論　文

予約業務を例題としたWebアプリケーション用フレームワークの再利用性の評価

中所　武司[†a)]　　津久井　浩[†*]

Reusability of Web Application Framework for Reservation Systems

Takeshi CHUSHO[†a)] and Hiroshi TSUKUI[†*]

あらまし　インターネットやイントラネットの普及に伴い，現在，様々な業務がWebアプリケーション化されている．そこで，Webアプリケーション開発技法の研究の一環として，問題領域に特化したフレームワークを取り上げ，その再利用性について分析した．すなわち，問題領域として予約業務を選択し，Webアプリケーションの具体例として会議室予約システムと座席予約システムを開発し，その共通部分を分析することで，フレームワークの抽出を行った．具体的には，三層アーキテクチャを前提として，ユーザインタフェース部分についてはオープンソースフレームワークStrutsを適用し，業務処理部分については業務ロジックの共通部分を抽出し，データベース処理部分についてはテーブル設計に関するビジュアルツールを用いたテーブル処理コードの自動生成を実現した．そして，本フレームワークを再利用性の観点で分析するために，フレームワーク抽出に用いた例題とは性質の異なる図書予約と商品予約を選択し，適用実験を行った結果，本フレームワークで定めたデータ処理方式に基づくことで，幅広い予約業務に対して高い再利用性を挙げられることを確認した．

キーワード　Webアプリケーション，フレームワーク，三層アーキテクチャ，再利用

1. まえがき

インターネットやイントラネットの普及に伴い，ユーザインタフェースにWebブラウザを利用するWebアプリケーションが増加している．しかしWebアプリケーションをゼロから開発するのは，開発コストが増大するだけでなく，開発方法の習得時間もかかるため，短期開発の要求にこたえられない．

そこで，最近の情報システム構築では，オープンなアーキテクチャとフレームワーク[1]，デザインパターン[2]，コンポーネント[3]などの構成要素からビジュアルツールを用いてアプリケーションを再帰的に構築していくことが追求されている．特に問題領域に特化したフレームワークは，その領域のアプリケーション構築を行うときの開発効率の向上を図れることが確かめられている[4]．

本研究[5]では，フレームワークを利用したWebアプリケーション開発に関して，再利用性の観点から分析した．類似の研究としては，あらかじめフレームワークを用意しておいて，プログラミング言語とフレームワークに関して初心者の院生にアプリケーションを開発させて，品質と生産性を評価する実験が報告[9]されている．その中で，再利用率の測定も行われ，77〜85%であったとのことである．対象分野としては，遠隔授業，オークション，テレビ会議などのネットワークアプリケーションを選択し，CORBAを用いたフレームワークを使用しているが，アプリケーションの仕様の詳細やプログラム構造の詳細は述べられていない．

本論文では，対象問題領域として，ビジネス分野の一般的なアプリケーションで，かつ，適用範囲が広い予約業務を選択するとともに，具体的に開発した複数のアプリケーションから共通部分をフレームワークとして抽出し，それを新たなアプリケーション開発に利用するというアプローチを取った．再利用性については，詳細なプログラム構造を対象に分析し，その改善策を示す．アプリケーションの開発者についても，よ

† 明治大学理工学部情報科学科，川崎市
Department of Computer Science, Meiji University, Kawasaki-shi, 214–8571 Japan
* 現在，（株）日立製作所
a) E-mail: chusho@cs.meiji.ac.jp

 © (社)電子情報通信学会 2005

り実世界に近い条件として，プログラミング言語とフレームワークに関して，ある程度のスキルを有する者が担当した．

なお，オブジェクト指向技術に基づくフレームワークの研究は幅広く行われており，その定義も多様である．フレームワークは，クラス階層を構成するクラスライブラリとそれらのクラスから生成されるインスタンスの間の相互作用を規定したモデルから構成されるという構造に着目した定義，あるいは，カスタマイズによって特定のアプリケーションを簡単に作成するためのもとになる骨組みないし準完成アプリケーションであるという利用形態に着目した定義などが一般的である [1], [6], [7]．実際にフレームワークを利用してアプリケーションを構築するときのカスタマイズ方法についても種々の方法があるが，本研究では，フレームワークの概念を再利用性の観点からとらえ，アプリケーション構築時に再利用可能な部分をフレームワークとする．詳細はその該当部分で述べる．

本論文では，最初に三層 Web アプリケーション及びフレームワーク構築の基本方針について述べる．次に，対象問題領域とした予約業務の定義と分類及び抽出した予約業務フレームワークについて述べる．更に，テーブル設計に関するビジュアルツールを用いたテーブル処理コードの自動生成について述べる．最後に構築したフレームワークの例題業務への適用実験と評価の結果をまとめる．

2. 三層 Web アプリケーション

2.1 三層アーキテクチャ

これまで分散システムとして広く使用されてきたクライアント/サーバシステムは，クライアント側にアプリケーションをインストールする必要があり，アプリケーションに修正が必要なときには再インストールしなければならないなどのデメリットが大きい．一方，Web アプリケーションは，クライアント側はブラウザだけなので，このような問題は生じない利点がある．三層 Web アプリケーションは，図 1 のようにユーザインタフェースを提供するプレゼンテーション層，アプリケーションの処理を行うアプリケーション層，データを管理するデータソース層の三つの層で構成されており，それぞれが，Web ブラウザ，アプリケーションサーバ，データベースサーバに対応している．

