# ドメイン特化の Web アプリケーションフレームワークの再利用性の評価

周　鋒　　中所　武司

明治大学理工学研究科ソフトウェア工学研究室

**概要**

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．Web アプリケーションを効率的に開発するための基盤を提供するフレームワークを使うことは基本の技法となっている．今回，ドメイン特化のフレームワークを開発し，再利用性の観点での評価を行った．具体的には，予約業務に特化したドメインを選択し，例題としての会議室予約システムを開発し，共通の機能を分析することで，フレームワークの抽出を行った．そして，例題と性質の異なる書籍予約システムとチケット予約システムを用いて適用実験を行った結果，本フレームワークで定めた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．さらに，ドメインの範囲と再利用率のトレードオフ関係を明確にするため，予約業務の子領域の時間予約業務に対し，フレームワークの開発と再利用性評価を行い，ドメイン特化フレームワークで再利用性の向上可能な部分を確認できた．

# The Reusability Evaluation of a Domain-Specific Web Application Framework

ZHOU FENG　and　TAKESHI CHUSHO

Software Engineering Laboratory, Graduate School of Science and Technology, Meiji University

**Abstract**

Web application is used in various business fields on Internet and intranets. It is an efficient way to develop Web application on the base of a framework. In this paper, a domain specific framework for reservation is developed based on a mee- ting room reservation system. Then, the framework is applied to two other types of reservation systems, online book store and soccer ticket reservation system, and its reusability is evaluated. In addition, another framework for time-based reservation, is developed, and the trade-off relationship between the range of the domain and the reusability is confirmed.

## 1　はじめに

インターネットの普及とデジタルコンテンツの拡大により，掲示板，ショッピングサイトなど，Web アプリケーションの利用が一般化している．

しかし，各アプリケーションをゼロから開発するのは容易ではない．そこで，再利用技術を用いて Web アプリケーションを効率的に開発するために，これらの Web アプリケーションに共通するアーキテクチャやクラス間の連携方法などの仕組みに着目して，基盤となるフレームワークを開発する技法が重要になっている．

本稿では，我々の今までの研究の続き[1][2]として，Struts[3]，Hibernate[4]といった汎用的なフレームワークを用いてドメイン特化のフレームワークを開発し，再利用性の観点から分析した．

さらに，一般的には，問題ドメインを狭い範囲に限定すると再利用率が高くなり，広い範囲を想定すると再利用率が低くなるというトレードオフの関係があるので，今回，最初に開発したフレームワーク

よりもドメインの狭いもう一つのフレームワークの開発と評価を行い，再利用性を向上できる部分を確認した．

## 2　システムアーキテクチャ

### 2.1　3層アーキテクチャ

従来の3層アーキテクチャ[5]は，ユーザインター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層で構成されている．実装レベルでは，それぞれWebブラウザとＷｅｂサーバ，アプリケーションサーバ，データベースサーバに対応する．

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MySQL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．

### 2.2　Strutsの適用

JSP/Servletを中心とした開発では，HTMLとJavaコードの混在のため，互いの変更がもう一方に与える影響が大きいことや．画面フローが把握しづらいことなどにより，システム全体を理解しづらく，開発と保守の容易性を低下させてしまうという問題があった．

Webアプリケーションフレームワーク StrutsはMVCモデルに準拠しており，アプリケーションの状態を保持するモデルと，表示・出力を司るビュー，入力を受け取ってその内容に応じてビューとモデルを制御するコントローラの役割が明確に分離されている．

Strutsを適用した後のアーキテクチャは図1のようになる．それぞれの役割について簡単に説明しておく．アクション・サーブレット(Action Servlet)は，アプリケーションの中に１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(Action Form Bean)は，Web ページ上でのフォーム情報を扱うクラスで，Struts が提供するクラスを拡張することが前提となる．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(Action)は，ビジネスロジック(モデル)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ない，必要に応じて結果をアクション・クラスに返す．

