線形モジュール追加型抽象階層およびモデル駆動型アーキテクチャを用いた顧客管理システムの開発

松村, 智之 / MATSUMURA, Tomoyuki

(発行年 / Year)
2008-03-24
(学位授与年月日 / Date of Granted)
2008-03-24
(学位名 / Degree Name)
修士(理学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

修士論文

線形モジュール追加型抽象階層およびモデル駆動型アーキテクチャを用いた顧客管理システムの開発

2008 年 1 月 29 日 提出

指導教官　大森健児 教授

法政大学　情報科学研究科　情報科学専攻

学籍番号 06t0018　松村 智之

概要

Abstract

The conventional software development methodologies are subject to software defects causing serious problems in complex and large-scale software systems. Software defects arise from artificial mistakes in the making of programs. In this paper, the number of defects is reduced using the incrementally modular abstraction hierarchy (IMAH) and model driven architecture (MDA). The IMAH defines invariants at each level while climbing down from the abstraction level to concrete level in abstraction hierarchy. The IMAH can theoretically introduce UML class diagrams in its view level. Using the AndroMDA being a framework based on MDA, the majority of programs are automatically generated from UML class diagrams represented by the IMAH. In this paper, the development of a customer management system is described as an example of this method. The effectiveness of the IMAH for software development has been examined. The core system has been firstly developed and then enforced by adding other modules of a customer management system and a login system. As a result, large portion of programs have been automatically generated, and the number of system defects and software development time have been reduced.

あらまし

従来のソフトウェア開発手法では、ソフトウェアシステムの大規模化、複雑化に従ってシステムエラーなどが頻繁に発生するという深刻な問題を抱えていた。これは、プログラムの誤記といった人為的なミスが原因だった。本研究では、Model Driven Architecture(MDA)と、Incrementally Modular Abstraction Hierarchy(IMAH)を用いて、問題解決を目指す方法を提案する。IMAH は、抽象レベルから具体レベルへと階層を進みながら、各レベルで不変量を定義していく。これにより、UML 図を論理的に作成することが出来る。ここで得られた UML 図を基に、MDA 対応のフレームワークである AndroMDA を用いて、大部分のプログラムを自動生成する。本研究では、顧客管理システムの開発を通して、本研究で提案するソフトウェア開発法の有効性を調べる。開発は、顧客管理システム、ログインシステムとモジュールを追加しながら進めていく。研究結果より、多くのプログラムを自動生成することができた。それにより、エラーの削減、開発時間の短縮をすることができた。

目次

第１章　まえがき

1960 年ごろからソフトウェア工学の研究が進められている[1]。ソフトウェア開発手法として、ウォーターフォール・モデルがよく用いられた[4]。これは、要求仕様、分析、設計、実装、テスト、運用に工程を分け、順番に開発を進めていく手法である。各工程では、上流工程から引き渡された成果物を元に作業が行われる。また、原則的にこの順序を飛び越えたり、逆戻りしたりすることを認めないとしている。しかし、実際の開発では、実装時に初めてバグに気づく事はよくある。これは、大規模化、複雑化したシステムでは頻繁に起きてしまう。バグが見つかれば後戻りしてやり直さなければならないが、それには莫大な費用が掛かる。

その後、プロトタイピングモデル、スパイラルモデルなど考えられたが、最近では、オブジェクト指向を活用したソフトウェア開発手法が開発現場の主流になりつつある。その代表的手法が RUP(Rational Unified Process)である[3]。RUP は、ユースケースを開発の基点とし、ウォーターフォールのように始めから完全な形の完成品を想定するのではなく、重要な機能やリスクの大きな機能をユースケース単位に構築する手法である。しかし、曖昧さを含んだ UML 図ではバグを引き起こしてしまい、多くの時間をデバッグに費やさなければならなくなる。

本研究では、上記の問題を解決する方法として、システム開発において数学的要素を取り入れる。具体的には、Model Driven Architecture(MDA)と Incrementally Modular Abstraction Hierarchy(IMAH)を組み合わせたシステム開発手法を提案し、その検証を行った。

IMAH は、ホモトピーレベル、集合論レベル、位相空間レベル、接着空間レベル、セル空間レベル、表現レベル、可視レベルの 7 層に分かれている。抽象的なレベルから具体的なレベルへと階層を巡りながら、それぞれの層で不変量を定めていく。これにより、UML 図を理論的に描き、曖昧さを取り除くことができる。

MDA は、OMG(Object Management Group)によって定義されている。これは、モデルを中心として開発を進めていく方法である。UML などの標準モデリング技法を使ってアプリケーションの機能をモデル化し、さらにそのモデル情報を基にコードを自動生成する開発法である。本研究では、MDA 対応のフレームワークである AndroMDA を用いた。AndroMDA は、カートリッジと呼ばれる変換モジュールをプラグインすることで、さまざまなプラットフォームに対応する多様なプログラミング言語のソースコードを自動生成することができる[5]。現段階での AndroMDA では、ビジネスロジックについてのみ自ら記述する必要である。

実験結果より、大部分でバグの発生を防ぐことができた。本研究では、ビジネスロジックについてのみプログラムを書く必要があり、その部分においてのみテスト、デバッグが

必要だった。しかし、それは全体の約 6 パーセントだった。

以下、2 章では、従来のソフトウェア開発手法について、3 章では、顧客管理システムについての概要と研究環境について示す。4 章では、顧客管理システムを基にした IMAH について示す。5 章では具体的な顧客管理システム、6 章では考察、7 章ではまとめについて示す。

第２章　従来のソフトウェア開発手法

２．１．ウォーターフォール・モデル

1970 年代に、ソフトウェアの開発プロセスのモデルとして最初に提案された手法である。要求仕様、分析、設計、実装、テスト、運用に工程を分け、順番に開発を進めていく手法である。各工程では、上流工程から引き渡された成果物を元に作業が行われる。また、原則的にこの順序を飛び越えたり、逆戻りしたりすることを認めないとしている。

しかし、テストフェーズがプロジェクト後半に設定されているが、これは実際の開発現場ではありえない。実際の現場では、少し開発してはテストし、それを踏まえてさらに開発を進めるといった形態がとられる場合が多い。要件仕様フェーズや設計フェーズなどの上流工程に欠陥があっても、それがプロジェクトの終盤まで発見できないということが起こる。終盤で欠陥が発見された場合、逆戻りによるコストは極めて大きなものになる。

このように、ウォーターフォール・モデルの問題点は、システム全体に対する分析や設計を一括して行うところに原因がある。最初にシステム全体を見通して問題領域を分析し、それが完成してから全体を設計するというやり方は、ソフトウェアの開発には向かないといえる。

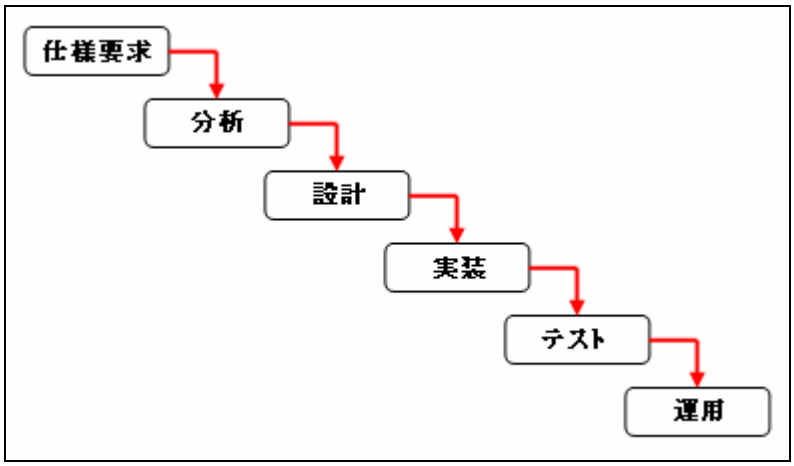

図 2-1　ウォーターフォール・モデルのプロセス

２．２．RUP（Rational Unified Process）

RUP とは、米国ラショナルソフトウェア社が開発したオブジェクト指向を前提にした開発プロセスである。従来型の開発プロセスであるウォーターフォール型とは異なるアプローチを採用している。

RUP の特徴は、ユースケースを開発の基点とし（ユースケース駆動）、開発するソフトウェアの基本構造を早期に設定する（アーキテクチャ中心）が、ウォーターフォール・モデルのように始めから完全な形の完成品を想定するのではなく、重要な機能やリスクの大きな機能をユースケース単位に構築する（反復型開発）ことなどである。

しかし、モデルを基に開発を進めていくので、曖昧さを含んだ UML 図ではバグを引き起こしてしまう。結果、多くの時間をデバッグに費やさなければならなくなる。

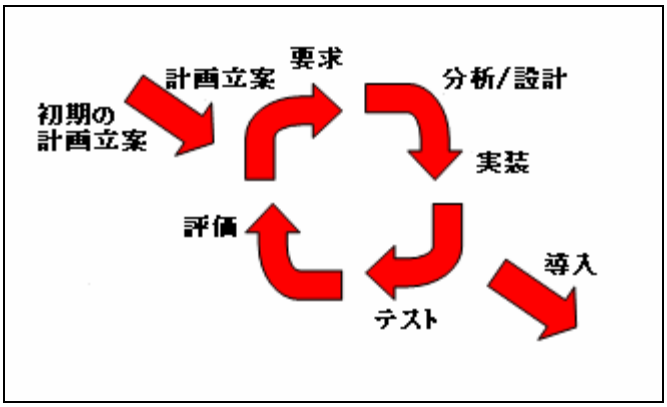

図 2-2　反復型開発のイメージ図

第３章　顧客管理システム

３．１．顧客管理システムの概要

顧客管理とは、一般的には、商品・サービスの販売などに利用することを目的として顧客の属性（住所、氏名、性別、年齢など)、購買履歴や問い合わせ履歴などを管理することである。現在、多くの顧客管理システムが存在し、多くの会社で用途にあった顧客管理システムが使われている。本研究では、会社Ａと会社Ｂ間に取引が生じたとき、会社Ａから見た顧客の会社Ｂを顧客伝票に書き込むことにする。

ここで、会社Ａによって作られる顧客伝票を表 3-1 のようにする。これを基に、顧客管理システムを考えていく。

表 3-1

顧客伝票			
顧客番号			
0001			
頭書			
作成日付	会社名	会社住所	
2008/2/1	法政株式会社	東京都千代田区富士見 2-17-1	
電話番号		メールアドレス	
03-3264-9240		ichigaya@k.hosei.ac.jp	
個人情報			
氏名	部署	役職	備考
田中博史	技術開発部	技術部長	なし
詳細書			
取引科目		金額	
部品		20000	
部品		50000	

顧客伝票は、伝票番号、頭書、個人情報、詳細書から成り立っている。また、頭書は、

更新日付、取引先の会社名、会社住所、会社電話番号、会社メールアドレスから、個人情報は、取引先の責任者の氏名、部署、役職、備考から成り立っている。詳細書は、取引科目、金額から成り立っており、それらは一つ以上で存在する。システムの利用者は、更新者(社員)と管理者である。以下に、システムの動作の詳細を述べる。

更新者

- 顧客伝票を作成する
  1. 伝票番号、日付、取引先の会社名、会社住所、会社電話番号、会社メールアドレス、取引先の責任者の名前、部署、役職、備考、取引科目、金額を入力する。
  2. 生成ボタンを押す。(すでに存在する伝票番号を入力するとエラーとなり、登録できない)
  3. システムは、データベースに書き込む。

- 顧客伝票を見る
  1. 表示ボタンを押す。
  2. システムは、データベースに書き込まれている伝票番号、取引先の会社名、会社アドレス、会社電話番号、会社メールアドレスを返す。

- 顧客伝票を検索する
  1. 取引先の会社名を入力する。
  2. 検索ボタンを押す。
  3. システムは、伝票番号、取引先の会社名、会社アドレス、会社電話番号、会社メールアドレスを返す。

- 顧客伝票を削除する
  1. 検索結果で表示された伝票から、削除したいものを選択する。
  2. 削除ボタンを押す。
  3. システムは、データベースから削除する。