本研究では，アプリケーションサーバに Tomcat，データベースサーバに Oracle を適用した．なお実装言語には，Java 言語と Java のサーバサイド技術であるサーブレット・JSP を適用している．

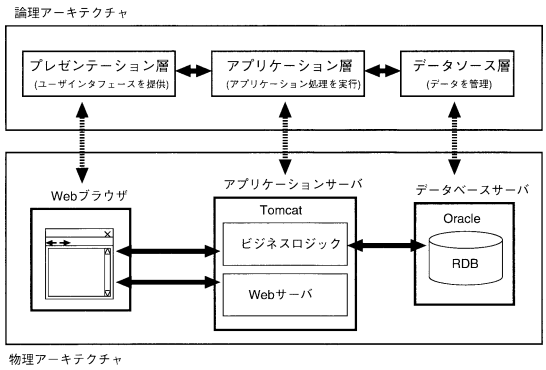

図 1　三層アーキテクチャ
Fig. 1　Three-tier architecture.

2.2 Struts の適用

サーブレット・JSP を中心とするサーバサイド Java テクノロジーは Web アプリケーションの実行速度や拡張性の面における課題を改善した．しかし，業務仕様の変更や機能拡張などの要求が頻繁に発生する状況に伴い，よりいっそうの開発期間の短縮や保守性の向上が課題となっている．このような背景からオープンソースの Web アプリケーション用フレームワークである Struts [8] が登場した．Struts は，特定の問題領域に特化せず，Web アプリケーションを開発するためのテンプレートを提供する．そこで，今回，三層のうちのユーザインタフェース関連処理には Struts を適用することにした．

Struts では，従来の Web アプリケーション開発では開発者自身が行う必要のあった HTTP リクエストの管理やフォームデータの格納や画面フローの制御などを機能として提供している．特に画面フローの制御方法は，プログラムとは別に外部に XML で定義するという方式をとっているので，フローの変更時にプログラムをトレースする必要がなく，頻繁に発生する画面関連の仕様変更に柔軟に対応できるという利点がある．更に，Struts は，Model-View-Controller (MVC) パターンに準拠して，アプリケーションの状態を保持するモデルと，そのモデルに対する表示形式を提供するビューと，クライアントからのリクエストを受け付けるコントローラの役割が明確に分離されており，保守性・拡張性が高い．

Struts を適用したシステムアーキテクチャを図 2 に示す．それぞれのサブシステムの役割について簡単

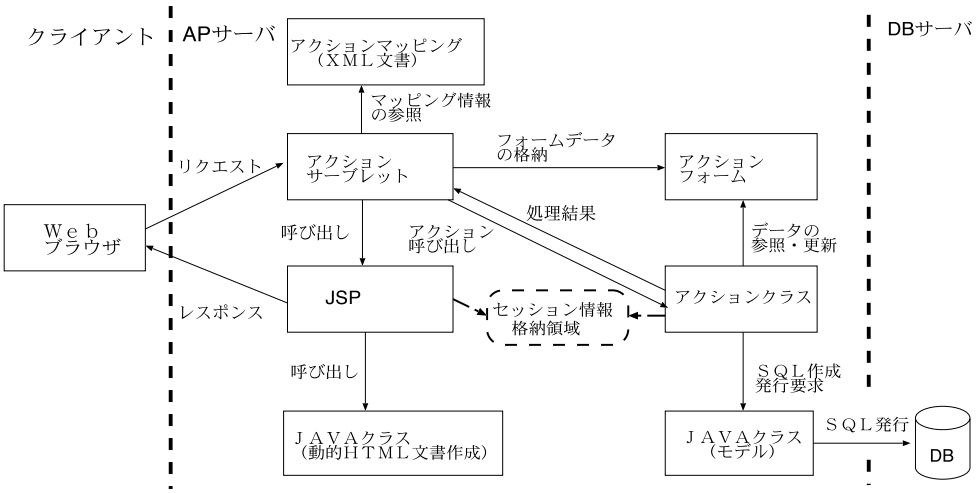

図 2　Struts を適用したシステムアーキテクチャ
Fig. 2　The system architecture based on Struts.