### 2.3　レイヤアーキテクチャ

システム開発において，レイヤアーキテクチャは有効な設計方法である．一般的に3層レイヤと言われ

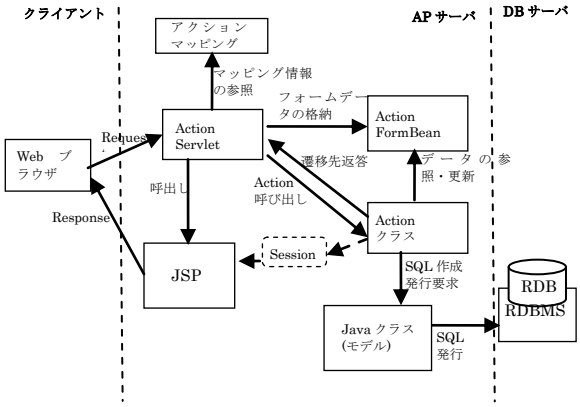


図 1.Struts を適用したシステムアーキテクチャ

れているのは，クライアントの画面表示や画面からの入力を受け付けるプレゼンテーションレイヤ，業務処理を行うビジネスロジックレイヤとデータベースなどのリソースにアクセスするデータアクセスレイヤからなる．

レイヤアーキテクチャの利点を大きく活かすための基本的な原則を述べておく．

・下位のレイヤは，上位のレイヤのためのインターフェースを提供する．

・上位のレイヤは，下位のレイヤの実装との関わりはない．

・上位のレイヤは，直下のレイヤのインターフェースしか使わない．

レイヤアーキテクチャの利点は以下に述べる．

・理解容易性

各レイヤは各自の業務のみを処理する．どんな業務がどこで処理されるのか明確なので，システム全体の理解が容易になる．

・開発効率の向上

各レイヤはほかのレイヤにインターフェースだけを提供するので，他のレイヤの実装を理解する必要がなくなり，開発効率が高くなる．

・保守性・拡張性

各部分の役割は明確に分担されるので，保守性・拡張性は高い．

### 2.4　DAOパターンの適用

DAOパターン[6]は永続的なデータにアクセスする部分とビジネスロジックを分割するためのJ2EEコアパターンの１つである．DAOパターンの適用により，各レイヤが疎結合になった．

そして，今回は，さらにGoFのFactory Methodパターン[7]を適用したDAOパターンを採用することで，データベースアクセスがビジネスロジックに対して完全に透過的になる．すなわち，データベースアクセス処理の実現において，JDBCにしても，O/Rマッピングにしても，ビジネスロジックには，共通のインターフェースでデータを処理する．アプ

