裕介

Yusuke Hiroto
yusuke-hiroto@exa-corp.co.jp

標準が定着しない主な要因として、標準の作成側と利用側での認識の差異がある。ここに着目して、今回Web開発の標準化を目的として構築した「軽量型Webフレームワーク」を実プロジェクトに適用することで、遭遇した実際の課題を明らかにし、解決に取り組んだ。これにより、対象プロジェクトでは標準の適用をスムーズに行うことができ、同時に標準もより実用的に改善することができた。

1. はじめに

当社では、システム開発に適用するプロセス、成果物、製品・技術、ツール類など、様々な対象についての標準化を進めている。その一環として、昨年度からWebアプリケーション開発に利用するフレームワークの標準化に取り組んできた。

近年、当社で扱う開発案件は、Javaを使用したWebアプリケーション開発が多い。そして、オープンソースのフレームワーク製品の数と質が充実してきた昨今、このようなJava・Web開発案件においては、何らかのフレームワーク製品を利用することが当然のこととなっている。これらを背景に、お客様からはフレームワークに関する技術提案を含めたRFPを求められる例も出てきている。

オープンソースのフレームワーク製品が次々と生まれることは、製品間の競争と学習を加速させ、より高品質な製品を生み出す原動力となるため、SI業界全体としては歓迎すべきことである。しかし以下の理由から、開発者一人ひとりの立場からは必ずしも良いことばかりとは言えない。

- 習得すべき技術の数が増え、キャッチアップが難しくなっている
- フレームワーク製品の数が多く、習得した技術が、他のプロジェクトにも通用できる機会が減少している
- フレームワーク製品の変遷が著しく、習得した技術が、いつまで通用するのか不透明である

これらは、開発者個人が万遍なく技術を習得することを困難にするため、開発者全体としては技術の専門分化が進むことを意味する。言い換えれば、技術に対する属人性を高める結果を生んでいる。技術の属人性が高まると、次のような問題が発生する。

- プロジェクトと開発者の結びつきが強くなり、人材の流動性が悪くなる
- 必要になる都度、個別に知識習得が必要となるため、プロジェクトの費用対効果が悪くなる
- プロジェクトのレビューのために、レビューアが数々の技術を習得しなければならなくなる

これは、開発者だけでなく全社的な立場からも、避けるべき状況であると言える。そのため、精選した特定の製品で社内標準化を進めることが大きな意義を持つことになる。

本稿では、大きく二つの段階に分けて、標準フレームワーク実用化の取り組みを述べる。まず第２章で標準策定までの活動を述べる。次の第３章では、標準フレームワークのプロジェクトへの適用事例を取り上げ、標準作成者と利用者の間の認識の違い、それに起因する実用化課題と解決策について述べる。最後に第４章で、標準策定時に考慮すべき点をまとめる。

2. 標準の策定

この章では、社内実績調査に基づく標準フレームワークの開発と関連資料整備までの標準化作業について述べる。

2.1. 社内実績の把握とフレームワークの開発

まず、標準として採用すべきフレームワーク製品を選定する際の判断材料を得るため、社内実績を該当プロジェクトの開発リーダにヒアリングした。この結果、次の３種類の製品が複数プロジェクトで採用されていることが判明した。

- WebフレームワークとしてStruts Framework（以下Strutsと称す）[1)]
- DI/AOPコンテナとしてSpring Framework（以下Springと称す）[2)]
- RDBラッパーとしてiBATIS Data Mapper Framework（以下iBATISと称す）[3)]

これら３種類の製品を軸に多面的な検討・評価を行い、Struts・Spring・iBATISの組み合わせを標準として採用した。Strutsは開発者の確保しやすさと社内に抱える資産の多さ、Springはオフショア開発を考慮した場合の世界的な知名度の高さ、iBATISはオブジェクト指向開発に限らない適用範囲の広さ、という点がそれぞれの製品を採用した主な理由である。