- 顧客伝票を更新する
  1. 検索結果で表示された伝票から、更新したいものを選択する。
  2. 新しい情報を入力する。
  3. 更新ボタンを押す。
  4. システムは、データベースを更新する。

- 個人情報を見る
  1. 検索結果で表示された伝票から、個人情報を見たいものを選択する。
  2. List ボタンを押すと画面が移る。
  3. Personal ボタンを押すと画面が移る。
  4. システムは、取引先の責任者の名前、部署、役職、備考を返す。

- 個人情報を更新する
  1. 新しい情報を入力する。
  2. 更新ボタンを押す。
  3. システムは、データベースを更新する。

- 詳細を見る
  1. 検索結果で表示された伝票から、詳細を見たいものを選択する。
  2. List ボタンを押すと画面が移る。
  3. Detail ボタンを押すと画面が移る。
  4. システムは、取引科目、金額を返す。

- 詳細を作成する
  1. 取引科目、金額を入力する。
  2. 作成ボタンを押す。
  3. システムは、データベースに書き込む。

- 詳細を削除する
  1. 表示された詳細から、削除したいものを選択する。
  2. 削除ボタンを押す。
  3. システムは、データベースから削除する。

管理者

- 社員を生成する
  1. 社員名、コード、部署、役職、パスワードを入力する。
  2. 生成ボタンを押す。
  3. システムは、顧客伝票をデータベースに書き込む。

- 社員を検索する
  1. 社員名、コード、部署、役職、パスワードを入力する。

2. 検索ボタンを押す。
3. システムは、社員名、コード、部署、役職、パスワードを返す。

- 社員を削除する
  1. 検索結果で表示された社員から、削除したいものを選択する。
  2. 削除ボタンを押す。
  3. システムは、データベースから削除する。

- 社員を更新する
  1. 検索結果で表示された社員から、更新したいものを選択する。
  2. 新しい情報を入力する。
  3. 更新ボタンを押す。
  4. システムは、データベースを更新する。

- 顧客伝票を作成する
  1. 伝票番号、日付、取引先の会社名、住所、電話番号、メールアドレス、取引先の責任者の名前、部署、役職、備考、取引科目、金額を入力する。
  2. 生成ボタンを押す。(すでに存在する伝票番号を入力するとエラーとなり、登録できない)
  3. システムは、データベースに書き込む。

- 顧客伝票を見る
  1. 表示ボタンを押す。
  2. システムは、データベースに書き込まれている伝票番号、取引先の会社名、会社アドレス、会社電話番号、会社メールアドレスを返す。

- 顧客伝票を検索する
  1. 取引先の会社名を入力する。
  2. 検索ボタンを押す。
  3. システムは、伝票番号、取引先の会社名、会社アドレス、会社電話番号、会社メールアドレスを返す。

- 顧客伝票を削除する
  1. 検索結果で表示された伝票から、削除したいものを選択する。
  2. 削除ボタンを押す。
  3. システムは、データベースから削除する。

- 顧客伝票を更新する
  1. 検索結果で表示された伝票から、更新したいものを選択する。
  2. 新しい情報を入力する。
  3. 更新ボタンを押す。
  4. システムは、データベースを更新する。

- 個人情報を見る
  1. 検索結果で表示された伝票から、個人情報を見たいものを選択する。
  2. List ボタンを押すと画面が移る。
  3. Personal ボタンを押すと画面が移る。
  4. システムは、取引先の責任者の名前、部署、役職、備考を返す。

- 個人情報を更新する
  1. 新しい情報を入力する。
  2. 更新ボタンを押す。
  3. システムは、データベースを更新する。

- 詳細を見る
  1. 検索結果で表示された伝票から、詳細を見たいものを選択する。
  2. List ボタンを押すと画面が移る。
  3. Detail ボタンを押すと画面が移る。
  4. システムは、取引科目、金額を返す。

- 詳細を作成する
  1. 取引科目、金額を入力する。
  2. 作成ボタンを押す。
  3. システムは、データベースに書き込む。

- 詳細を削除する
  1. 表示された詳細から、削除したいものを選択する。
  2. 削除ボタンを押す。
  3. システムは、データベースから削除する。

３．２．研究環境

本研究で使用するツールは、以下の通りである。

- Maven
  Java プロジェクトを管理するツールである。本研究で使用した Maven バージョンは、2.0.4 である。

- JBOSS
  Java アプリケーションサーバである。本研究で使用した JBSS バージョンは、4.0.3 である。JBoss は、ウェブサーバ(Struts)、データベースサーバ(Hypersonic)の機能を持っており、本研究では、それらを利用する。

- MagicDraw
  UML モデリングツールである。本研究で使用したバージョンは、Personal Edition の 14.0 である。

- AndroMDA
  MDA を実現する開発ツールでる。AndroMDA は、オープンソースで開発が進められている。

- Eclipse
  オープンソースの統合ソフトウェア開発環境である。本研究では、Java での開発に使用する。

第４章　線形モジュール追加型抽象階層

本章では、顧客管理システムを例にとり、線形モジュール追加型抽象階層を用いたソフトウェア開発法について述べる。

モジュール追加型抽象階層は次の階層によって成り立っている。ホモトピーレベル、集合論レベル、位相空間レベル、接着空間レベル、セル空間レベル、表現レベル、可視レベルの７つである。以下、各項でそれぞれの階層の詳細を述べる。

４．１．ホモトピーレベル

ホモトピーレベルは、階層の中で一番抽象的なレベルである。この層では、それぞれの空間を定義する。また、これらの空間の関係もこの層で定義される。

会社間で取引が発生したとき、顧客伝票が作成される。顧客伝票は、伝票番号、頭書、個人情報、詳細書から成る。これらをファイバー束に写像する。ここで、全空間を E、底空間を B、ファイバーを F とする。ファイバー束は、全空間を底空間とファイバーに分け、次のように表される。

$$E = B \times F \qquad (1)$$

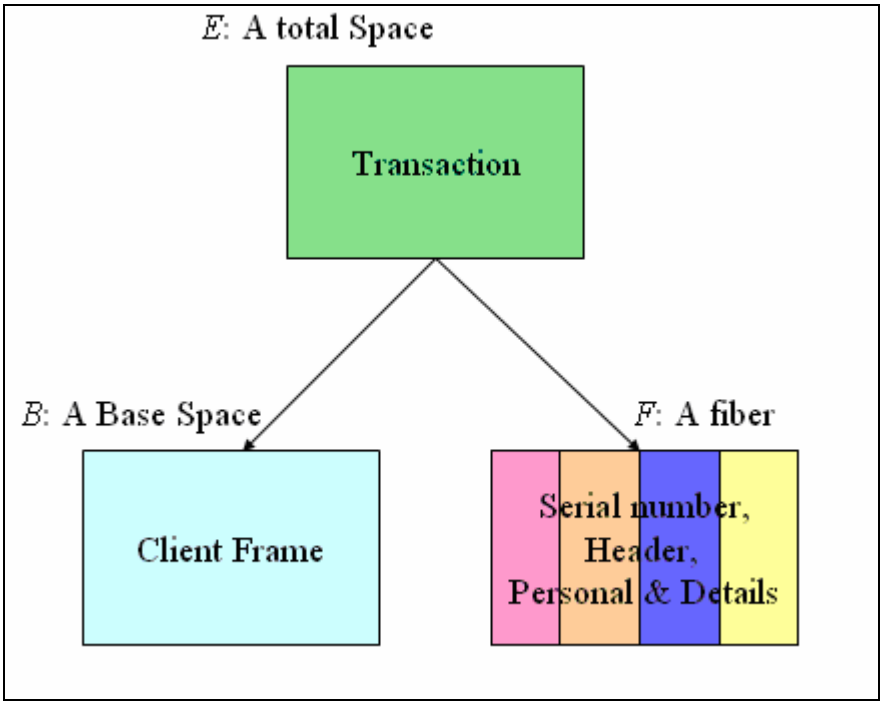

図 4-1　顧客伝票

底空間 B は、顧客伝票空間 C であり、空の顧客伝票と考える。ファイバー F は、顧客ナンバー空間 S、頭書空間 H、個人空間 P、詳細空間 D から成り、次のように表される。

$$F = S \times H \times P \times D \qquad (2)$$

取引から、顧客伝票に移す作業をファイバー束 $\xi = (E, B, F, p)$ で表す。また、空間の関係は、図 4-2 のようになる。

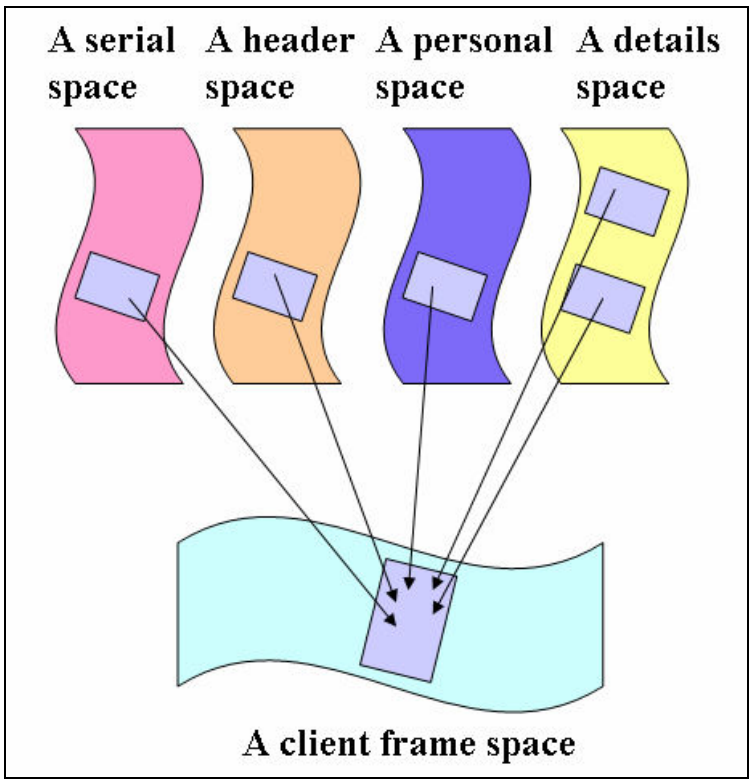

図 4-2　顧客システムの空間

４．２．集合論レベル

この層では、それぞれの空間の要素を定義する。

顧客伝票を集めたものを顧客帳 C とすると、C は、顧客伝票 c_i の集合で構成されているので、次のように表される。

$$C = \{c_1, c_2, c_3, \cdots, c_n\} \qquad (3)$$

顧客伝票は、伝票番号 s、頭書 h、個人情報 p、詳細書 DS で構成されており、次のように表される。

$$c_i = (s_i, h_i, p_i, DS_i) \in C \qquad (4)$$

ここで、頭書のリスト H について考える。H は、日付 t、取引先会社名 cn、住所 ca、電話番号 cp、メールアドレス cm で構成されており、次のように表される。

$$H = \{h_1, h_2, h_3, \cdots, h_n\} \qquad (5)$$

$$h_j = (t_j, cn_j, ca_j, \mathrm{cp}_j, cm_j) \in H \qquad (6)$$

次に、個人情報のリスト P について考える。P は、取引先の責任者の名前 n、部署 d、役職 s、備考 r で構成されており、次のように表される。

$$P = \{p_1, p_2, p_3, \cdots, p_n\} \qquad (7)$$

$$p_j = (n_j, d_j, s_j, r_j) \in P \qquad (8)$$

最後に、詳細書のリスト D について考える。D は、取引科目 ds、金額 dm で構成されており、次のように表される。

$$D = \{d_1, d_2, d_3, \cdots, d_n\} \qquad (9)$$

$$d_j = (ds_j, dm_j) \in D \qquad (10)$$

４．３．位相空間レベル

この層では、位相を導入し、連続性、近さの概念を定義する。

ホモトピーレベルで定義された空間は、全て離散的である。ここでは、離散集合での位相空間を次のように定義する。集合 S に導入された位相空間 T は、集合 S の要素のいかなる集まりも位相空間の要素とする。よって、顧客帳 C に対する位相空間 TC は、次のように表される。

$$TC = \{\phi, c_1, c_2, c_3, \ldots, c_n, (c_1, c_2), (c_1, c_3), \ldots, (c_{n-1}, c_n), \ldots, (c_1, c_2, c_3, \ldots, c_n)\} \quad (11)$$