に説明しておく．アクションサーブレットは，アプリケーションの中に一つだけ存在し，すべての HTTP リクエストを管理する．送られてきたリクエストを最初に受け取り，その内容を解析し，その他のサブシステムへ指示を送る．アクションフォームは，Web ページ上でのフォーム情報を扱うクラスで，ユーザが入力したフォームの値を格納しておく．アクションマッピングは，URL とアクションのマッピング情報を保持する XML 文書ファイルである．アクションクラスは，ビジネスロジックを呼び出したり，記述したりするクラスである．Java クラス（モデル）は，アクションクラスからの処理依頼を受け，ビジネスロジックを実行するクラスである．SQL 文の作成や発行を行い，必要に応じて結果をアクションクラスに返す．Java クラス（動的 HTML 文書生成）は，表示情報の動的生成を伴う HTML 文書の作成を行うクラスである．

2.3　フレームワーク構築の基本方針

特定分野向けのフレームワークは，ソフトウェアアーキテクチャの基本的な枠組みと，その枠組みを構成する基本的なクラスライブラリを備えたものである．個々のアプリケーションは，そのフレームワークをカスタマイズして作成することになるので，短期開発と低コストと高品質を実現する．

カスタマイズのレベルとしては，コンポーネントの属性値の設定や既存のコンポーネント群から必要なものを選択するだけの簡単なものから，部品として用意されたクラスにソースコードを書き加えるものや，新たに追加すべきクラスを開発するものもある．カスタマイズのレベルによって開発者に要求されるスキルが異なるが，既にソフトウェアのアーキテクチャとそこで必要とされるクラス群におけるクラス間の静的構造が決定しているので，設計作業は大幅に削減できる．

この種のフレームワークの適用分野に関しては，例えば，本論文で対象としている予約業務などのように特定業務に特化したものと，業務は特定しないがユーザインタフェースの処理が主になるフロントエンド系やデータベース処理が中心のバックエンド系などのように，特定の情報処理内容に特化したものがある．前者の特定業務に特化したものは，既に業務の知識が業務コンポーネントとして用意されているので，カスタマイズは比較的容易である．一方，後者の特定の情報処理内容に特化したものは，アプリケーション固有の業務処理（業務ロジック）を新たに付加する必要があるが，業務に横断的に適用可能である．先に説明した Struts は，フロントエンド系の情報処理内容に特化したものの例である．

一般性のある三層アーキテクチャを前提として，ユーザインタフェース関連処理には Struts を適用することから，再利用性を重視したフレームワーク全体の構築方針は次のようになる．

すなわち，ユーザインタフェースについては，Struts を利用して画面遷移や入出力制御を共通化する．個々のアプリケーションに依存した画面の作成処理は再利用対象外とする．

業務処理については，分野を特定することにより分野共通部分の再利用率を高める．一般に分野を狭い範囲に限定すると再利用率が高くなると考えられるが，必然的に適用範囲が狭くなるため，再利用率と適用範囲はトレードオフの関係にある．今回の実験では，実際に我々がアプリケーションの利用者として実用化が可能な身近な例題として，会議室予約システムに注目した．そして，業務ロジックに共通性があると思われる範囲で，できるだけ適用範囲を広くするという観点で，予約業務に関して業務ロジックの共通部分を抽出することとした．

データベースについては，既存のものを利用するが，更に，テーブル設計を支援するビジュアルツールを開発して，テーブル関連処理部分をできるだけ自動生成する．

3. 予約業務フレームワークの抽出

3.1 フレームワークの対象問題領域の定義

フレームワークの対象問題領域として選択した予約業務は，従来のような書類や電話による施設予約やチケット予約手続きを Web ブラウザ上で実現することで，利用者と運営者の双方に手間や人件費の削減などの効果をもたらす身近な業務の一つである．本研究では，予約業務における予約の定義を「存在する資源を一定期間占有することをあらかじめ宣言すること」とする．また，本フレームワークが対象とする予約業務の基本機能は，予約，取消，照会の三つとする．

次に，予約業務における概念モデルを図 3 に示す．ユーザは，実際に予約，取消，照会の手続きを行う利用者である．資源は，予約対象を意味するが，ここでは概念的に時間と空間を組み合わせたものと見る．なお，空間にはものを含むものとする．そして，枠は，予約単位を意味し，時間分割したものや空間分割したものである．これらの概念について，具体的な例で説明する．もちろん予約には種々の変形例があり得るが，ここでは概念的な説明にとどめ，必要に応じて言及することとする．

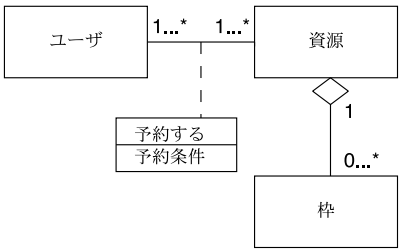

図 3　予約業務の概念モデル
Fig. 3　Conceptual model of reservation.

- 会議室予約

会議室予約に代表される施設予約は，個々の会議室やテニスコートなど，予約対象の最小単位の施設が資源となり，決められた単位時間が枠という形で区切られている．したがって，予約手順は，まず空間を特定して，次に時間を特定することになる．

- 座席予約

劇場などの座席予約では，イベントが開催される時間など，予約対象の最小単位の時間が資源となり，決められた単位空間（座席）が枠という形で区切られている．したがって，予約手順は，まず時間（イベント）を特定して，次に空間（座席）を特定することになる．乗り物の座席予約も同様に，まず時間（出発時間）を特定して，次に空間を特定することになる．

