3層 Webアプリケーションの実用実験による変更容易性の評価

津久井　浩[†]　中所武司[‡]
Hiroshi Tsukui　*Takeshi Chusho*

1　はじめに

近年、インターネットやイントラネットに接続されたパソコンの普及と共に、オフィス業務の効率化という観点から、業務の専門家が自ら情報システムを構築する必要性が高まっている [1]。基幹業務のような大規模システムでは、トップダウンに開発を行なっていくが、小さな部門あるいは個人を対象とした業務や頻繁に機能変更が伴うものには従来の方式はなじまない。そこで本実験では、最初にあらかじめ充分な要求分析を行なうような従来の開発方法ではなく、実際に使用しながら改善していくという開発方法を用いて、変更容易性の検証を行なった。システムの対象には、エンドユーザ主導の開発にふさわしい Web アプリケーションの例として、会議室予約システムを選択した。そして、我々の所属する学科内での実用実験を実施し、機能拡張や操作性向上のための実際の変更要求をもとに評価を行なった。

2　システム概要

従来は、事務室に貼り出された会議室のスケジュール表の予約時間に直接名前を書き込んでいく方式であった。これを Web ブラウザ上で実現することを考え、図 1 のように機能設計を行なった。

	一般利用者	運用管理者
ユーザ管理 サブシステム	ログイン パスワード変更	ログイン ユーザ登録・変更・削除
会議室管理 サブシステム	会議室予約・取消・照会 定期予約 個人予約状況照会	会議室代理予約・取消 会議室情報変更 休日設定

図 1: 機能分類

今回の実用実験では、使いながら機能拡張していくという開発方法をとるために、頻繁な変更に対応できる保守性が重要となってくる。そこで、本システムにおいては、Java クラスをモデル、サーブレットをコントローラ、JSP をビューとした MVC モデルに基づいた図 2 のような構成を適用した。

[†]明治大学大学院理工学研究科基礎理工学系専攻
ソフトウェア工学研究室

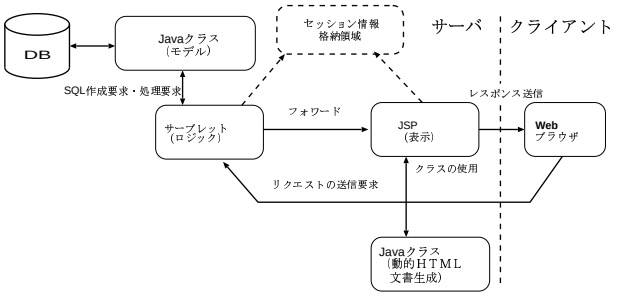

図 2: システム構成図

サーブレットは、ページのフロー制御・フォームの入力例外処理・Java クラス（モデル）への処理依頼の 3 つの役割を持つ。Java クラス（モデル）では、ビジネスロジックの実行と SQL 文の作成・発行を行なう。JSP では、HTML 文書の生成を行なう。Java クラス（動的 HTML 文書生成）では、表示情報の動的生成を伴う HTML 文書の作成を行なう。

なお、本システムのプログラムサイズは、サーブレットクラスが 53 クラスで約 6600 行、JSP クラスが 75 クラスで約 3700 行、Java クラス（モデル）が 12 クラスで約 1600 行、Java クラス（動的 HTML 文書生成）が 15 クラスで約 1800 行である。DB には Oracle を使用し、テーブルの数は、9 個である。

3　分析データ

本システムの運用開始後に寄せられた、システムに対するバグの報告および利用者からの要望等の 15 件に関して、変更内容の種別、対策レベル、変更・追加箇所の 3 つの視点から分析を行った。変更内容の種別とその件数の統計を表 1 に、対策レベルによる分類結果を表 2 に示す。対策レベルは、処理の作業量の大きさを変更作業にかかった時間で分類したもので、レベル 1 は 10 分未満、レベル 2 は 10 分〜1 時間、レベル 3 は 1〜2 時間となっている。

変更による影響範囲の分析方法として、15 件の各々に関して、図 2 に示した 6 つのサブシステムのうち、変更対象とならない Web ブラウザを除いた残りの 5 つのサブシス