次に、Struts・Spring・iBATISの組み合わせを採用したプロジェクトで構築したフレームワークのソースコードをベースとして、プロジェクトに特化した部分を取り除き、また必要な部分は改良を加えて、標準フレームワークを構成した。さらに、フレームワーク本体の開発だけではなく、標準をスムーズに普及させることを目的としたガイドや教材等の関連資料の整備を行った。

なお、骨格に据えたSpringがEJBコンテナに対比して軽量コンテナと呼ばれていたため、この標準フレームワークを「軽量型Webフレームワーク」と命名した。

2.2. 軽量型Webフレームワークの概要

軽量型Webフレームワークの特徴を表１に示す。標準化の定着に重きを置き、対象プロジェクトが多く、かつシステムの複雑度が少ない中・小規模プロジェクトに対象を絞った。

軽量型Webフレームワークの構造を図１に示す。Web3階層とMVC構造からなる、一般的、かつ基本的なWebアプリケーションシステムの構造である。同図に示すように、フレームワーク製品に留まらず、ミドルウェア製品まで固定することで、プロジェクトへの導入が容易になるように考慮している。軽量型Webフレームワークを構成する製品とその機能を、それぞれ表２と表３に示す。表を見てわかるように、当フレームワークはすべてオープンソースのフレームワーク製品・ライブラリで構成している。このようにオープンソース製品を採用することで、軽量型Webフレームワークは、アプリケーションと開発基盤全体の標準化を目的としている。

また、標準の策定に当たっては作り込みを極力減らしたシンプルなものとなるよう配慮した。その理由は、「プロジェクトへの適用から得られる優れたノウハウを追加することにより、技術ノウハウを標準の中に蓄積し、かつ標準を成長させていくことが標準化のあるべき姿である」との信念からである。このため、初期値としての標準はシンプルなものが望ましいと考えた。

表１　軽量型Webフレームワークの適用対象

	適したシステム対象	適さないシステム対象
規模	Web/APサーバが一台で収まるシステム	大規模（トランザクション数、サーバ数）
アプリ例	経理、（人事部向け）人事、データエントリ、イントラポータルサイト等	生産管理、オンライン証券、銀行の勘定系

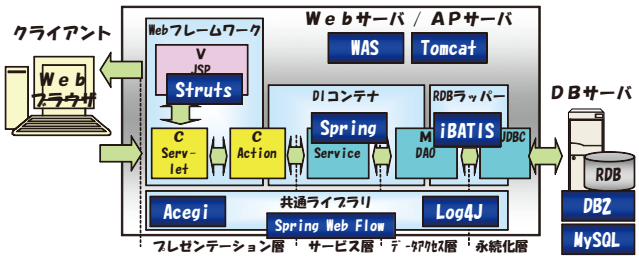

図１　軽量型Webフレームワークのアーキテクチャ

表２　軽量型Webフレームワークを構成する製品（1）

フレームワーク製品	分類	特徴、主な機能
Struts	Webフレームワーク	・Webフレームワークとして最も普及。 ・MVC構造を持ち、画面遷移や入力チェック機能を提供。
Spring	DI/AOPコンテナ	・世界的に最も有名なDI/AOPコンテナ。 ・部品を自由に接続する機構を提供、レイヤ間の接着剤として、全体の組み立てに使われる。 ・宣言的なトランザクション制御機構を提供。
iBATIS	RDBラッパー	・SQLをマッピングの基礎とするRDBラッパー。 ・既存のデータベース構造がそのまま利用可能。 ・SQLによるチューニングが容易。