3.2 フレームワークの抽出方法

フレームワーク抽出のために，最初にアプリケーションの具体例を二つ選択し，それを実際に構築することで共通部分の分析及び抽象化を行うという方法をとった．このような方法でフレームワークを抽出する場合，具体的なアプリケーションの選択が重要である．例えば，比較的類似性の強い 2 種類を選択した場合，共通部分が増えるが，適用範囲が狭くなる．したがって，想定した問題領域の中の典型的なものでありながら，両者の類似性が弱いような 2 種類の具体例を選択する必要がある．

そこで，例題システムには，先に典型的な例として説明に用いた会議室予約システムと座席予約システムを選択した．会議室予約は予約単位が時間となるのに対し，座席予約は予約単位が空間となる．このように予約単位の異なる二つの業務のプログラム上での共通個所を抽象化することで，抽出するフレームワークの対象問題領域を広く設定できるように考えた．

これらのシステムに関して，予約，取消，照会，の三つの機能を実現した．その結果，全体の規模は，会議室予約システムが 2250 ステップ，座席予約システムが 1420 ステップとなった．なお，サブシステムごとの規模については後述する．

画面の実装レベルについては，会議室予約システムの予約機能の利用例で説明する．まず，ブラウザのコマンド選択画面から予約をクリックすると図 4 の会議室選択画面が表示される．そこで，プルダウンメ

図 4　会議室予約システムの会議室選択画面
Fig. 4　The screen for selection of the room to be reserved.

図 5　会議室予約システムの予約時間選択画面
Fig. 5　The screen for selection of the time unit to be reserved.

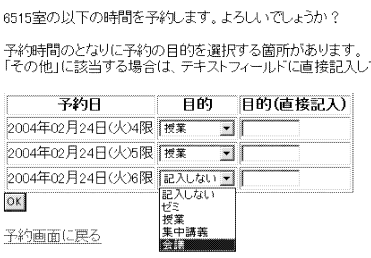

図 6　会議室予約システムの予約確認画面
Fig. 6　The screen for confirmation of reserved units.

ニューから予約する会議室を選択し，カレンダー（3か月分表示）から予約する週を選択すると，図 5 のように，選択した会議室と週の予約状況を表すカレンダーが表示される．次に，予約する時間をチェックボックスから選択し，予約ボタンを押下すると図 6 の予約確認画面が表示される．最後に，必要ならば予約目的を選択または直接入力し，OK ボタンを押下することで予約が完了する．

3.3　共通部分と業務固有部分の分類

会議室予約システムと座席予約システムの共通部分と業務固有部分の分析結果を図 7 に示す．点線の囲みは，図 2 のシステムアーキテクチャ上の各サブシステムに対応する．今回対象とした Web アプリケーションの処理内容に関しては，ユーザインタフェースと業務処理を比較的容易に分離できることから，採用した MVC のアーキテクチャと相性がよい．そのため，サブシステムごとのおおまかな役割分担があらかじめ決められており，2 種類の具体例のアプリケーションの間でシステム構成に大きな差異が生じることはないので，共通部分と業務固有部分の分析は比較的容易であった．

再利用の観点で各クラスを次の 4 種類に分類した．

- 共通クラス
- 一部共通クラス
- DB テーブル情報依存クラス
- 業務固有クラス

四つの分類のうち，共通クラスと一部共通クラスと業務固有クラスは，共通の程度の差による分類である．業務固有クラスは，クラス内で宣言されているメソッド名は共通であるが，その処理内容はアプリケーションに固有であったクラスである．

一方，DB テーブル情報依存クラスという分類は，基本的な処理は同じであるが，DB テーブルの構成に依存した部分のみ異なるというものである．個々のアプリケーションに依存してテーブルのデータ構造が異なるため，そのデータ構造を反映したデータ型クラス及びそのクラスとインタラクションのあるクラスのデータアクセス処理部分が異なることになる．したがって，テーブル設計の内容が決まれば自動的に決定するクラスである．すなわち，会議室予約における会議室と座席予約におけるイベントは図 3 における資源に対応し，会議室予約における予約時間と座席予約における座席は枠に対応していることから，予約，取消，照会の基本的な処理は共通している．ただし，資源と枠がもつ属性は異なり，各々の属性は DB テーブルの各列として表現される．そこで，属性は異なるがその属性に基づいて共通の処理をしているクラスについては，それらを DB テーブル情報依存クラスとして分類した．