ここで、ファイバー束について少し見ていく。底空間 B は、

$$B = \{\phi, c_1, c_2, c_3, \ldots, c_n, (c_1, c_2), (c_1, c_3), \ldots, (c_{n-1}, c_n), \ldots, (c_1, c_2, c_3, \ldots, c_n)\} \quad (12)$$

であり、ファイバー F は、$F = S \times H \times P \times D$（(2)より）である。これより、$B \times F$ は $c_i \in B$ に対して、s_i、h_i、p_i、DS_i を与える。これを示したのが図 4-3 である。

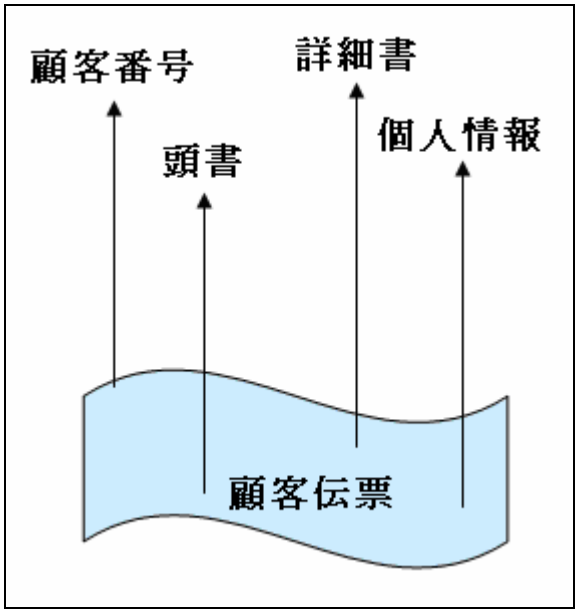

図 4-3　顧客管理システムのファイバー束

４．４．接着空間レベル

この層では、各空間の吸着、分離を定義する。

ここでは、会社間取引がまだ行われていない状態から、徐々に取引が行われる状態について考える。これは、顧客長 Cが、$C=\{\}$ から $C=\{c_1,c_2,c_3,\ldots,c_n\}$ と変わっていく状態を表す。ここで、

$B=C$　　((1)より)

$F=S\times H\times P\times D$　　((2)より)

なので、取引が全く行われていない状態では、空間Cと空間$S\times H\times P\times D$とは完全に分離していると考えられる。即ち、$C\sqcup(S\times H\times P\times D)$で表すことができる。これは、$(C\sqcup S)\times(C\sqcup H)\times(C\sqcup P)\times(C\sqcup D)$と書き換えることができる。

ここで、取引が初めて行われたとする。このとき、顧客帳にその内容が書き込まれる。これは、何も書かれていない顧客伝票 c_iに、各内容 s_i、h_i、p_i、DS_iが書き込まれると考えることができる。これを、関数 fを用いて表す。ここでの関数の役割は、顧客伝票に書き込むという行為である。ただし、記入間違い、情報の消去の可能性がある。これは、顧客伝票と取引は、一時的にくっついていると考えられる。接着関数は、このような性質を持っている。

$$f:C_0\to S\times H\times P\times D \qquad (13)$$

ここで、詳細書 DSについて考える。顧客伝票 c_{ij}には、$DS=\{d_{j1},d_{j2},d_{j3},\ldots d_{jm}\},d_{jk}\in D$が書き込まれる。この状況を説明したのが図 4-4 である。

接着空間 D_fは、以下のように表せる。

$$\begin{aligned}D_f&=D\sqcup C/\sim\\&=D\sqcup_f C\\&=D\sqcup C/(c_j\sim f(y)\,|\,c_i\in C,y\in D_0)\end{aligned}\qquad (14)$$

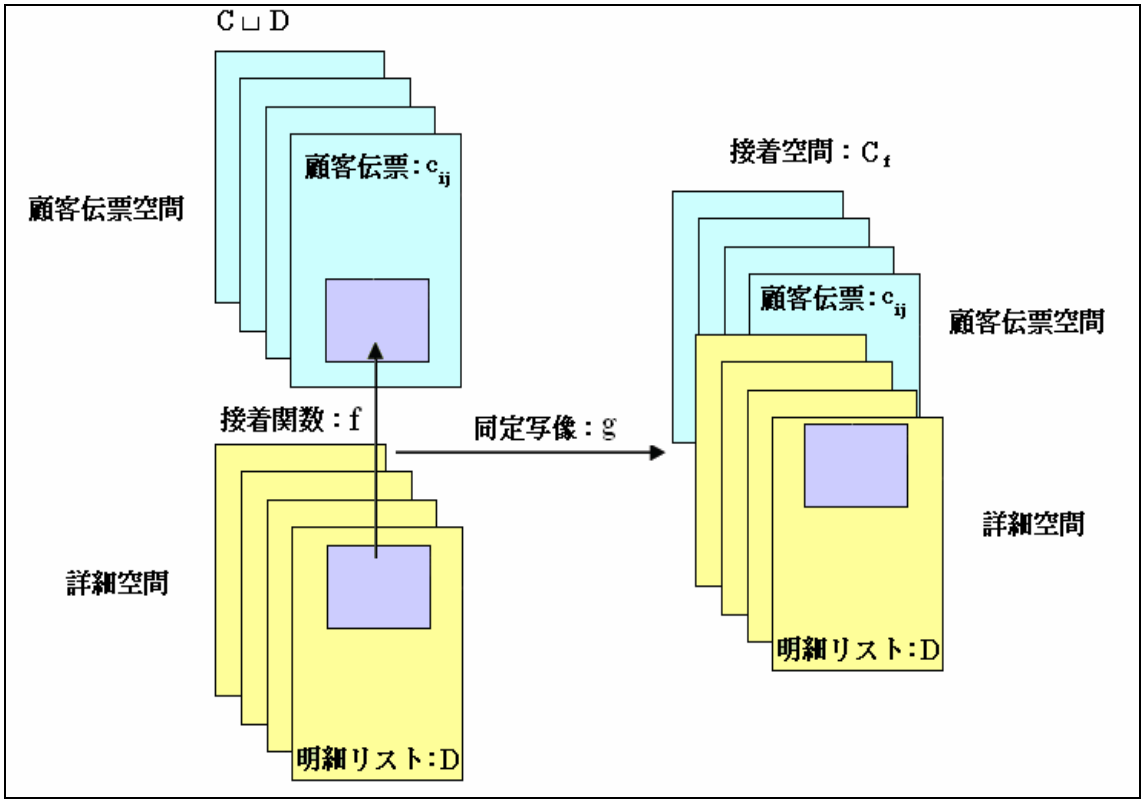

図 4-4　顧客伝票と詳細リストの接着

同様に、伝票番号 S、頭書 H、個人情報 P も以下のように接着空間を定義することができる。

$$\begin{aligned} S_f &= S \sqcup C / \sim \\ &= S \sqcup_f C \\ &= S \sqcup C / (c_j \sim f(y) \mid c_i \in C, y \in S_0) \end{aligned} \qquad (15)$$

$$\begin{aligned} P_f &= P \sqcup C / \sim \\ &= P \sqcup_f C \\ &= P \sqcup C / (c_j \sim f(y) \mid c_i \in C, y \in P_0) \end{aligned} \qquad (16)$$

$$\begin{aligned} H_f &= H \sqcup C / \sim \\ &= H \sqcup_f C \\ &= H \sqcup C / (c_j \sim f(y) \mid c_i \in C, y \in H_0) \end{aligned} \qquad (17)$$

４．５．セル空間レベル

この層では、セルを利用することで物理的な構造を構築する。

セル構造空間では、各オブジェクトはセルのセットとして表現され、それらは位相空間上に存在する。その概念の理解のために、2 次元の 3 角形を考えると、3 角形は 0 次元の点の空間として表され、それに 1 次元の線がアタッチング関数により結合され、さらに、2 次元の面がアタッチング関数により結合されることにより、順番に構成される。それは図 4-5 のようになる。

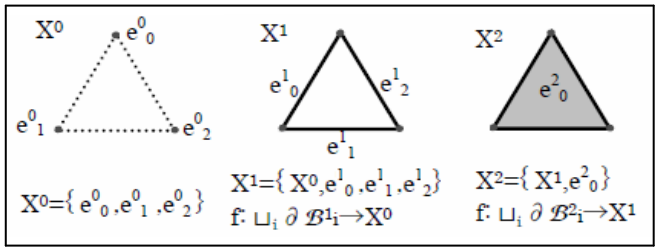

図 4-5　三角形の表現

伝票番号 s_i は 1 つの整数型の変数を持ち、B_{si}^{1} と表現できる。伝票番号の空間 S は伝票番号の互いに素な集合であるので、$\sqcup\ B_{si}^{1}$ によって表される。頭書 h_i 、個人情報 p_i、詳細 d_i はそれぞれ 5 つ、4 つ、2 つの変数を持つので、それぞれの空間 H、P、D は、$\sqcup B_{hi}^{5}$、$\sqcup B_{pi}^{4}$、$\sqcup B_{di}^{2}$ で表される。顧客伝票 c_i は、これらの要素の入れ物であり変数を持たないので、空間 C は $\sqcup B_{ci}^{0}$ で表される。接着空間のところで見たように、これらは接着関数 f で関連付けられているので、これらの関係は次のように表すことが出来る。

$$f_1 : \partial^1 B_{sj}^{\ 1} \to \partial^1 B_j^{\ 1} \quad or \quad \partial^1 B_j^{\ 1} \sqcup_{f_1} B_{sj}^{\ 1} / \sim \qquad (18)$$

$$f_2 : \partial^5 B_{hj}^{\ 5} \to \partial^1 B_j^{\ 1} \quad or \quad \partial^1 B_j^{\ 1} \sqcup_{f_2} B_{hj}^{\ 5} / \sim \qquad (19)$$

$$f_3 : \partial^4 B_{pj}^{\ 4} \to \partial^1 B_j^{\ 1} \quad or \quad \partial^1 B_j^{\ 1} \sqcup_{f_3} B_{hj}^{\ 4} / \sim \qquad (20)$$

$$f_4 : \sqcup \partial^2 B_{djj}^{\ 2} \to \partial^1 B_j^{\ 1} \quad or \quad \partial^1 B_j^{\ 1} \sqcup_{f_4} (B_{djj}^{\ 2}) / \sim \qquad (21)$$

これを用いて CW 空間を構成することができる。

詳細はその変数として、取引科目と金額を持っており、それぞれを明確に区別するためにインデックスも必要となる。これらは DS_i のスケルトン X_{detail}^{0} として使用される。

$$X_{\mathrm{det}\ ail}^{\ 0} = \{ e_{did}^{\ 0}{}_1, \cdots, e_{did}^{\ 0}{}_k, e_{item}^{\ 0}{}_1, \cdots, e_{item}^{\ 0}{}_k, e_{amount}^{\ 0}{}_1, \cdots e_{amount}^{\ 0}{}_k \} \qquad (22)$$

スケルトン $X_{detail}{}^{1}$ に関しては、インデックス、取引科目、金額の中の 2 つが 1 次元のセルを通して接着される。

$$
\begin{aligned}
X_{\det ail}{}^{1} = \{ & X_{\det ail}{}^{0}, e_{diditem}{}^{1}{}_{1}, \cdots, e_{diditem}{}^{1}{}_{k}, e_{itemamount}{}^{1}{}_{1}, \cdots, e_{itemamount}{}^{1}{}_{k}, e_{amountdid}{}^{1}{}_{1}, \cdots, e_{amountdid}{}^{1}{}_{k} \\
& | f_1 : \partial e_{diditem}{}^{1}{}_{i} \to e_{did}{}^{0}{}_{i}, f_2 : \partial e_{diditem}{}^{1}{}_{i} \to e_{item}{}^{0}{}_{i}, f_3 : \partial e_{itemamount}{}^{1}{}_{i} \to e_{item}{}^{0}{}_{i}, f_4 : \\
& \partial e_{itemamount}{}^{1}{}_{i} \to e_{amount}{}^{0}{}_{i}, f_5 : \partial e_{amountdid}{}^{1}{}_{i} \to e_{amount}{}^{0}{}_{i}, f_6 : \partial e_{amountdid}{}^{1}{}_{i} \to e_{did}{}^{0}{}_{i} \}
\end{aligned}
\tag{23}
$$