表３　軽量型Webフレームワークを構成する製品（2）

機能	機能の概要	対応製品
入力チェック	・宣言的なvalidateの定義が可能	Struts
画面遷移制御	・ビジネスロジック実行結果による画面遷移制御が可能	Struts
	・URLベースまたはメソッドベースでのアクセス制御の結果により、画面遷移を制御可能	Acegi Security
	・フロー定義ファイルによる、構造的な画面遷移の定義ができる	Spring Web Flow
ロギング処理	・細かなログ設定が可能	Log4J
	・実行時パフォーマンスを考慮	
	・ログの出力先がコンソール、ファイル等様々に選べる	
共通例外処理	・例外の種類ごとに、共通の例外遷移が指定可能	Struts
	・メソッド単位の、特定の例外処理を追加可能	Spring
セッション管理	・Servlet/JSPベースの一般的なセッション機能を提供	Struts
	・Flowスコープ等の独自スコープを利用できる。	Spring Web Flow
トランザクション制御	・指定したメソッドの前後でトランザクション開始・終了できる	Spring
アクセス制御	・URLベースでのアクセス制御 ・メソッドに対してのアクセス制御 ・ロールにより、ハイパーリンクの表示/非表示を切り替え可能	Acegi Security

2.3. 関連資料の整備

2.1節でも述べたように、標準化作業ではフレームワーク本体の開発だけでなく、ガイドや教材等の関連資料も整備した。これにより、フレームワークの導入教育から、開発の実施に至るまでの、すべての局面をサポートできると想定した。以下に作成物を列挙する。

- アーキテクチャ概要
  軽量型Webフレームワークのアーキテクチャの概要を説明した資料。
- 個別部品説明書
  軽量型Webフレームワークを構成するフレームワーク製品・ライブラリや、当社で開発したライブラリを説明した解説書。解説の対象はSpring、iBATISのほか、Acegi、Commons-Logging、Spring Web Flowなどのほかの製品・ライブラリから、当社で開発した汎用Daoと呼ばれる機能までに及ぶ。
- 開発標準類
  開発プロセス・成果物標準（半完成のアーキテクチャ設計書、詳細設計書テンプレートを含む）、コー

ディング規則、単体・結合試験のガイドなど。これらの多くは、別途進めているほかの標準化作業で作成済み、または作成中のものである。軽量型Webフレームワーク標準では、単体テスト方針書を作成している。

- 開発環境ガイド
  必要なライブラリを搭載した開発環境のテンプレート、および、開発環境導入・運用のガイド。軽量型Webフレームワーク本体は、開発者への配布を考慮して、Eclipseプロジェクトのテンプレートとして用意している。また、アプリケーションをWAS（WebSphere Application Server）上にデプロイするためのガイドも作成している。
- 開発者向け教材
  短期間で部品を理解し、使えるようにするための開発者向けのガイド、実装サンプル。軽量型Webフレームワークを使った、実装方法を理解するためのクイックスタートガイドを作成している。ガイドに従い、サンプルアプリケーションを実際に開発しながら、基本的な使い方をマスターすることができる。

3. 実用化

この章では、軽量型Webフレームワークを実際のプロジェクトに適用した事例について述べる。2008年９月現在、軽量型Webフレームワークは４つのプロジェクトで採用されているが、そのうち標準作成者（著者）がフレームワークの適用を支援したプロジェクトを事例として取り上げる。表４にプロジェクトで開発するシステムの概要を示したが、Web/APサーバとDBサーバが各１台で構成されるWebアプリケーションであり、前掲の表１に示した軽量型Webフレームワークの適用対象として誂え向きのシステムと言える。

表４　当該プロジェクトのシステム概要

対象	数量
Web/APサーバ、DBサーバ	各1台
画面	約40枚
テーブル	約10個
帳票、CSVファイル	約10枚
想定ユーザ	30～100名

第２章で説明した標準および関連資料の的確性を検証し、標準をより実用的に改善することを目的に適用支援を行った。具体的に支援した作業として、軽量型Webフレームワークの導入支援、標準が考慮していないプロジェクト固有課題の洗い出しと解決策の選定、および解決策の標準への展開がある。以下では、これら作業の中で明らかになった課題とその解決策について考察する。

3.1. 軽量型Webフレームワークの導入支援

3.1.1. 開発者向けの導入教育

関連資料としてクイックスタートガイドやWASデプロイガイド、個別部品説明書などの教材を整備し、クイックスタートガイドをベースにハンズオン形式の教育を実施した。ここで前提スキルに対する、標準の作成者と利用者間での認識の違いが課題として明らかになった。