なお一部共通クラスが JSP の中に存在する理由は，動的な HTML 文書と静的な HTML 文書を明確に分けることにより，共通部分と固有部分を分離したため

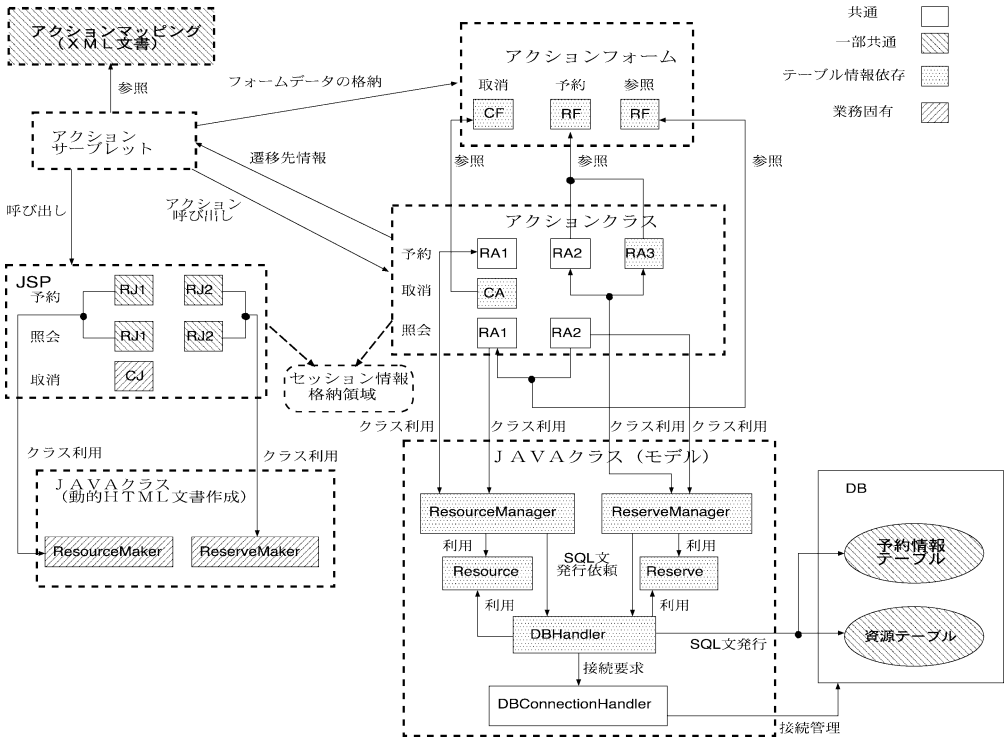

図 7　二つのシステムの共通部分と固有部分
Fig. 7　The shared part between two systems.

である．すなわち，動的な HTML 文書の作成については，Java クラス（動的 HTML 文書生成）に必要な情報を渡して依頼するという処理を共通部分として抽出したので，静的な HTML 文書を記述する個所のみが JSP クラスの共通でない部分ということになる．

3.4　各サブシステムの構成

次に各サブシステムの詳細な構成を説明する．まずはじめに，DB テーブルは，予約の対象となる資源を管理する資源テーブルと，予約のために必要な情報を管理する予約情報テーブルの二つから構成される．両システムのテーブルを図 8 に示す．資源テーブルには，会議室あるいはイベントに関する情報が格納される．そして，資源を特定するための列が主キーとして設定されている．予約情報テーブルは，誰がどの資源のどの枠を予約したかを管理するもので，会議室予約では予約者 ID・会議室名・予約時間が，座席予約では予約者名・イベント ID・座席が対応する．それ以外に存在する列は各業務に固有なものである．

Java クラス（モデル）は以下の構成である．

Reserve：予約情報テーブルのデータ型クラス

ReserveManager：上記の Reserve オブジェクトを

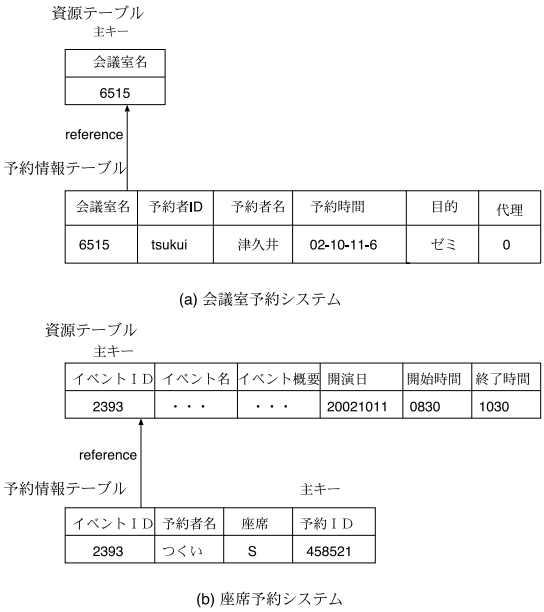

図 8　両システムにおける DB テーブルの構成
Fig. 8　DB tables for room reservation and seat reservation.

利用して予約処理のための SQL 文を作成するクラス
Resource：資源テーブルのデータ型クラス
ResourceManager：上記の Resource オブジェクトを利用して新しい資源の追加や削除のための SQL 文を作成するクラス
DBHandler：更新と検索の SQL 文を発行するクラス（検索結果は Reserve オブジェクトや Resource オブジェクトの配列として処理）
DBConnectionHandler：DB へのコネクションを管理するクラス

Java クラス（動的 HTML 文書生成）は以下の構成である．
ResourceMaker：予約対象の資源の一覧を表示し選択させる HTML 文書の作成クラス（会議室予約ではセレクトタグ，座席予約ではラジオボタンによる選択方式）
ReserveMaker：予約状況を表示し枠を選択させる HTML 文書の作成クラス（カレンダーや座席表の作成）

アクションクラスは以下の構成である．
RA1：ResourceManager からの資源情報取得クラス
RA2：ユーザが選択した資源の予約情報を ReserveManager からとってくるクラス
RA3：予約に必要な情報を集め ReserveManager に予約依頼するクラス
CA：取消に必要な情報をリクエストから受け取り ReserveManager に取消依頼するクラス