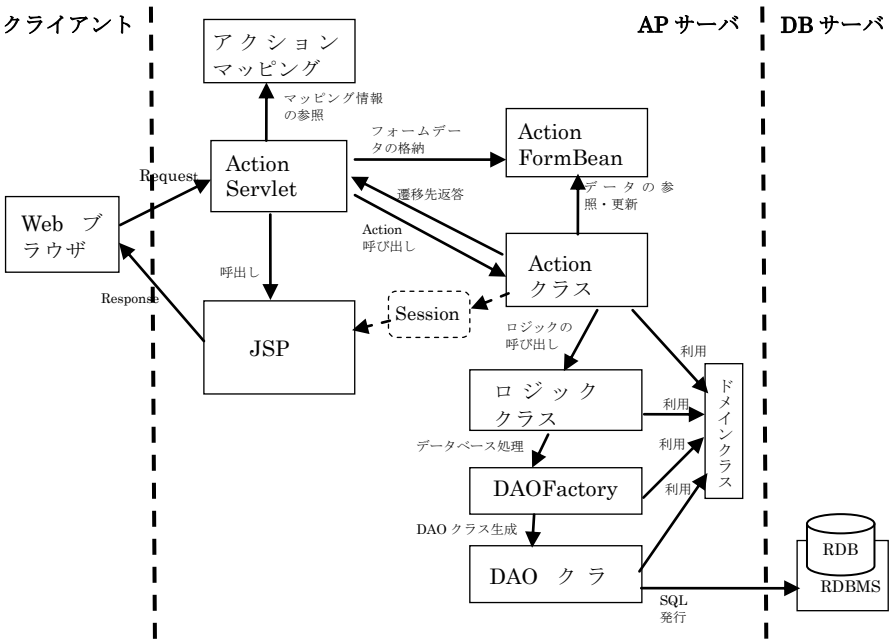


図 2. 本フレームワークのアーキテクチャ

リケーションを開発しやすくなり，可読性も向上できる．

具体的な実現方法として，インターフェースを定義するだけの DAO クラスを定義し，データアクセスを実現するクラスは，それらのインターフェースを実装する．そして各 DAO を生成するための DAOFactory クラスがあり，ロジック側は，この DAOFactory クラス経由で必要な DAO クラスを生成する．

### 2.5 本フレームワークのアーキテクチャ

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．

今回はドメインを特化するとともに，フレームワークのレベルでレイヤアーキテクチャを適用し，より高い再利用性と使い容易性を目指している．

今回開発するフレームワークのアーキテクチャは図 2 に示す．具体的には，プレゼンテーションレイヤは，Struts をベースにドメインフレームワークを開発する．ビジネスロジックレイヤは，共通化できる部分を提供する．データアクセスレイヤは，DAO パターンを適用する．

## 3 特化したドメインの選択と定義

### 3.1 ドメインの選択

ドメインとして選択したのは予約業務である．ここで予約業務の定義は「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．従来の書類や電話による施設予約やチケット予約手続きを Web 上で実現することである．

### 3.2 予約業務の体系

選択した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などの予約最少単位を組み合わせたものと見る．そして，枠は，予約単位を意味し，時間，空間などに該当する．

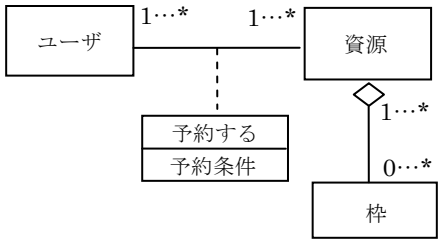


図 3.予約業務の概念図

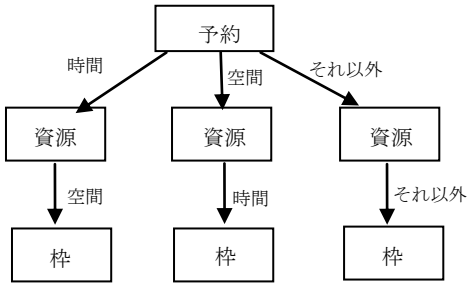


図 4.予約業務の体系

予約業務の全体的な体系について，予約手順で「時間→空間」，「空間→時間」，「それ以外」の3つに分け，図4に示す．

例で説明すると，「空間→時間」は，会議室予約のように，最初空間で予約の対象(どこの会議室)を特定し，その後時間の単位で予約を行う．「時間→空間」は，チケット予約のように時間(いつのイベント)を特定してから空間(どの席)単位で予約を行う．それ以外は，図書予約など上記両方とも該当しない予約業務である

今回は，「空間→時間」の会議室予約システムを例題としてフレームワークを開発し，「時間→空間」に属するサッカ試合チケット予約システムとそれ以外」の書籍販売システムで適用実験を行い，本フレームワークはすべての予約業務形式に対応できることが確認できた．

### 3.3 機能仕様

予約業務の機能は，システム管理者用と一般ユーザ用に分け，表1に示す．

表 1.予約業務の機能仕様

| | 一般ユーザ | システム管理者 |
|---|---|---|
| ユーザ管理 | ログイン | ログイン<br>ユーザー登録・変更・削除 |
| 資源管理 | — | 資源登録・変更・削除 |
| 予約機能 | 新規予約・予約照会・変更・削除 | 新規予約・予約照会・変更・削除 |

なお，今回開発したフレームワークでは，表 1 の機能のうち，予約業務の核となる一般ユーザの予約機能とログイン機能の2つの機能を対象とする．

## 4 フレームワークの開発

実際のフレームワークの開発は，3．までで述べたアーキテクチャと機能仕様に基づき，「例題開発」→「クラスとファイルの抽出」→「フレームワークの詳細化」という流れとなっている．