クイックスタートガイドは、次のような読者を対象として作成した。

- Webアプリケーションの仕組みを理解している
- Strutsを使った開発経験がある
- DI（Dependency Injection、依存性の注入）[4] やAOP（Aspect Oriented Programming、アスペクト指向プログラミング）[5] を理解している

しかし、当該プロジェクトのメンバーは、必ずしもこの３つのスキルすべてを備えているわけではなく、Web開発経験のまったくない開発者も含まれていた。すでに広く普及しているStrutsはともかく、SpringやiBATISを知らない開発者が未だに多いという事実は、標準作成者が事前に把握、もしくは考慮しておくべき事項であった。

また、一口に軽量型Webフレームワークを使うと言っても、開発リーダと一般の開発者では、軽量型Webフレームワークに求める理解の広さ・深さがまったく異なるということも明らかになった。一般の開発者は、プロジェクト内の開発標準を的確に設定することにより、DIやAOP、さらにはStrutsなどの理解が不十分でも開発に携わることができる。より一般の開発者の実態に合わせた教材が不足した点は大きな反省点である。

ここから、教材を作成する前に、利用者を区分し、区分ごとの前提スキルレベルと目標スキルレベルを特定することが欠かせないということが学べる。利用者の特性を踏まえ、初級者には必要十分な範囲の内容について詳しく説明

し、上級者には幅広く高度な内容を説明する教材を準備する必要がある。

3.1.2. 開発リーダ向けの導入教育

開発リーダは、軽量型Webフレームワークを使って、システムの要件が実現できるかどうかを判断する責任を持つ。そのためすべての個々の機能について、それをどのように実現するかを、ソースコードレベルで理解する必要がある。また、一般の開発者にできるだけ実装の詳細を見せない方針をとる場合、軽量型Webフレームワークのラッパーを開発することになるかもしれない。つまり、開発リーダは軽量型Webフレームワークの詳細まで理解する必要がある。

クイックスタートガイドは、開発リーダが素早く軽量型Webフレームワークの概要を知るためには、非常に有効であり、実際に開発リーダの知識獲得に役立てられた。これに加え、当該プロジェクトでは、開発リーダに対して約一週間をかけ、DIやAOPの概念から、各種フレームワーク製品の設定ファイルの書き方に至る具体的な内容まで、マンツーマンでフレームワークの詳細を知識伝達した。

3.2. システム固有課題の洗い出しと解決策の選定

3.2.1. 実現可能性の検証

標準適用に当たり、システム実現のために解決すべき課題の洗い出しと、その解決策を見つけることを目的に、開発リーダと標準作成者（著者）でプロトタイピングを実施した。表５に、洗い出した課題の一部を示す。

表５　当該プロジェクトの課題（一部）

分類	課題	説明
開発者のスキル（本文の3.1.1, 3.1.2項と関連）	教育	開発リーダに、最低限必要となる軽量型WebFWの知識を伝えるにはどうするか。一般の開発者がスムースに開発に入れるようにするためには、何を準備し、どのように教育するか。
個別の設計方針（同3.2.1）	入力チェック	入力チェックをどう実現するか。JavaScriptをどの程度流用するか、一部をStruts Validatorに置き換えたほうがよいか。
	例外処理	例外の種類ごとに、それぞれどの階層でどのように処理するか。
	画面DTO	JSPとアクション間のデータ授受のため、画面の構造を持たせた画面DTOを作成するか。作成する場合、どのように構造化するか。
	画面/DB間のデータ形式変換	画面からDBまでの間で、どの階層でデータの形式（型、フォーマット、桁数など）を変換するか。
	ダウンロード機能	軽量型WebFWで、データのダウンロード機能をどう実現するか。
	JSPのタグ	Strutsタグを採用するか、それとも通常のHTMLタグを使うか。
開発の生産性（同3.2.2）	開発工数の見積もり	決められた開発者数（４名）で、どの程度の開発期間が必要か。
全体の設計・開発方針（同3.3.1）	クラス・メソッドの粒度	各種クラス（アクション・サービス・DTOなど）やメソッドをどのような単位で作るか（※DAOの粒度は標準に存在）。
	命名規則	各種クラスをどのような名前にするか（URLの定義も含む）。
	作成物の配置	アプリケーションのディレクトリ構成（Javaのパッケージ構成も含む）をどうするか。
開発プロセス（同3.3.2）	開発の分担	開発者にどのように作業を割り当てるか。サブシステム・機能ごとか、アーキテクチャの階層ごとか、あるいはそれ以外か。
	開発の順序	どのような順序で開発するか。プレゼンテーション層→サービス層→データアクセス層か、その逆か、あるいはそれ以外か。