スケルトン $X_{detail}{}^{2}$ は以下のように得られる。

$$
\begin{aligned}
X_{\det ail}{}^{2} = \{ & X_{\det ail}{}^{1}, e_{\det ail}{}^{2}{}_{1}, \cdots e_{\det ail}{}^{2}{}_{k} \mid f_1 : \partial e_{\det ail}{}^{2}{}_{i} \to e_{diditem}{}^{1}{}_{i}, f_2 : \partial e_{\det ail}{}^{2}{}_{i} \to e_{itemamount}{}^{1}{}_{i}, \\
& f_3 : \partial e_{\det ail}{}^{2}{}_{i} \to e_{amountdid}{}^{1}{}_{i} \}
\end{aligned}
\tag{24}
$$

次に、顧客番号のリストは、以下のように得られる。

$$
X_{number}{}^{0} = \{ e_{nid}{}^{0}{}_{1}, \cdots, e_{nid}{}^{0}{}_{k}, e_{serial}{}^{0}{}_{1}, \cdots, e_{serial}{}^{0}{}_{k} \} \tag{25}
$$

$$
X_{number}{}^{1} = \{ X_{number}{}^{0}, e_{number}{}^{1}{}_{1}, \cdots, e_{number}{}^{1}{}_{k} \mid f_1 : \partial e_{number}{}^{1}{}_{i} \to e_{nid}{}^{0}{}_{1}, f_2 : \partial e_{number}{}^{1}{}_{i} \to e_{serial}{}^{0}{}_{i} \} \tag{26}
$$

頭書リストについては以下のようになる。

$$
\begin{aligned}
X_{header}{}^{0} = \{ & e_{hid}{}^{0}{}_{1}, \cdots, e_{hid}{}^{0}{}_{k}, e_{date}{}^{0}{}_{1}, \cdots, e_{date}{}^{0}{}_{k}, e_{comname}{}^{0}{}_{1}, \cdots, e_{comname}{}^{0}{}_{k}, e_{comaddress}{}^{0}{}_{1}, \cdots, \\
& e_{comaddress}{}^{0}{}_{k}, e_{comphone}{}^{0}{}_{1}, \cdots, e_{comphone}{}^{0}{}_{k}, e_{comemail}{}^{0}{}_{1}, \cdots, e_{comemail}{}^{0}{}_{k} \}
\end{aligned}
\tag{27}
$$

$$
\begin{aligned}
X_{header}{}^{5} = \{ & X_{header}{}^{4}, e_{header}{}^{5}{}_{1}, \cdots, e_{header}{}^{5}{}_{k} \mid \\
& f_1 : \partial e_{header}{}^{5}{}_{i} \to e_{hiddatecomnamecomaddresscomphone}{}^{4}{}_{i} \\
& f_2 : \partial e_{header}{}^{5}{}_{i} \to e_{datecomnamecomaddresscomphonecomemail}{}^{4}{}_{i}, \\
& f_3 : \partial e_{header}{}^{5}{}_{i} \to e_{comnamecomaddresscomphonecomemailhid}{}^{4}{}_{i}, \\
& f_4 : \partial e_{header}{}^{5}{}_{i} \to e_{comaddresscomphonecomemailhiddate}{}^{4}{}_{i}, \\
& f_5 : \partial e_{header}{}^{5}{}_{i} \to e_{comphonecomemailhiddatecomname}{}^{4}{}_{i}, \\
& f_6 : \partial e_{header}{}^{5}{}_{i} \to e_{comemailhiddatecomnamecomaddress}{}^{4}{}_{i} \}
\end{aligned}
\tag{28}
$$

個人情報リストについては以下のようになる。

$$X_{personal}^{0} = \{e_{pid}^{0}{}_{1}, \cdots, e_{pid}^{0}{}_{k}, e_{name}^{0}{}_{1}, \cdots, e_{name}^{0}{}_{k}, e_{department}^{0}{}_{1}, \cdots, e_{department}^{0}{}_{k}, e_{status}^{0}{}_{1}, \cdots, e_{status}^{0}{}_{k}, e_{remarks}^{0}{}_{1}, \cdots, e_{remarks}^{0}{}_{k}\} \quad (29)$$

$$\begin{aligned} X_{personal}^{4} = \{ & X_{personal}^{3}, e_{personal}^{4}{}_{1}, \cdots, e_{personal}^{4}{}_{k} \mid \\ & f_1 : \partial e_{personal}^{4}{}_{i} \rightarrow e_{pidnamedepartmentstatus}^{3}{}_{i}, \\ & f_2 : \partial e_{personal}^{4}{}_{i} \rightarrow e_{namedepartmentstatusremarks}^{3}{}_{i}, \\ & f_3 : \partial e_{personal}^{4}{}_{i} \rightarrow e_{departmentstatusremarkspid}^{3}{}_{i}, \\ & f_4 : \partial e_{personal}^{4}{}_{i} \rightarrow e_{statusremarkspidname}^{3}{}_{i}, \\ & f_5 : \partial e_{personal}^{4}{}_{i} \rightarrow e_{remarkspidnamedepartment}^{3}{}_{i}\} \end{aligned} \quad (30)$$

最後に、顧客帳について考える。

$$X_{journal}^{0} = \{e_{jid}^{0}{}_{1}, \cdots, e_{jid}^{0}{}_{k}\} \quad (31)$$

これに、課客番号、頭書、個人情報、詳細を接着する。

$$\begin{aligned} X_{journal}^{1} = \{ & X_{journmal}^{0}, e_{journalnumber}^{1}{}_{1}, \cdots, e_{journalnumber}^{1}{}_{k} \mid \quad f_1 : \partial e_{journalnumber}^{1}{}_{i} \rightarrow e_{jid}^{0}{}_{i}, f_2 : \\ & \partial e_{journalnumber}^{1}{}_{i} \rightarrow e_{number}^{0}{}_{i}\} \end{aligned} \quad (32)$$

$$\begin{aligned} X_{journal}^{6} = \{ & X_{journal}^{1}, e_{journalheader}^{6}{}_{1}, \cdots, e_{journalheader}^{6}{}_{k} \mid f_1 : \partial^5 e_{journalheader}^{6}{}_{i} \rightarrow e_{journalnumber}^{1}{}_{i}, \\ & f_2 : \partial e_{journalheader}^{6} \rightarrow e_{header}^{5}{}_{i}\} \end{aligned} \quad (33)$$

$$\begin{aligned} X_{journal}^{10} = \{ & X_{journal}^{6}, e_{journalpersonal}^{10}{}_{1}, \cdots, e_{journalpersonal}^{10}{}_{k} \mid f_1 : \partial^4 e_{journalpersonal}^{10}{}_{i} \rightarrow e_{journalheadr}^{6}{}_{i}, \\ & f_2 : \partial^6 e_{journalpersonal}^{10}{}_{i} \rightarrow e_{personal}^{4}{}_{i}\} \end{aligned} \quad (34)$$

$$X_{journal}^{12} = \{X_{journal}^{10}, e_{journaldetail}^{12}{}_1, \cdots, e_{journaldetail}^{12}{}_k \mid f_1 : \partial^2 e_{journaldetail}^{12}{}_i \to e_{journalpersonal}^{10}{}_i, \quad f_2 : \partial^{10} e_{journaldetail}^{12}{}_i \to e_{detail}^{2}{}_i\} \tag{35}$$

これを示したものが図 4-6 である。

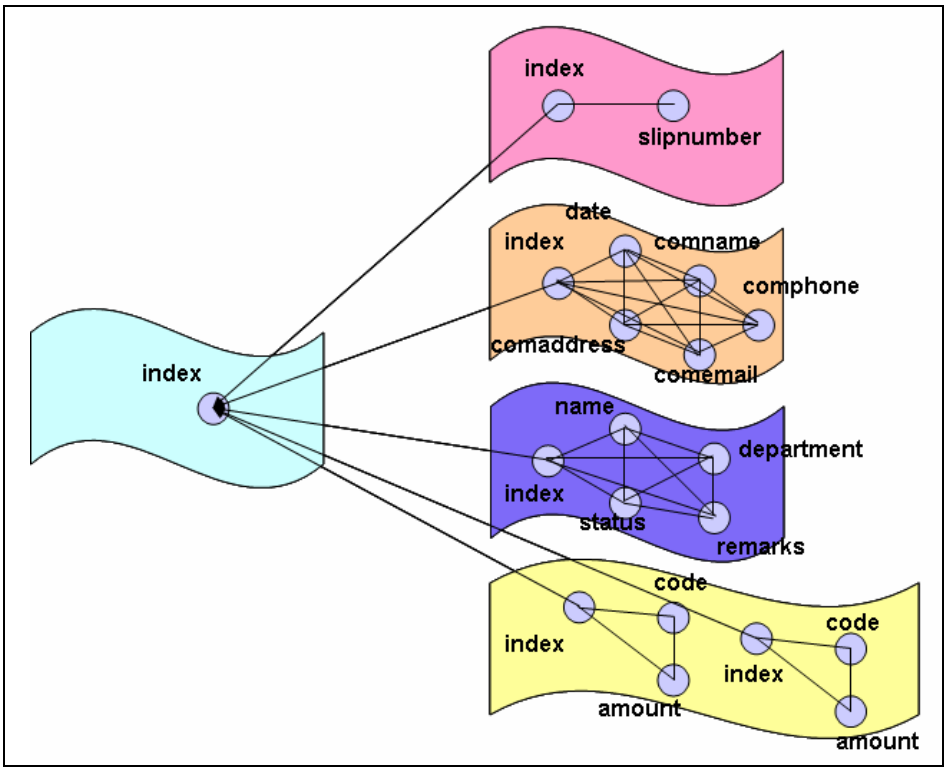

図 4-6　顧客管理システムのセル空間

４．６．表現レベル

この層では、ここまでに決まった空間の関係をクラス図で表す。クラス図は図 3-7 のようになる。

ファイバー束での底空間としての顧客帳の空間Cは Kokyaku というクラスで表されている。また、ファイバーを構成する空間に対しては、伝票番号の空間Sは SlipNumber というクラスで、頭書のリストの空間Hは Header というクラスで、個人情報のリストの空間Pは Personal というクラスで、明細のリストの空間Dは Detail というクラスで実現されている。また、接着関数は、クラス間を結ぶ関連(Association links)で表されている。これは、接着空間レベルで定められた保存量を維持しているものである。伝票番号、頭書に対する多重度は 1 であるが、明細への多重度は n となっていて、複数許される。これは、セル空間レベルで決められた次元に対する不変量を保持している。

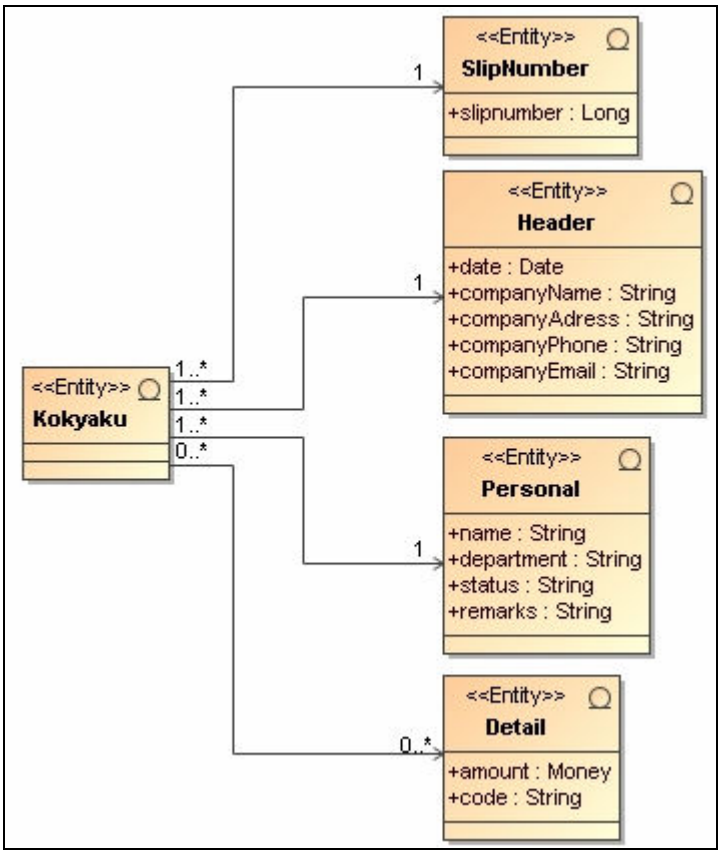

図 4-7　顧客管理システムのクラス図

４．７．可視レベル

可視レベルは、階層の中で一番具体的なレベルである。この層では、表現レベルで書かれた UML を基に AndroMDA を利用して、プログラムを自動生成する。

AndroMDA を用いて、上のクラス図からデータベースを作成する。JSP での出力結果を以下の図に示す。また、クラス図のそれぞれのクラスにステレオタイプ<<manageable>>を加えることで、以下の web ページを自動生成できる。