### 4.1 例題開発

例題としては，会社など利用される会議室予約システムを選択した．予約業務の体系の「空間→時間」予約に該当する．

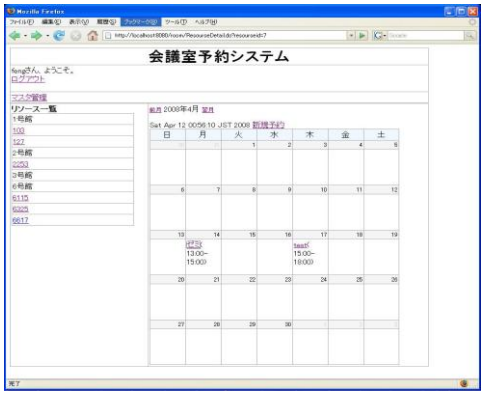


図 5.選択した会議室の当月の予約状況

構築した例題システムでは，最初に，ヘッダ，ログイン画面，会議室一覧，説明画面からなる初期画面が表示され，会議室を選択すると月間予約状況照会(図 5)でき，さらにログインした後に新規予約，既存の予約詳細の照会・変更・取消(図 6)ができるというようになっている．

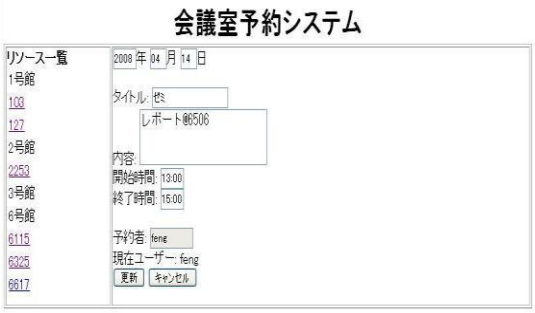


図 6.予約変更・取消画面

### 4.2 クラスとファイルの抽出

開発した例題に基づき，予約業務のフレームワークに必要となるクラスとファイルおよびデータベース設計の抽出を行った．それぞれ新規予約，変更などの機能ごとに行った．抽出した結果を図 7 に示す．

以下，主要な抽出部分の詳細について述べる．

データベース：予約対象とする資源を保存する Resource テーブル，資源分類の Grou テーブル，ログインのための member テーブルと roles テーブル，予約情報を管理する Reserve テーブルが共通となる．

Action クラス： ActionServlet から呼び出され，必要な初期化を行い，適切なロジッククラスを呼び出し，結果を受けてリンク先に遷移する．資源表示の ResourceShowAction，予約一覧表示ための ResourceDetailShowAction，予約・照会・変更・取消のための EntryReserveAction と EntryReserveShow-Action を抽出した．

Action フォーム：ユーザから送られた値をアクションへ転送する役割を担う．ResourceForm などのフォームクラスと Struts.xml 設定の中で定義したダイナミックフォームを抽出した．

ドメインクラス：Resource，Grou，Reserve，Member を抽出した．

Logic クラス：ビジネスロジックを処理するクラスである．AuthLogic，ResourceLogic，ReserveLogic を抽出した．

データベースアクセスクラス(DAO)：各実際の DAO クラスを生成するための DAOFactory が必要で，実際のデータベースアクセスクラスとしては，MemberDAO，ResourceDAO，ReserveDAO および開発者がそれらを継承した作った MemberDAOImp のようなクラスが必要である．