この結果、比較的に基本的な設計・開発方針について不足があることが判明した。また、これら不足した設計・開発方針が、実プロジェクトへの具体的な適用を検討して初めて明らかになるもので、プロジェクト運営ノウハウそのものであることも判明した。

標準化のもたらす効果として、ノウハウ蓄積の促進が挙げられる。実プロジェクトに適用される設計・開発方針などのノウハウを一層活用するためには、鮮度良く信頼性高いノウハウを随時収集して標準に取り込み、それを共有するための仕組みを設けることが重要である。標準を背骨にしたノウハウ蓄積の仕組みを実現することにより、全社での知識の活用というより大きな目標に、一歩近づける可能性があると考える。

3.2.2. 開発工数の見積もり

軽量型Webフレームワークの標準として想定していなかったことの一つに、概算の開発工数見積もりの提供があった。

実プロジェクトでは当然の要求であるが、スケジュールを作成するため、開発工数の見積もりが必要になった。そこで、プロトタイピング時の実作業時間から、作成物ごとの平均的な作業時間を推定し、それに開発するシステムの規模を掛け合わせて、開発工数を算出した。その結果が表６である。

ここで特に注目したいのが、教育工数の見積もりである。開発工数の中でも教育工数は、往々にして少なく見積もられがちである。当該プロジェクトにおいても、教育工数が

表６　当該プロジェクトの工数見積もり

作業内容	時間（h）		時間計（h）
開発環境構築	2		2
軽量型Webフレームワークの理解 Strutsの理解 Springの理解（開発に必要な範囲） iBATISの理解	 6 2 2		10
作業内容	時間（h/本）	本数（本）	時間計（h）
web.xmlの作成	0.25	1	0.25
struts-config.xmlの作成	3	1	3
JSPの作成	2	40	80
JavaScriptの修正	2	80	160
アクションクラスの作成	2	80	160
アクションフォームクラスの作成	2	40	80
DTOクラスの設計・作成	4	20（ビューを含む）	80
サービスインタフェースの設計・作成	8	5	40
サービスクラスの設計・作成	1	5	5
DAOクラスの設計・作成	0.25	20	5
applicationContext.xmlの作成	10	1	10
sqlMapConfig.xmlの作成	0.25	1	0.25
SQL文の解析	3	80	240
iBATISマッピングファイルの作成	2	20	40

（注意）プロジェクトの実際の見積もり値に一部改変を加えている

１人当たり10時間で見積もられたのに対して、実際には、開発者が軽量型Webフレームワークの資料と、プロトタイプのソースコードを参照して理解を深めることに、約１週間が費やされた。

複数のフレームワーク製品を組み合わせている軽量型Webフレームワークの場合、理解しなければならない要素が多く、特に多くの教育時間が必要になる。Javaは理解しているが、フレームワーク製品の知識がまったくない場合、開発者として開発に参加できるようになるには、１週間前後必要である。また、開発リーダの作業に必要な知識を身に付けるには、当然経験や素養による個人差は大きいが、最低でも１～２か月を要するだろうと推測する。ただし、標準化により利用技術を選択・集中すれば、人材の流動化や教育投資の集中が可能になり、初期教育コストを削減することができるようになる。