表 1: 変更内容の種別とその件数

バグ	操作性向上	機能拡張	理解容易性向上
5	4	4	2

表 2: 対策レベルによる分類結果

レベル１	レベル２	レベル３	レベル４	レベル５
7	6	2	0	0

テムの何箇所を変更したかという観点で分類した結果を図 3 に示す。

	JSP	Javaクラス (動的HTML 文書生成)	サーブレット	Javaクラス (モデル)	DB	合計
1箇所	6	1	1	1	0	9
2箇所						5
3箇所						1
合計	8	5	5	4	0	

図 3: 影響範囲による分類結果

4 評価結果

4.1 入出力フォームの追加・変更

変更作業時間が最も長かった対策レベル３の２件は、共に操作性向上に該当した。１つは影響範囲が３ヶ所に及ぶもので、会議室選択画面上で、予約する週の選択のために３ヶ月分の月間カレンダーを表示させたいという要求であった。もう１つは、予約時の週間カレンダー表示時に同じ週の別の会議室の予約状況の表示を指示できるようにしたいという要求であった。この２件の共通点は、操作性向上のために新たにフォームの拡張を行なったということである。

このことより、ユーザインターフェイスについては、Web ページごとに事前に明確化するのが好ましいと考えられる。

4.2 データベーステーブルの変更

今回の変更内容においては、対策レベル４と５に分類されるものはなかったが、この分類に該当するものとして、変更箇所に DB が関係するものが考えられる。たとえば、本人による予約か運用管理者による代理予約かを色分けで表示できるように、DB の予約データテーブルの列を追加することを考える。このとき、Java クラス（モデル）、サーブレット、Java クラス（動的 HTML 文書生成）の変更も必要となる。このように DB に変更を及ぼすものは、その他のシステム構成要素にも影響を及ぼすことが多いため、必然的に変更作業も多くなる。

4.3 MVC アーキテクチャの評価

変更箇所を個別に見ると、サーブレットは、処理を呼び出すためにわずかに変更・拡張したものがほとんどであり、作業時間も少なかった。JSP の変更も HTML 記述部分がほとんどであったので、変更量としては少なかった。Java クラスについても、新規にクラスを追加することはなく、既存のクラスの簡単な拡張のみであった。各サブシステムごとの変更量は要求内容によって変わってくると考えられるが、各々の役割を明確に分担していたので、変更箇所を局所的に押さえることができ、変更時間の減少につながった。

4.4 ソフトウェアアーキテクチャの考察

今回適用したアーキテクチャでは、サーブレットクラスの中に、フォームの内容に対する例外処理と Java クラスへの処理要求操作とページフローの制御が混在していた。そのため、ページフローを変更するときは、該当するサーブレットクラスを見にいき、そのクラスの中のページフロー記述部分を探す必要がある。この改善策として、ページフローの制御は各サーブレットクラスの中に分散して記述するのではなく、ページフローを管理する専用のクラスを用意することで、ページフローの変更を局所化できる。また、サーブレットクラスごとに分散して記述された方式よりも、全体のアプリケーションの流れを理解することが容易になる。

5 おわりに

本報告では、最近ニーズが高まっている身近な業務の Web アプリケーション化に対応するために、作りながら変更していくという開発方法の実現可能性を検証した。

参考文献

[1] Takeshi Chusho, Katsuya Fujiwara, Hisashi Ishigure and Kei Shimada : A Form-based Approach for Web Services by Enduser-Initiative Application Development, SAINT2002 Workshop (Web Service Engineering), IEEE Computer Society, pp.196-203 (Feb. 2002).

[2] 原田洋子:Java サーバサイドプログラミング 技術評論社 2001 年出版

[3] 石榑久嗣，紺田直幸，中所武司：メッセージフローモデルに基づくエンドユーザ主導型アプリケーション構築・検証技法、情報処理学会 オブジェクト指向 '2000 シンポジウム論文集，pp.133-140（Sep. 2000）

[4] ジム・コナレン：UML による Web アプリケーション開発 ピアソン・エデュケーション (2000)