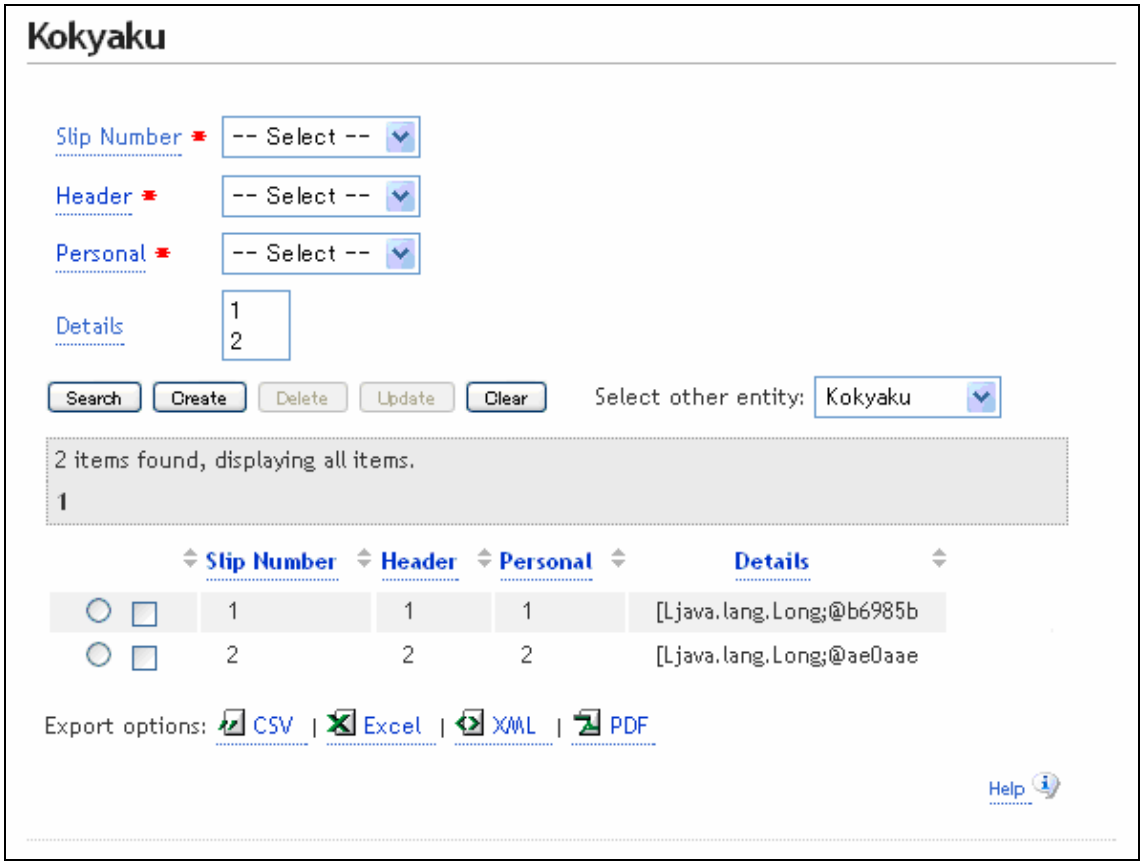

図 4-8

図 4-8 において、検索結果テーブルの Details 部分に文字化けが起きているが、値は正しく代入されている。


Slip Number
Slipnumber
Search
Create
Delete
Update
Clear
Select other entity:
Slip Number
2 items found, displaying all items.
1
Slipnumber
1
2
Export options:
CSV
Excel
XML
PDF
Help

図 4-9

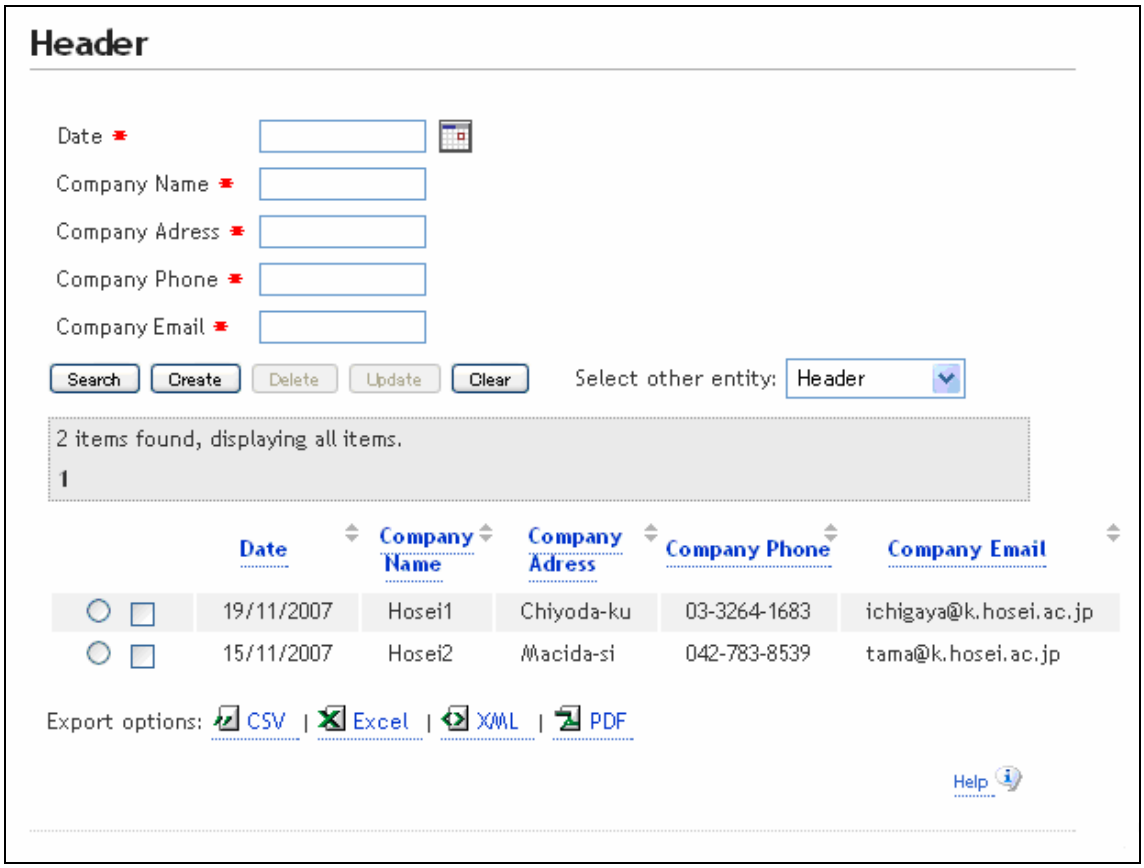
Header
Date
Company Name
Company Adress
Company Phone
Company Email
Search
Create
Delete
Update
Clear
Select other entity:
Header
2 items found, displaying all items.
1
Date
Company Name
Company Adress
Company Phone
Company Email
19/11/2007
Hosei1
Chiyoda-ku
03-3264-1683
ichigaya@k.hosei.ac.jp
15/11/2007
Hosei2
Machida-si
042-783-8539
tama@k.hosei.ac.jp
Export options:
CSV
Excel
XML
PDF
Help

図 4-10

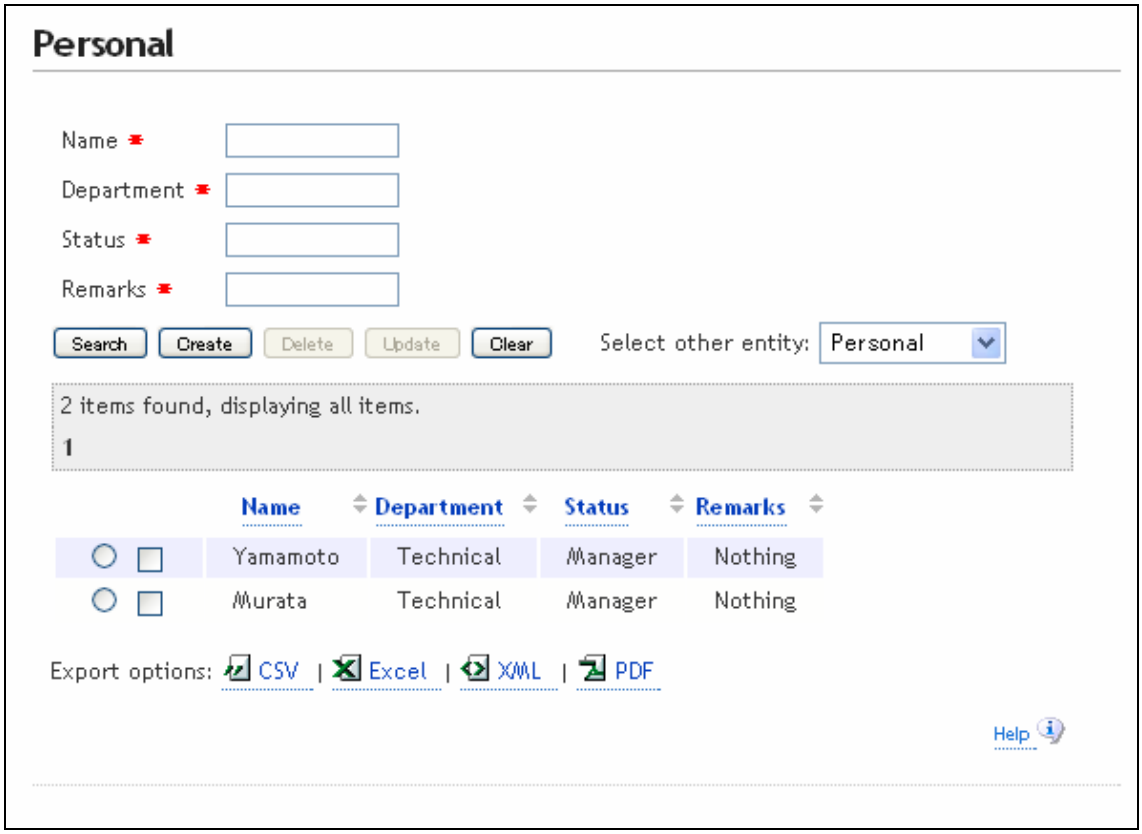
Personal
Name
Department
Status
Remarks
Search
Create
Delete
Update
Clear
Select other entity:
Personal
2 items found, displaying all items.
1
Name
Department
Status
Remarks
Yamamoto
Technical
Manager
Nothing
Murata
Technical
Manager
Nothing
Export options:
CSV
Excel
XML
PDF
Help