前項で述べた設計・開発指針と同じように、開発工数についても、収集・共有の仕組みを実現することが重要である。より精度の高い開発工数見積もりが可能となることで、より積極的に標準が活用されるようになることを期待している。

3.3. 解決策の標準へのフィードバック

3.3.1. 作成物の標準化

開発の開始に先立ち、事前に作成物の粒度（作成単位）と、作成物の命名規則・配置（ディレクトリ構成）について決定しておく必要がある。

前掲の表５にも課題として挙げているが、軽量型Webフレームワークでは、アクションクラス・アクションフォームクラス・サービスクラス・DAOクラス・DTOクラスなどを作成することになるため、それらの作成物の粒度を決める必要がある。

当該プロジェクトでは、次のように方針を定めた。

- アクションクラス：画面のボタンごとに１個作成
- アクションフォームクラス：画面ごとに１個作成
- サービスクラス：機能（サブシステム）ごとに１個作成
- DAOクラス：RDBのテーブルごとに１個作成
- DTOクラス：RDBのテーブルごとに１個作成

作成物の粒度については、プロジェクトごとに一から考えるのではなく、標準として提供されていることが望ましい。今後軽量型Webフレームワーク標準に取り込んでいく必要がある。

また、命名規則と配置については、以下の対象が標準化された。

- 機能分類
- 機能
- イベントID
- 画面ID
- テーブルID
- JSPファイル名
- アクションフォームのビーン名
- 遷移先画面のパス名（URL）
- JavaScriptファイル名
- DTOクラス名
- iBATISのマッピングファイルにおけるDTOクラスのエイリアス
- iBATISのマッピングファイルにおけるResultMapのID
- iBATISのマッピングファイルにおけるSQLマッピングのID
- ファイルのディレクトリ構成
- Javaクラスのパッケージ構成

これら命名規則・配置の対象は、作成物の粒度とは異なり、プロジェクトごとに決定すべき性格のものが多い。標準としては、その決め方や例を提供することで、プロジェクトをガイドする立場をとるべきである。

3.3.2. 開発プロセスの具体化

開発に着手する前に、開発の分担と順序を決定しておく必要がある。

まず分担については、図２のように、大きく分けて二とおりの方法が考えられる。

① アプリケーションの機能・サブシステムごとに開発者を割り当てる方法
　利点：一度分担を決めれば、互いに干渉することなく開発を進めやすい。開発者ごとの責任範囲も明確になる。
　欠点：開発者全員が、すべてのフレームワーク製品を理解する必要がある

② アーキテクチャの階層ごとに開発者を割り当てる方法
　利点：一部のフレームワーク製品のみを理解する開発

者を活用することができる
欠点：あらかじめ階層間のインタフェースを明確に決定しなければならない

当該プロジェクトでは、開発者ごとの責任範囲を明確にすることを重視したため、①の方法を採用した。

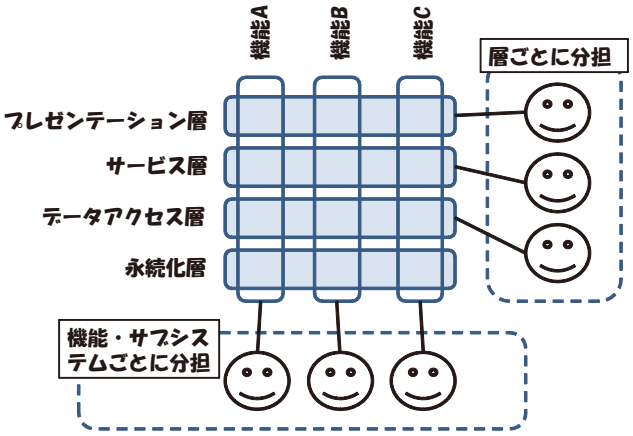

図２　開発の分担方法

次に開発の順序だが、これも大きく分けて二とおりの方法が考えられる。なお以下、プレゼンテーション層→サービス層→データアクセス層→永続化層の順を「上位層→下位層」と表現する。