ビュー：表示する JSP ファイル，レイアウトを管理するテンプレートページおよびその設定ファイルを抽出した．

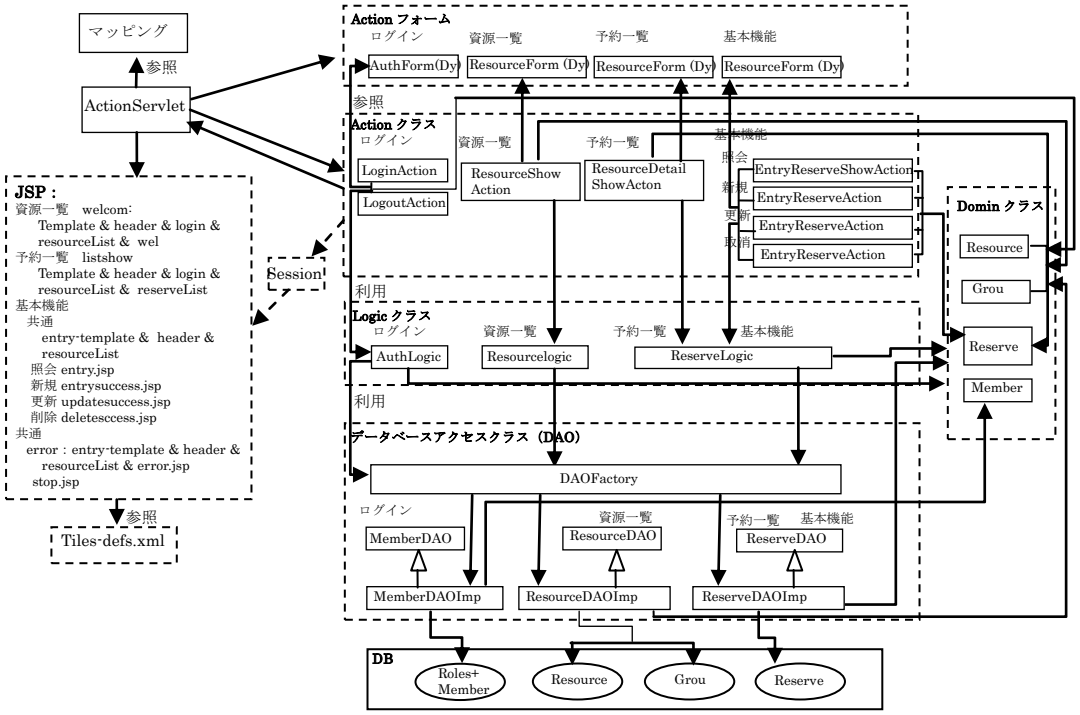


図 7.抽出したフレームワークのファイルとクラス

マッピングファイル：アクションマッピングを設定するもので，画面コントローラとして必要である．抽出したのは Struts.xml と Web.xml の二つの xml 設定ファイルである．

### 4.3 フレームワークの詳細化

前節でフレームワークに必要なファイルとクラスを抽出した．それらのファイルとクラスをフレームワークの中身として詳細化する．

具体的には，例題をベースに，予約業務のシステム開発において，共通で再利用可能な部分を，それらのファイルとクラスに追加し，親クラス，ライブラリなどとして再利用できるように整形して最終のフレームワークを構築する．

結果として，例題システムの 1549 ステップのうち，予約業務フレームワーク部分は 996 ステップとなった．再利用率は 64%といえる．

## 5 適用実験と評価

開発した予約業務フレームワークについて，予約業務体系において例題と違う形式の二つのシステムを用いて適用実験を行った．なお，どちらも単純なビジネスロジックのみを実装した．

### 5.1 適用実験 I

適用実験 I は，「時間→空間」予約のサッカ試合チケット予約システムを選択した．まず時間の概念で試合を選択し，空間の概念で席を予約するというようになる．この形式の予約業務システムでフレームワークを使う時，ドメインクラスの Resource と Group は，「空間→時間」の会議予約システムではそれぞれ会議室，ビルとするが，このシステムでは，それぞれ席と試合にする．

予約照会は，予約対象の状況照会から予約者の予約状況照会にすることで，開発した予約業務フレームワークを用いて，図 8 のような新しいサッカ試合チケット予約システムを構築した．