図 4-11

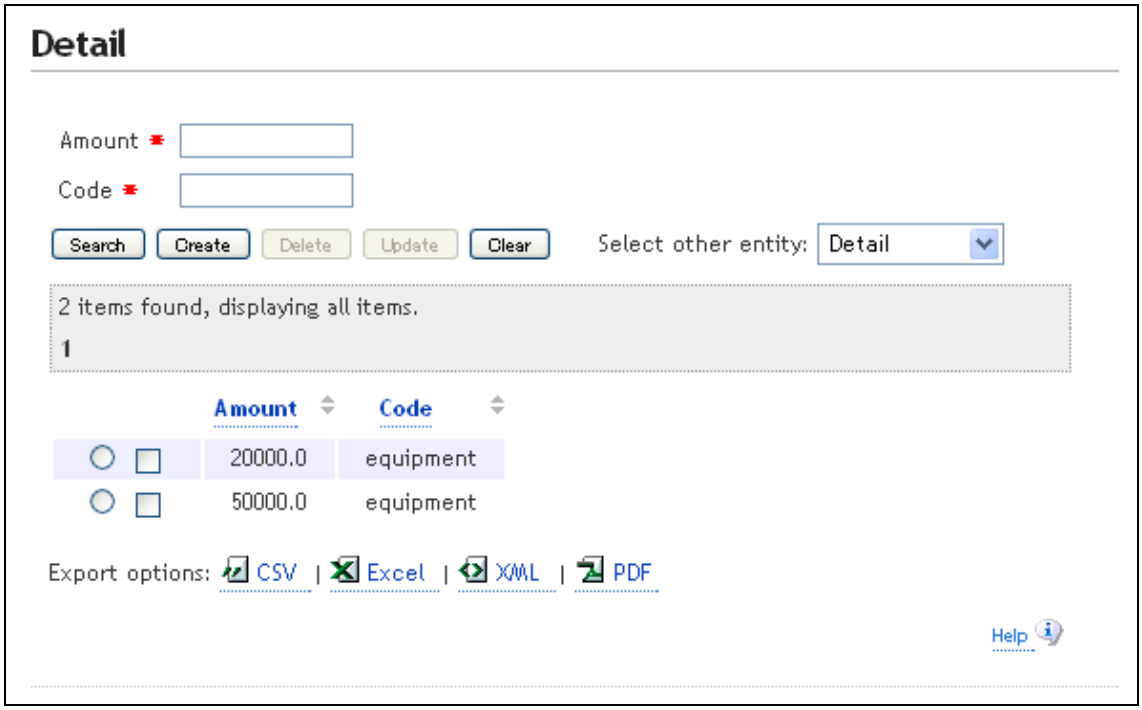
Detail
Amount
Code
Search
Create
Delete
Update
Clear
Select other entity:
Detail
2 items found, displaying all items.
1
Amount
Code
20000.0
equipment
50000.0
equipment
Export options:
CSV
Excel
XML
PDF
Help

図 4-12

第５章　具体的な顧客管理システム

クラスが出来上がり自動生成された JSP（Java サーバー・ページ）を用いて、システムの動きを把握することが出来た。ここでは、各データベースを用いて、操作性のよい JSP を考える。操作は、作成、検索、更新、削除である。これらの機能を加えるのに、新たなクラス図が必要になる。これを図 5-1 に示す。ここで追加されたクラス図は、ビジネスロジックの部分であり、プログラムは自動生成されない。よって、自らプログラムを書く必要がある。