①　上位層→下位層の順で作成する方法
利点：ユースケースなどの仕様書をもとに、明確なイメージを持って開発を進めやすい
欠点：下位層のインタフェースを想定しなければならない

②　下位層→上位層の順で作成する方法
利点：下位層で共通化を図りやすい
欠点：上位層を作成するタイミングで、下位層への手戻りが発生しやすい

当該プロジェクトでは、手戻りの発生による開発遅延のリスクを大きく見て、①の方法を採用した。

以上のような、開発の分担と順序は、前項の作成物の標準化と同様、プロジェクトごとに一から検討するよりは、標準として提供されることが望ましい。当社で進めているほかの標準化作業によって、基本方針は提供される予定だが、軽量型Webフレームワークとしては、今後さらに、各種クラス（アクション・アクションフォーム・サービス・DAO・DTOなど）や画面（JSP・HTMLなど）、フレームワーク製品の設定ファイルなどの詳細なレベルで、開発プロセスを提供する必要がある。

4. まとめ

本稿では、軽量型Webフレームワークを題材に、実用的な標準のあり方について実際の適用事例に基づき考察した。事例から得られた主な知見は次のとおりである。

- 利用者の区分、前提とするスキルレベル/目標とするスキルレベルを明らかにし、利用者ごとに適切な説明範囲と難易度を設定した教材を整備すること。これと同時に、前提となるスキルレベルを、利用者に明確に提示することもまた重要である。
- 設計・開発方針を漏らさず提示すること。また、プロジェクトで見出された有用な設計・開発方針を標準に取り込み、標準をブラッシュアップしていくこと。
- 標準を利用した場合の開発生産性を提示すること。また、プロジェクトでの生産性を収集し、見積もりの精度を高めていくこと。
- 個々のプロジェクトで標準化すべき対象については、標準化のための考え方や例を提示すること。
- 標準の一部としてプロセスを提示すること。なお当社ではMDAD[6]による標準化を進めており、これと連携することでより包括的なものとなる。

このように標準を定着させるための課題と対策の一部が明らかになった。ここで得た知見を一言でまとめれば、「標準を定着させるためには、標準を利用するための前提条件を明らかにした上で、利用者が求める標準になるよう継続的に改善を図っていくことが重要である」ということである。

最後に、得られた知見をもとに、標準の利用者と作成者、および会社が対処すべき課題を提起する。

- 利用者の課題：プロジェクトではできるだけ既存の標準を利用し、新たに独自の標準を作らない。
- 作成者の課題：プロジェクトの有用な資産を抽出し、標準にフィードバックするPDCAサイクルを回す。これにより、プロジェクト個々の資産が会社全体の資産として生まれ変わり、他のプロジェクトにも利益をもたらすようになる。
- 会社の課題：標準利用者は、標準を利用するための、最低限のスキルを保持しなければならない。そのための教育コースを整備し、また、自己研鑽を喚起する。

以上のように、標準を浸透させるには、関係者それぞれに相応の責任が課せられるのではないだろうかと考える。標準化活動をより意義あるものにするためには、利用者も積極的に作成者に問題を指摘するなど、相互の努力と協力が必要である。本稿で提示した知見はその指針を与えるものと考える。立場のいかんによらず、標準化への関わり方について問い直すきっかけを提供できたとすれば幸いである。

参考文献

1) Struts Framework
http://struts.apache.org/
2) Spring Framework
http://www.springframework.org/
3) iBATIS Data Mapper Framework
http://ibatis.apache.org/
4) DI（Dependency Injection、依存性の注入）
http://kakutani.com/trans/fowler/injection.html
5) AOP（Aspect Oriented Programming、アスペクト指向プログラミング）
http://aosd.net/
6) 松山 圭一, “プロジェクトのUMLモデル化 －MDADのアプローチー“, exa review vol.8, P47-58, 2007.10