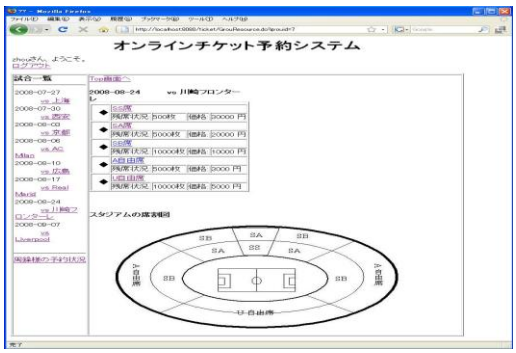


図 8. 適用実験 I で開発したシステム

構築したシステムにおけるフレームワークの割合は，表 2 に示す．本予約業務フレームワークは 62%の再利用率となった．即ち，「時間→空間」の予約業務形式において高い再利用率が確認できた．

表 2.適用実験Ⅰにおけるフレームワークの割合

| | Action クラス | Action Form | ドメインクラス | ロジッククラス | DAO クラス | ビュー | アクションマッピング | 合計 |
|---|---|---|---|---|---|---|---|---|
| 全体 | 290 | 94 | 182 | 139 | 419 | 397 | 70 | 1591 |
| フレームワーク | 245 | 47 | 84 | 90 | 294 | 175 | 61 | 996 |
| 業務固有 | 45 | 47 | 98 | 49 | 125 | 222 | 9 | 595 |
| フレームワークの割合 | 84% | 50% | 46% | 65% | 70% | 44% | 87% | 62.% |

### 5.2 適用実験Ⅱ

適用実験Ⅱは，「それ以外」予約の書籍販売システムを選択した．時間と空間の概念がなくなり，ある本を選択して予約することを行う．返却する必要がないので，永遠に占有することの宣言と考えればよい．

書籍購入の流れとしては，書籍分類により，それぞれの一覧から本を選んで，詳細画面に進む．購入画面に進む場合は，ログインしているかをチェックする．購入手続きを実行した後は購入結果画面を表示する．そのほか，購入履歴，変更，取消などの機能も実装した．

設計においては，ドメインクラスの Resource と Group は，それぞれ本と分類に当たる．予約照会は購入者の購入履歴にする．

構築した書籍販売システムにおけるフレームワークの割合は，表 3 に示す．

表 3.適用実験Ⅱにおけるフレームワークの割合

| | Action クラス | Action Form | ドメインクラス | ロジッククラス | DAO クラス | ビュー | アクションマッピング | 合計 |
|---|---|---|---|---|---|---|---|---|
| 全体 | 286 | 84 | 168 | 124 | 402 | 402 | 70 | 1536 |
| フレームワーク | 245 | 47 | 84 | 90 | 294 | 175 | 61 | 996 |
| 業務固有 | 41 | 37 | 84 | 34 | 108 | 227 | 9 | 540 |
| フレームワークの割合 | 85% | 56% | 50% | 72% | 73% | 43% | 87% | 65% |

表 3 により，65%の再利用率ができている．即ち，時間・空間以外の予約形式に対しても，かなり高い再利用性がある．

### 5.3 総合評価

適用実験ⅠとⅡ，およびフレームワーク開発用の例題の会議室予約システムでの再利用率を図 9 に示す．予約業務体系において，各形式のシステムを 1 つずつ開発して，それぞれ 64%，62%， 65%の再利用率となり，本予約業務フレームワークを使うことで，すべての予約業務に対して，開発効率を向上できると確認した．

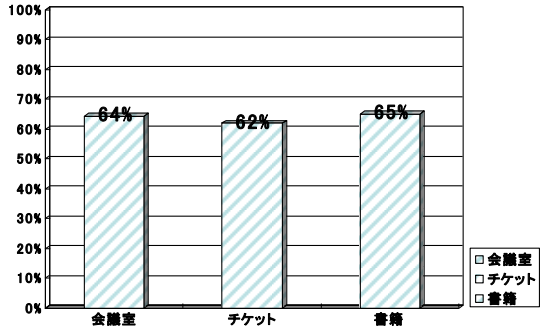


図 9. 3 つの予約業務における再利用率

コード量の削減のほか，以下の利点もある．

(1)設計の軽減

フレームワークのレベルでレイヤ化し，Struts と DAO パターンを適用することで，アプリケーション開発においてシステム設計の仕事を軽減することができる．

(2) 拡張性・保守性

より優れたアーキテクチャを提供しているため，各レイヤが独立でき，拡張性・保守性の向上することができる．

## 6 トレードオフ関係の考察

ドメイン特化の Web アプリケーションフレームワークには，一般的に，問題ドメインを狭い範囲に限定すると再利用率が高くなり，広い範囲を想定すると再利用率が低くなるというトレードオフの関係がある．