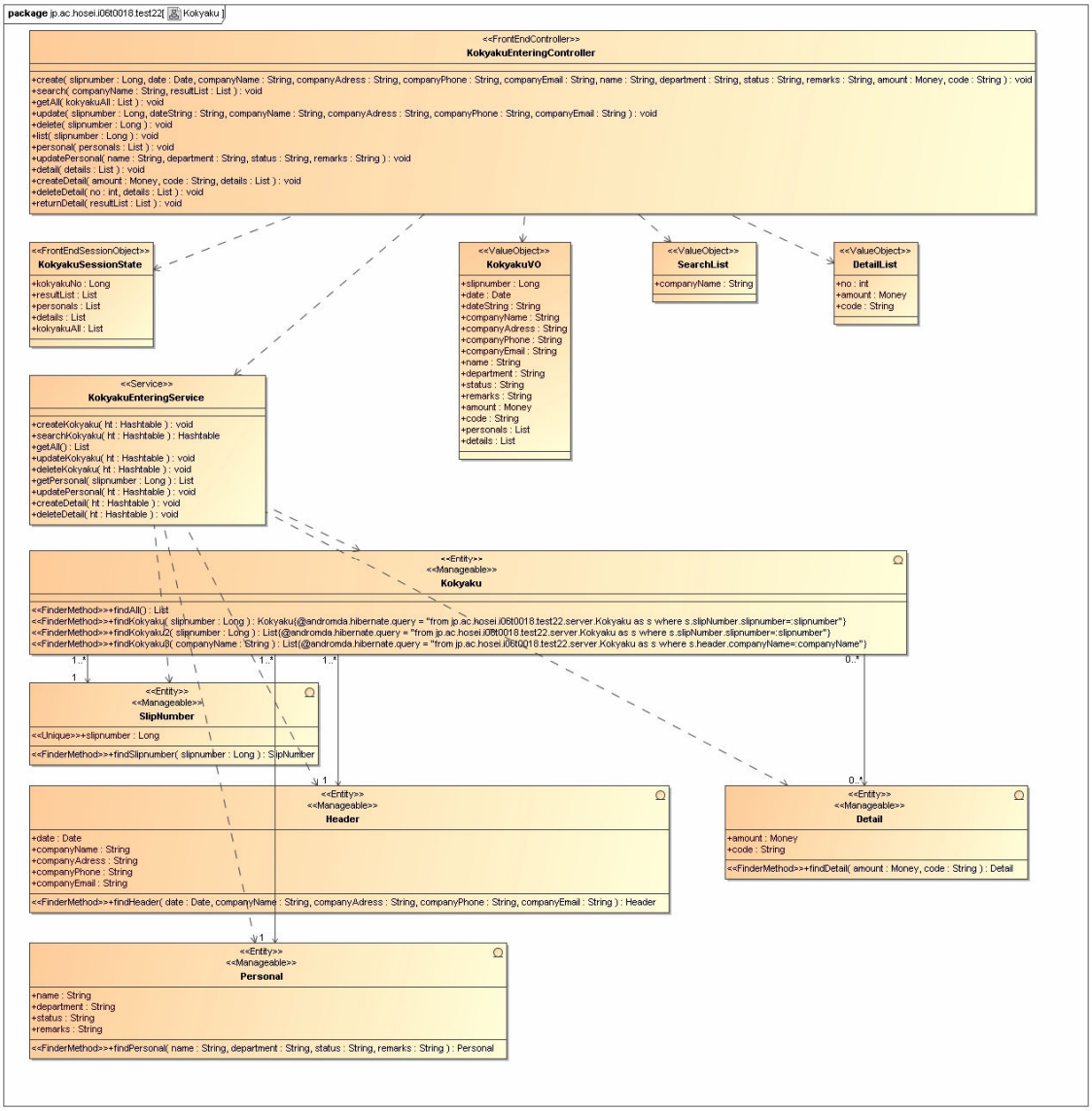

図 5-1　具体的な顧客管理システムのクラス図

追加したクラスは以下の通りである。

● KokyakuEnteringController

ステレオタイプ<<FrontEndController>>を加える。

Web 側の操作を定義する。画面表示データの設定やサービスメソッドの呼び出しをする。

● KokyakuEnteringService

ステレオタイプ<<Service>>を加える。

EJB 側の操作を定義する。データベースからデータを取得し、要求された形に加工して戻す。

● KokyakuVO、SerchList、DetailList

ステレオタイプ<<ValueObject>>を加える。

Controller と Service 間でやり取りする値を格納する。

● KokyakuSessionState

ステレオタイプ<<FrontEndSessionObject>>を加える。

ページをまたいで同じデータを保持する。

次に、ユースケース図は、図 5-2 のようになる。

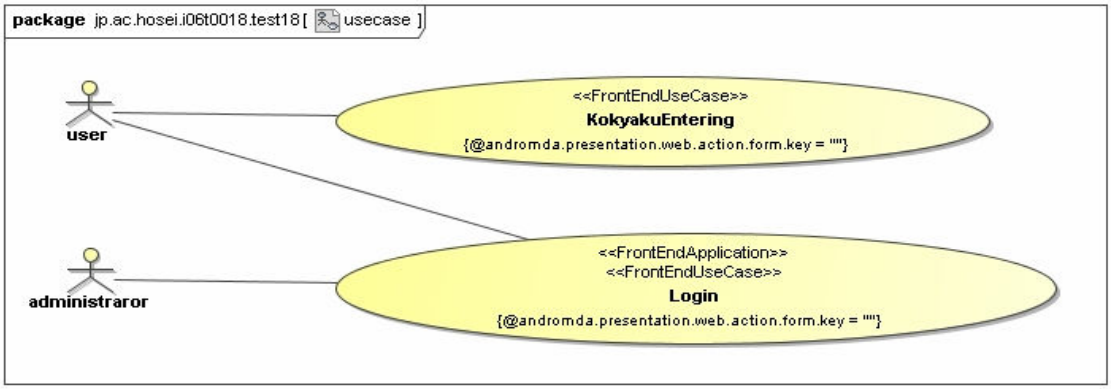

図 5-2　ユースケース図

● KokyakuEntering

ステレオタイプ<<FrontEndUseCase>>

設定したユースケースをアプリケーションに含めることを意味する。

● Login

ステレオタイプ<<FrontEndApplication>>

アプリケーションの開始点を意味する。

アクティビティ図は、図 5-3 に示す。

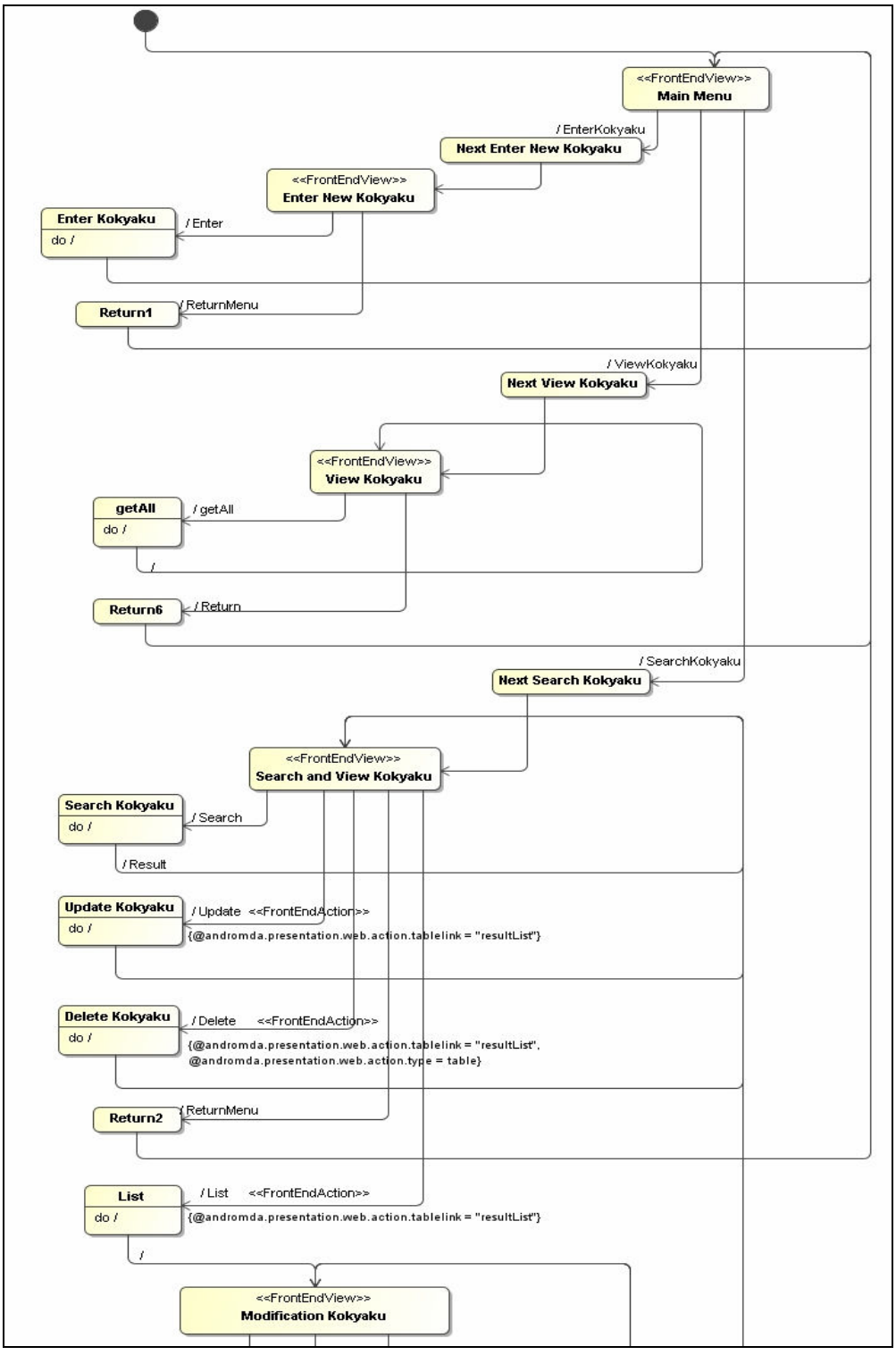

図 5-3　アクティビティ図

ここで、本アプリケーションにおけるアーキテクチャを図 5-4 に示す。

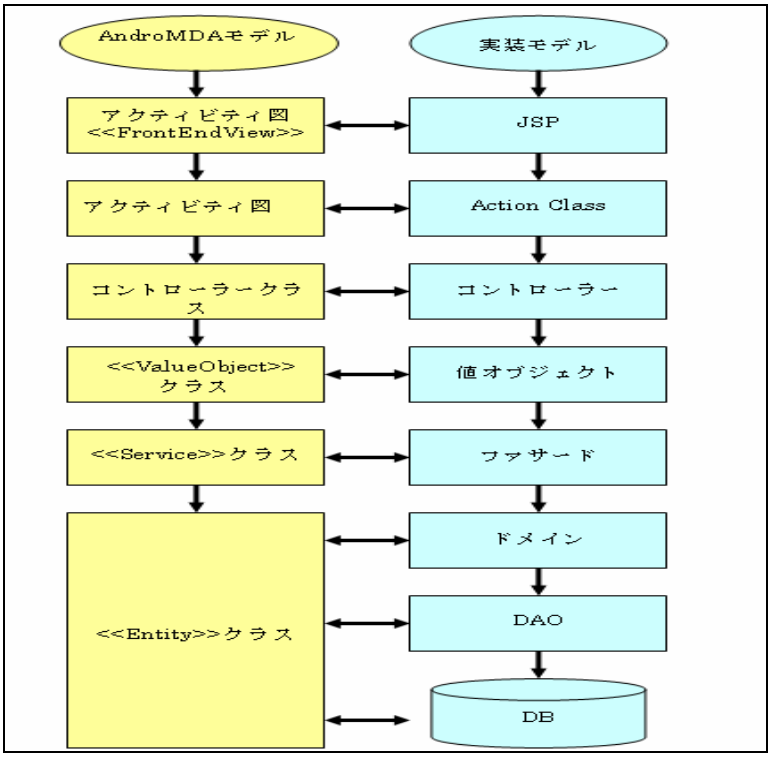

図 5-4　アーキテクチャ図

- JSP
  画面表示。

- Action Class
  コントローラーに対するロジックの呼び出しや画面の遷移を制御する。

- コントローラー
  画面表示データの設定やドメイン層のサービスメソッドの呼び出しをする。

- 値オブジェクト
  コントローラー層とドメイン層の間でやり取りする値を格納する。

- ファサード
  サービスメソッドを用意し、DAO などを利用して DB よりデータを取得し、加工する。

- ドメイン
  永続化されるドメインクラス DB のテーブルにマップする。

● DAO

DB へのアクセスを制御する。

● DB

データを永続的に保持する。

また、本研究のプロジェクトの構成は以下のようになっている。

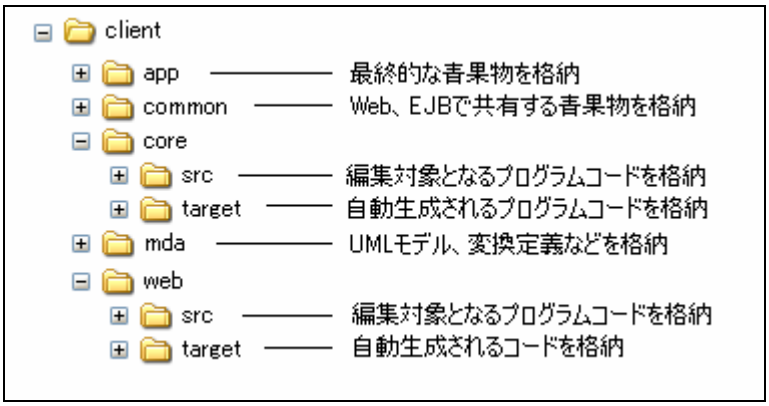

図 5-5　プロジェクトの構成

フォルダのうち、「target」フォルダについては、ビルドのたびに再作成される。「src」については、ファイルが存在している限り再作成はされない。

以下に、顧客管理システムの実行画面を示す。

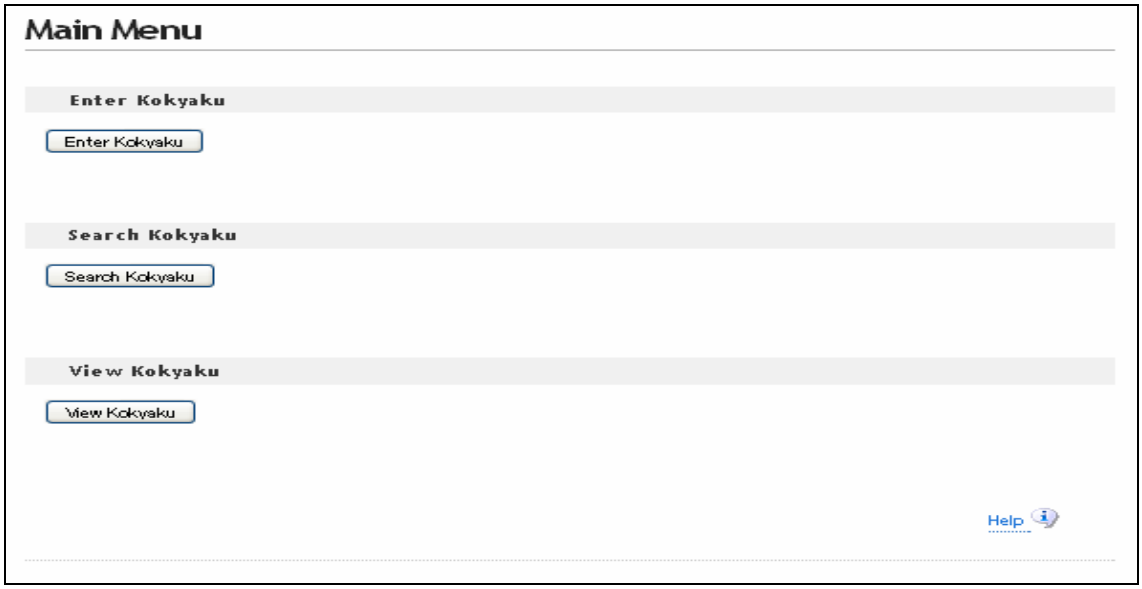

図 5-6

図 5-6 が、メイン画面である。「Enter Kokyaku」ボタンを押すと顧客伝票作成画面（図 5-7）へ遷移する。「Search Kokyaku」ボタンを押すと顧客検索画面（削除、更新）（図 5-8）へ遷移する。「View Kokyaku」ボタンを押すと顧客確認画面（図 5-9）へ遷移する。

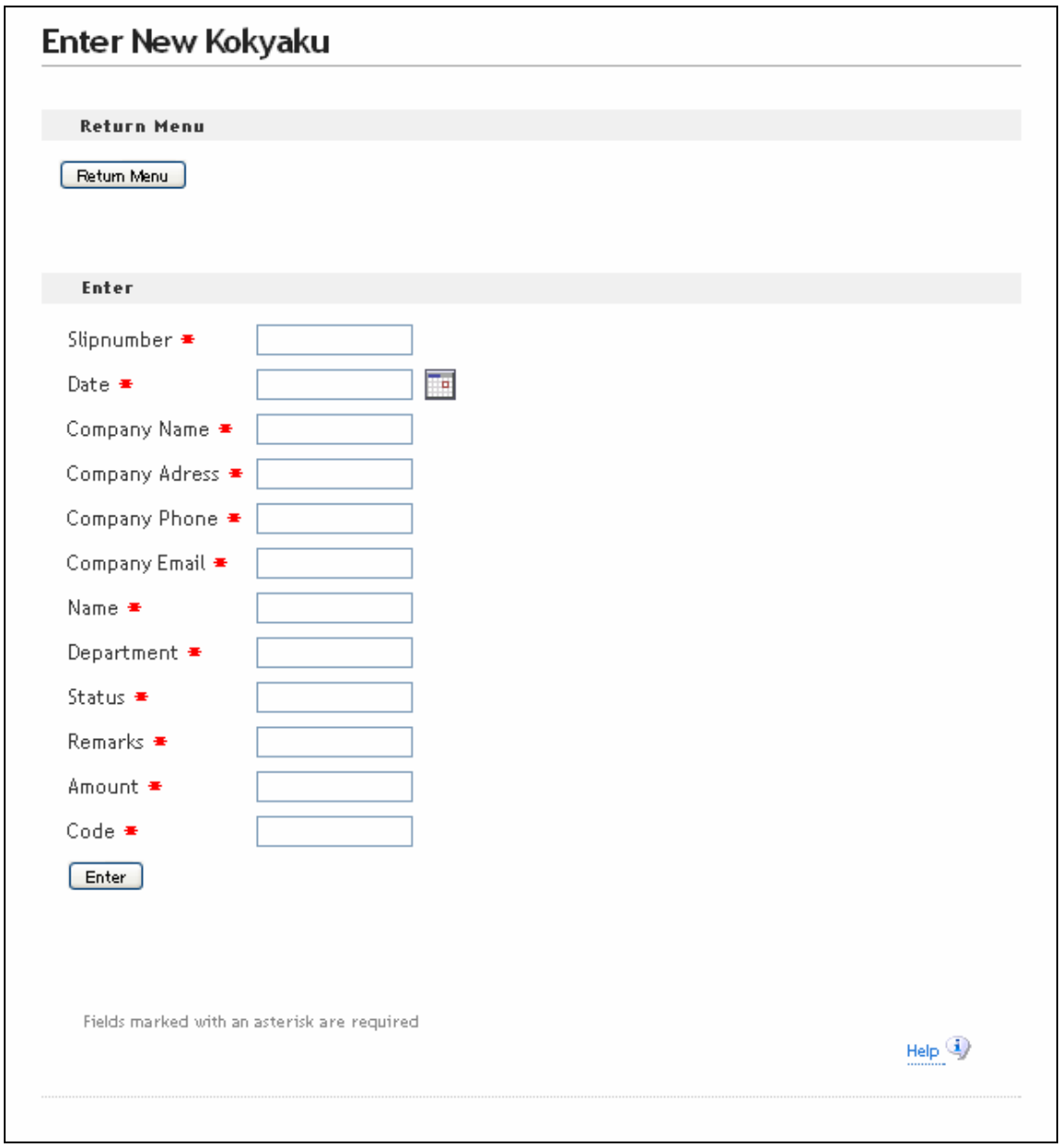

図 5-7

図 5-7 は、顧客生成画面である。ここで、顧客の作成を行う。すべての項目を入力し、「Enter」ボタンを押すとデータベースへ書き込まれる。ここでは、すべての項目を入力しなければエラーとなる。また、「Slipnumber」は重複を許さないので、すでに存在している番号を入力するとエラーとなる。「Return Menu」ボタンを押すとメイン画面へ戻る。