JSP は以下の構成である．
RJ1：予約資源選択用 Web ページの構築クラス
RJ2：対象資源の予約状況を表示し，枠を選択させる Web ページの構築クラス
CJ：予約取消に必要な情報を表示し，取り消す枠を選択させる Web ページの構築クラス

アクションフォームは以下の構成である．
RF：予約手続きの開始から完了までの間にユーザから受け付けた入力情報を保持するクラス
CF：取消手続きの開始から完了までの間にユーザから受け付けた入力情報を保持するクラス

アクションマッピングは，予約手続きのページフロー（資源の選択→枠の選択→予約処理）を定義している．

アクションサーブレットは Struts が提供するクラスで，MVC パターンにおけるコントローラの要の部分である．

3.5 システムの流れ

抽出したフレームワークを適用した場合の予約システムの処理の流れを示す．例として，ユーザによる予約したい資源の選択をきっかけに，選択された資源の現在の予約状況を表示する Web ページが作成されるまでの流れを図 9 に示す．具体的な手順は以下のようになる．

（1） アクションサーブレットによるユーザリクエストの受付
（2） 選択された資源を RF にセット
（3） RA2 にアクション実行命令
（4） RA2 が ReserveManager に選択された資源の予約情報取得の依頼
（5） 受け取った予約情報をセッション情報に格納
（6） アクションサーブレットはアクションマッピングを参照し，RJ2 にフォワード
（7） RJ2 はセッションにある予約情報を受け取り，ReserveMaker に送付
（8） ReserveMaker はその予約情報をもとに HTML 文書を作成し，RJ2 に送付
（9） RJ2 が HTML 文書をクライアントに送付

3.6 ステップ数によるフレームワークの割合

会議室予約システムと座席予約システムの各サブシステムのプログラムの中で抽出したフレームワークが占めるステップ数の割合を図 10 に示す．フレームワークのステップとして含めるのは，

（1） 共通クラス
（2） 一部共通クラスの共通記述個所
（3） テーブル情報依存クラス

である．業務固有クラスはフレームワークのステップに含めない．なお，フレームワークの占めるステップ

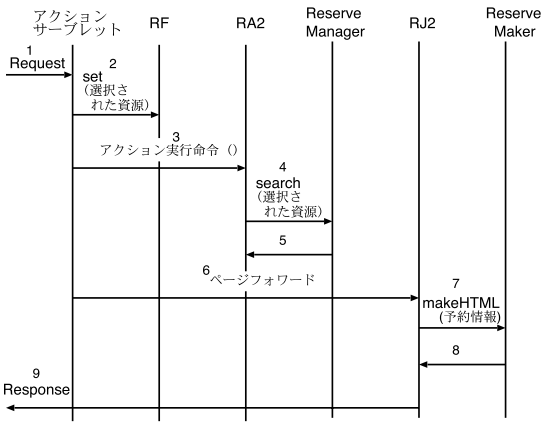

図 9 システムの実行の流れ（一部分）
Fig. 9 The flow of system execution.

	サブシステム	Javaクラス（モデル）	アクションクラス	アクションフォーム	JSP	Javaクラス（動的HTML文書生成）	アクションマッピング	合計
会議室予約	全体	740	370	170	320	550	100	2250
	フレームワーク	460	280	140	190	0	60	1130
	業務に固有	280	90	30	130	550	40	1120
	フレームワークが占める割合	62%	75%	82%	63%	0%	60%	50%
座席予約	全体	550	300	90	260	130	90	1420
	フレームワーク	480	270	90	190	0	60	1090
	業務に固有	70	30	0	70	130	30	330
	フレームワークが占める割合	87%	90%	100%	73%	0%	67%	77%

図 10　プログラムサイズ（ステップ数）におけるフレームワークの割合
Fig. 10　The rate of the framework in the number of steps.

数が二つのシステムで異なるのは，テーブル情報依存クラスのステップ数がテーブルの列の数などによって変化するためである．

フレームワークの占める割合の合計が，座席予約の 77%に対し，会議室予約が 50%と低くなった原因は，以下のような業務に固有の処理が発生したためである．

（1） Java クラス（動的 HTML 文書生成）の ReserveMaker でのカレンダー作成のステップ数が多くなった．特に，休日の計算や月や週単位の表示期間変更の計算処理が大きかった．

（2） Java クラス（モデル）においても，休日管理や時間の計算処理があった．

（3） アクションクラスにおいて，予約手続きの操作性向上のための処理や，予約確認画面の挿入に伴う処理が増加した．

すなわち，会議室予約では時間という枠の概念に伴う休日管理や時間の計算処理が発生したことと，予約確認画面の挿入の影響がそのページの処理を行うアクションクラスにも及んだことが再利用率の低下を招いたといえる．