本章では，ここまで開発した予約業務フレームワークよりドメインの狭いもう一つのフレームワークを開発して適用実験を行い，ドメインの範囲と再利用性のトレードオフ関係を明確にする．すなわち，ドメインの変更による再利用性の変化を確認する．

### 6.1 フレームワークの開発

狭くなるドメインとして選択したのは，図 4 の予約業務体系の「空間→時間」予約業務となる．予約手順は空間を特定してから時間単位で予約するという形式である．予約業務の子ドメインとして，まずは，同じ例題の会議室予約システムを用いてフレームワークを開発する．

「空間→時間」予約業務は予約業務の子ドメインであるので，抽出したフレームワークのファイルとクラスは，図 7 とほぼ同じである．ドメインが狭くなった後にさらにフレームワークに追加した部分のみを述べる．

**DAO クラス：**

・予約時間が既存の予約と重複しているかどうか

のチェックためのデータベースアクセス処理；

**ロジッククラス：**

・予約時間が既存の予約と重複しているかどうかのチェックためのロジック処理；

**ビュー：**

・予約一覧表示の月間画面を生成するクラス

・画面レイアウトをコントロールするテンプレートファイルと設定ファイルに共通のレイアウトとして表示できる部分

・すでに予約が入っているなどの例外発生時の表示画面

**アクションマッピング：**

・例外のマッピング設定

抽出した「空間→時間」予約業務フレームワークのステップ数は 1227 となっている．予約業務より 231 ステップを増加した．

## 6.2 適用実験

開発したフレームワークは，「空間→時間」予約の教室予約システムで適用実験を行った．

教室予約システムは，学校で授業などのため，あいている教室を調べて事前に予約するシステムである．例題の会議室予約システムと同じく「空間→時間」に属するが，違うところは，予約の単位が何時何分から何時何分までと予め決めた授業の時限という単位になる．そのほか，予約対象の属性と予約の必要とするユーザから入力してもらう内容も違う．

適用実験で構築した教室予約システムにおける「空間→時間」予約業務フレームワークの割合は，表 4 に示す．

表 4.適用実験におけるフレームワークの割合

| | Action クラス | Action Form | ドメインクラス | ロジッククラス | DAO クラス | ビュー | アクションマッピング | 合計 |
|---|---|---|---|---|---|---|---|---|
| 全体 | 296 | 98 | 119 | 116 | 397 | 459 | 63 | 1548 |
| フレームワーク | 245 | 47 | 84 | 100 | 316 | 372 | 63 | 1227 |
| 業務固有 | 51 | 51 | 35 | 16 | 81 | 87 | 0 | 321 |
| フレームワークの割合 | 83% | 48% | 70% | 86% | 80% | 81% | 100% | 79% |

この適用実験で 1548 ステップの中，1227 ステップはフレームワークで提供した部分，業務固有として追加したのは 321 ステップのみとなる．79%の再利用率を達成できたことからわかるように，ドメインを狭くすることで，再利用性を大幅に向上できたといえる．

そして，より狭いドメインでは，処理する業務が明確になり，システムの処理の流れがほぼフレームワークで決定できるので，システム設計の作業も大幅に削減した．

## 6.3 異なるドメインにおける再利用性の比較

予約業務フレームワークと空間→時間予約業務フレームワークの両フレームワークを開発し，それぞれ適用実験を行った上で，再利用性の比較は図 10 に示す．

なお，予約業務フレームワークの再利用率は，適用実験のチケット予約システムと書籍販売システムの平均値とする．

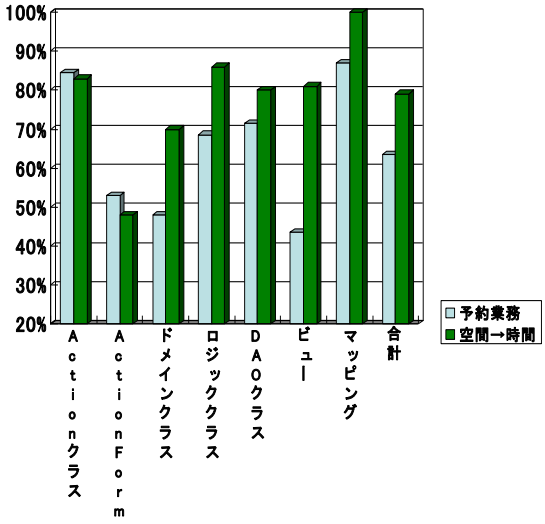


図 10.ドメイン範囲の異なるフレームワークの比較

図 10 により，フレームワークの各部分とドメインの範囲の関係を以下とまとめる．

(1) Action クラス
初期化とロジッククラスの呼び出し結果を受けてリンク先に遷移する役割で，ドメインの範囲と関係なく，抽出した両フレームワークとも高い再利用率ができている．

(2) Action フォーム
データ渡しの役割で，ドメイン範囲とは関係しない．

(3) ドメインクラス
ドメインが狭くなると，各アプリケーションのドメインクラスの類似性が高くなり，再利用率が高くなる．

(4) ロジッククラス
狭いドメインで再利用率が高くなる．ただし，ロジッククラスは，一番カスタマイズが多いところであり，ドメイン範囲と関係なく，ビジネスロジック処理の増加により再利用性が低くなる部分である．

(5) DAO クラス
狭いドメインでは，再利用性が向上する．

(6) ビュー
ドメインを広く取る場合，かなり低い再利用性しか実現できない．狭いドメインでは，共通の画面が多くなり，ライブラリ，テンプレートなどをフレームワークで提供することで，

再利用率を向上できる．

(7) アクションマッピング
両方とも高い再利用性を示すが，狭いドメインでは，今回の実験に限り，完全に再利用できた．

(8) 合計
予約業務ドメインに平均 63.5%の再利用率に対し，子ドメインの「空間→時間」予約業務では，79%となり，15.5%の向上率を達成できた．システム開発の際，最もドメインの狭いフレームワークが使えば，コード量を大幅に削減できることを確認できた．

よって，ドメイン範囲の変化に伴う再利用性への影響度は，「ビュー，ドメインクラス」→「DAO クラス，ロジッククラス，アクションマッピング」→「アクションクラス，アクションフォーム」の順になっている．ドメインフレームワークを開発する際，影響度大きい部分に注目し，さらに支援ツールなどを提供すると，より高い再利用性を挙げられると予想できる．

## 7　おわりに

本稿では，ニーズの高まっている Web アプリケーションにドメイン特化の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．開発したフレームワークの再利用性検証のため，性質の異なる二つの適用実験を行った結果，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．ドメイン範囲と再利用性の関連に関しては，子ドメインとしての狭い範囲のもう一つのフレームワークの開発と適用実験によって，トレードオフ関係及びその影響する部分を確認した．

## 参考文献


[1] 中所武司，津久井浩，
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，電子情報通信学会 和文論文誌 D-I 分冊，Vol.J88-D-I, No.5, pp.930-939,　May. 2005.

[2] 周鋒，中所武司，
問題領域を特化した Web アプリケーションフレームワーク構築方法の実験と評価，FIT2008, B-018, 147-149,　Sep. 2008.

[3] The Apache Software Foundation
http://struts.apache.org/

[4] Red Hat, Inc.
http://www.hibernate.org/

[5] 中所武司，藤原克哉，
Java による Web アプリケーション入門，サイエンス社，Feb. 2005.

[6] Deepak Alur, John Crupi and Dan Malks,
Core J2EE Patterns: Best Practices and Design Strategies，Prentice Hall / Sun Microsystems Press, June, 2003.

[7] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides,
Design Patterns: Elements of Reusable Object-Oriented Software , Addsion Wesley, 1995.