一方，座席予約では，Java クラス（モデル）とアクションクラスとアクションフォームの部分は平均して 90%近い部分がフレームワークによって構築されている．これは，フレームワークに沿ったページフローであったこととビジネスロジックが比較的単純であったことに起因する．例えば，図 8 に示す DB テーブルの構成からも分かるように，座席予約ではイベントを選択した時点で開演日を決定済みと想定しているので，その後の枠の選択が容易であり，会議室予約におけるような複雑な時間関連の管理は不要であった．

4. ソースコード自動生成ツール

4.1 概　　要

抽出したフレームワークには，テーブル情報依存クラスが存在するが，これらのクラスの内容は，抽出した資源テーブルと予約情報テーブルの設計によって決定できる．また，抽出したフレームワークの Java クラス（動的 HTML 文書生成）内の 2 クラスはともに業務固有クラスであるが，空き状況を示すカレンダーや座席表などの動的な HTML 文書は通常予約業務において必要となるものである．

そこで，テーブル設計及びテーブル情報依存クラスと Java クラス（動的 HTML 文書生成）に必要な情報を定義することにより，ソースコードを自動生成するツールを開発した．その結果，テーブル作成用の SQL 文や，動的情報の表示用のコードをゼロから開発する必要がなくなった．

4.2 定 義 手 順

本ツールでは，次の項目を順に定義する．

1. 資源テーブル
2. 予約情報テーブル
3. Web 上での表示方法

資源テーブルと予約情報テーブルの定義は，通常の DB テーブル設計と同じように，各列の定義と一意な列の指定からなる．ただし予約情報テーブルは，資源テーブルの第 1 列に外部キーを設定する．また，定義した各列の値をユーザがいつ入力するかを定義する．この入力方法は，{予約時，予約時以外，不要} の 3 種類に分けられる．

予約時以外に入力する情報とは，ログイン時のように，予約を行う前に別の機能で既にユーザから入力されている情報をいう．不要とは，システム内部の処理によって決定される情報をいう．例として，図 8 に示した座席予約における予約情報テーブルの定義を図 11 に示す．一意な列と外部キーとなる列を○で示している．

手順 3 の Web 上の表示方法の定義では，Java クラス（動的 HTML 文書生成）で作成する HTML 文書を，ツール側が定義した表示形式の分類から選択し，それに応じた必要な情報をツールに入力していく．例えば，予約する枠の表示形式をリストにするかカレンダーにするか，選択方法をチェックボックスにするかラジオボタンにするかなどである．

ツールは，これらの定義からアプリケーションの

予約情報	イベントID	予約者名	座席	予約ID
一意				○
外部キー	○			
入力方法	予約時	ログイン時	予約時	不要

図 11　座席予約の予約情報の定義
Fig. 11　The definition of the information for seat reservation.

会議室予約		Resource Maker	Reserve Maker	合計	システム全体
ツール未使用	全体	70	480	550	2250
	フレームワーク	0	0	0	1130
	業務に固有	70	480	550	1120
	フレームワークが占める割合	0%	0%	0%	50%
ツール使用	全体	70	480	550	2250
	フレームワーク	40	200	240	1370
	業務に固有	30	280	310	880
	フレームワークが占める割合	57%	42%	44%	61%

図 12　ツールの使用による再利用率の変化
Fig. 12　Improvement in the reuse rate by using a tool.

ソースコードを自動生成する．通常の Java コードの記述と同様に，フレームワークを構成する各クラスの中にソースコードを出力していく．

なお，ソースコード自動生成ツールは，Java で開発し，32 クラス，2900 ステップである．

4.3　ツールによる再利用性の向上

本ツールをフレームワークの抽出に利用した会議室予約システムに適用した結果，Java クラス（動的 HTML 文書生成）を構成する 2 クラスの自動生成できる割合は図 12 のようになった．Java クラス（動的 HTML 文書生成）内で 44%が自動生成され，システム全体としては，フレームワークの割合が 11%上昇することを確認した．

5.　適用実験と評価

5.1　適 用 例 題

抽出したフレームワークの適用実験の例題として，図書予約システムと商品予約システムの二つを選択した．図書予約は，本の貸出を時間によって管理するので，会議室予約に似ている．しかし，会議室予約のように予約した期間を必ず占有するわけではなく，貸出期限前の返却を通常の処理として扱う必要がある．すなわち，図書を返却した時点で次の貸出を可能とし，次の予約者がいれば貸し出すことになる．このように

	サブシステム	Javaクラス（モデル）	アクションクラス	アクションフォーム	JSP	Javaクラス（動的HTML文書生成）	アクションマッピング	合計
図書予約	全体	410	290	120	270	180	70	1340
	フレームワーク	310	260	120	190	100	60	1040
	業務に固有	100	30	0	80	80	10	300
	フレームワークが占める割合	76%	90%	100%	70%	56%	86%	78%
商品予約	全体	510	310	160	300	200	70	1550
	フレームワーク	350	280	160	190	60	60	1100
	業務に固有	160	30	0	110	140	10	450
	フレームワークが占める割合	69%	90%	100%	63%	30%	86%	71%

図 13　適用例題におけるフレームワークの割合
Fig. 13　The rate of the framework in applied systems.

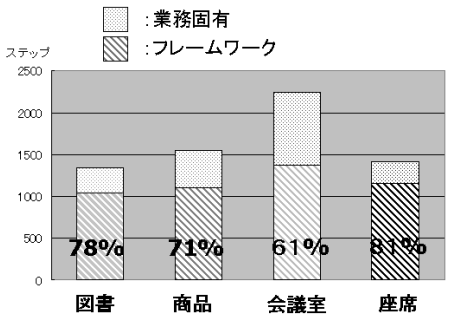

図 14　四つの予約業務における全体の再利用率
Fig. 14　The reuse rate in four reservation systems.

予約の中で特殊な位置付けとなる業務を本フレームワークに適用することで，どの程度の再利用率が得られるかを確認する．商品予約は，購入後は半永久的に所有するので，時間管理の概念が不要である．時間管理と空間管理の予約業務から抽出したフレームワークがこのような業務にも適用可能かどうかを確認する．なお，どちらの例題システムも単純なビジネスロジックのみを実装する．

5.2　実 験 結 果

フレームワーク部分と業務に固有部分をサブシステムごとにステップ数によって比較した結果を図 13 に，全体の再利用率を図 14 に示す．図 14 には，フレームワークの抽出に用いた会議室予約と座席予約の全体の再利用率として，図書予約と商品予約と同条件で比較を行うために自動生成ツールを使用した結果を示している．

業務固有部分の開発にはそれぞれ 1 日程度を要したが，各サブシステムごとの作業時間は，作成したステップ数にほぼ比例している．

図書予約システムの開発では，新規に追加したクラスは，Java クラス（モデル）で 1 クラスと JSP に 2 ページ（予約完了ページと取消完了ページ）のみで

あったので，合計の再利用率は 78%と高い値を示した．資源を時間で管理する業務の中では特殊な業務であると思われたが，実際にはフレームワークの用意したコードを全く修正することがなくアプリケーションを構築することができた．

商品予約システムの開発においても，図書予約と同様に再利用率 71%という高い数値を得ることができた．新規に追加したクラスも図書予約と同様で，Java クラス（モデル）に 1 クラス，JSP に 2 ページである．この商品予約は，フレームワークの抽出に用いた時間管理の会議室予約や空間管理の座席予約とは分類上は異なるものであったが，DB テーブルの構成を同様の形で設計すれば，本フレームワークに沿った形でアプリケーションを構築できることを確認した．

6. むすび

本論文では，ニーズの高まっている Web アプリケーションにフレームワークを適用することによって開発負担を軽減する方式を提案した．構築したフレームワークの再利用性検証のため，例題システムへの適用を行った結果，フレームワークの指定した DB 設計に基づけば，幅広い予約業務に対して高い生産性を挙げられることを確認した．

文　献

[1] M.E. Fayad, D.C. Schmidt, and R.E. Johnson, Building Application Frameworks, John Wiley & Sons, 1999.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns, Addison Wesley, 1995.

[3] A.W. Brown, Component-based software engineering, IEEE CS Press, 1996.

[4] 藤原克哉，中所武司，“窓口業務を例題としたエンドユーザ向き分散アプリケーションフレームワーク wwHww の開発と適用評価，” 情処学論，vol.41, no.4, pp.1202–1211, April 2000.

[5] 津久井浩，Web アプリケーションにおける予約業務フレームワークの実現と評価，明治大学大学院理工学研究科基礎理工学専攻情報科学系修士論文，March 2003.

[6] T. Lewis, G. Andert, P. Calder, E. Gamma, W. Pree, L. Rosenstein, K. Schmucker, A. Weinand, and J.M. Vlissides, Object-Oriented Application Frameworks, Prentice Hall, 1995.

[7] R.E. Johnson, “Frameworks = (Components + Patterns),” Commun. ACM, vol.40, no.10, pp.39–42, Oct. 1997.

[8] Jakarta Project, The Struts Web Application framework, http://jakarta.apache.org/struts/

[9] M. Morisio, D. Romano, and I. Stamelos, “Quality, productivity, and learning in framework-based development: An exploratory case study,” IEEE Trans. Softw. Eng., vol.28, no.9, pp.876–888, Sept. 2002.

（平成 16 年 8 月 16 日受付，11 月 15 日再受付）

中所　武司　（正員）

1971 東大大学院修士課程了．同年（株）日立製作所入社．1993 から明治大学理工学部情報科学科教授，現在に至る．ソフトウェア工学の研究に従事．コンポーネントベースのアプリケーション開発方法論に関心をもつ．工博（東京大学）．1982 情報処理学会論文賞，1986 大河内記念技術賞受賞．著書「ソフトウェア工学」（朝倉書店），「ソフトウェア危機とプログラミングパラダイム」（啓学出版），「プログラミングツール」，「人工知能」（昭晃堂，共著），「Java による Web アプリケーション入門」（サイエンス社，共著）など．

津久井　浩

2003 明大大学院理工学研究科基礎理工学専攻情報科学系修士課程了．同年，（株）日立製作所入社，現在に至る．メッセージ指向ミドルウェア (Message Oriented MiddleWare: MOM) の開発に従事．フレームワーク技術に関心をもつ．