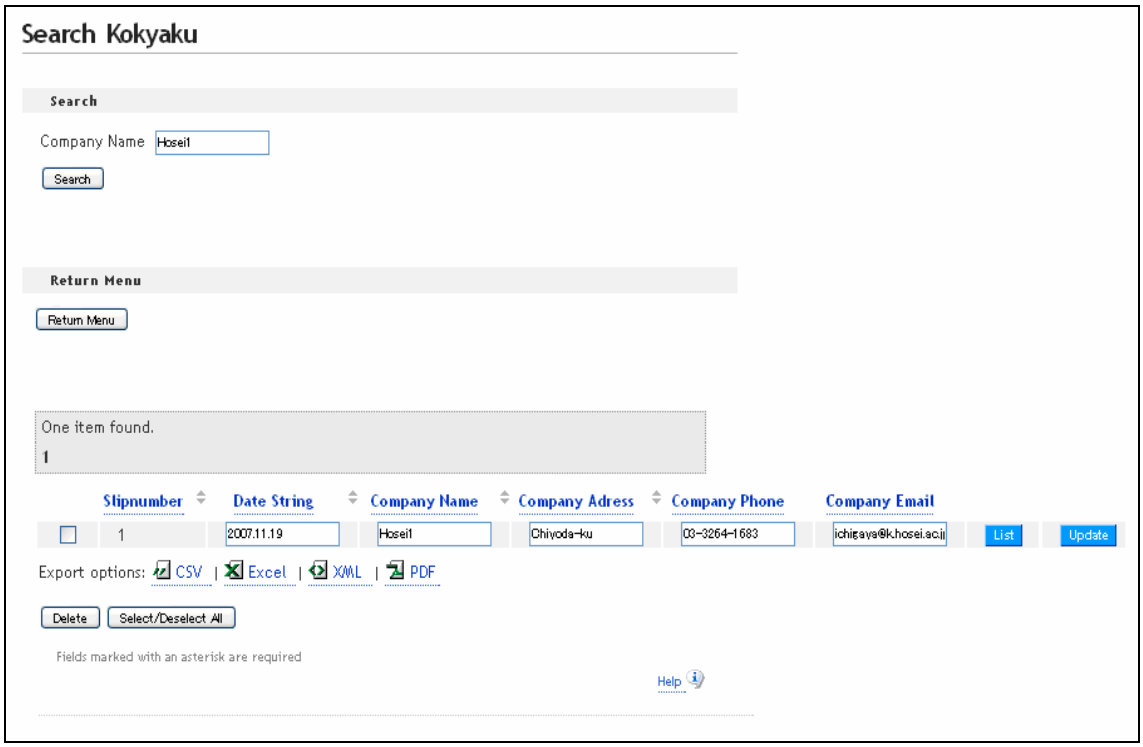

図 5-8

図 5-8 は、顧客検索画面である。「Company Name」を入力し、「Search」ボタンを押すと検索結果が表示される。「Return Menu」ボタンを押すと、メイン画面に戻る。検索結果のチェックボタンをチェックし、「Delete」ボタンを押すと削除する。検索結果の値を変更し、「Update」ボタンを押すと更新される。ただし、Slipnumber の変更はできないようになっている。「List」ボタンを押すと、図 5-10 に遷移する。

図 5-9 は、顧客確認画面である。「Get All」ボタンを押すと、データベースに書き込まれている値がすべて表示される。「Return Menu」ボタンを押すと、メイン画面に戻る。

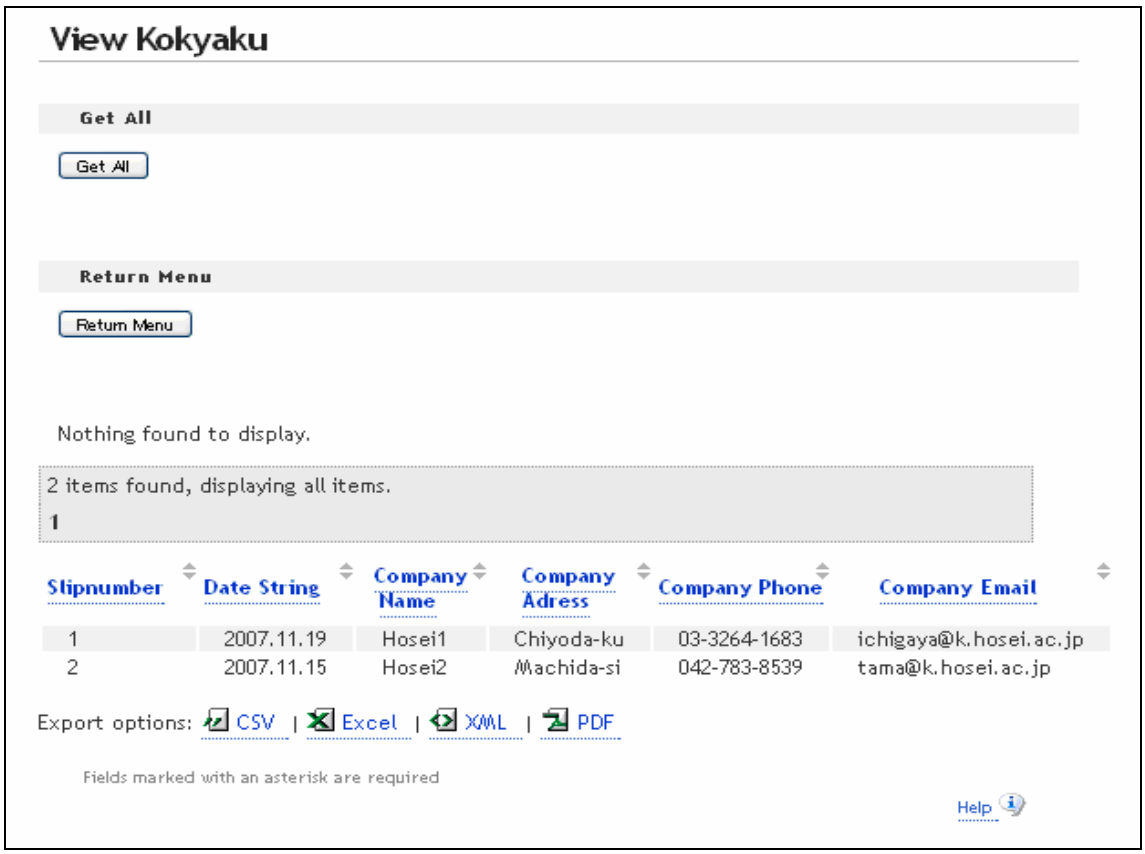

図 5-9

図 5-10 の画面で、「Personal」ボタンを押すと、個人情報編集画面（図 5-11）へ遷移する。「Detail」ボタンを押すと、詳細編集画面へ遷移する。「Return」ボタンを押すと顧客検索画面へ戻る。

図 5-11 は、個人情報編集画面である。値を変更し、「Update」ボタンを押すと更新される。「Return」ボタンを押すと図 5-10 へ戻る。

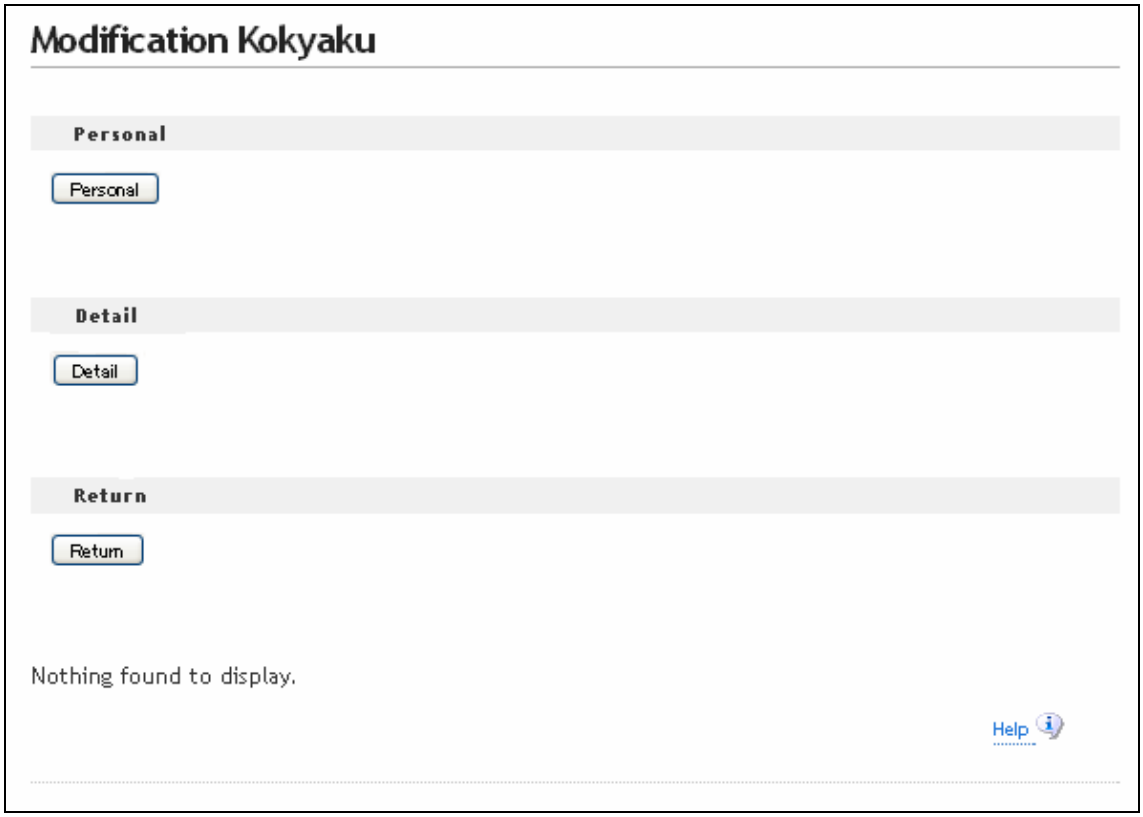
Modification Kokyaku
Personal
Personal
Detail
Detail
Return
Return
Nothing found to display.
Help

図 5-10

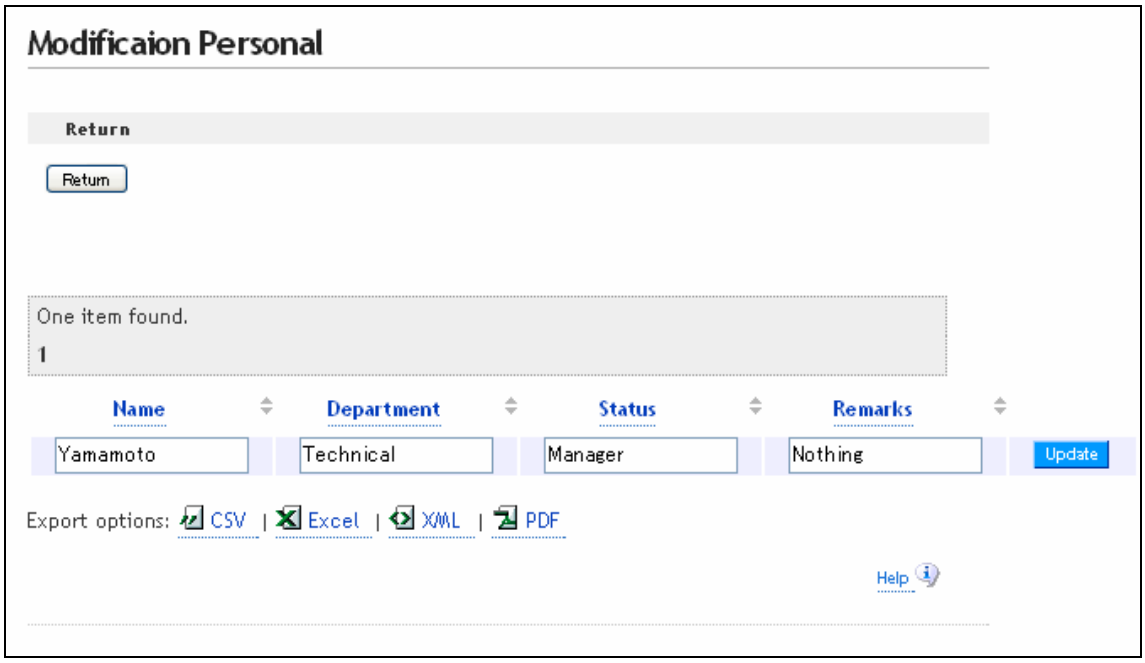
Modificaion Personal
Return
Return
One item found.
1
Name
Department
Status
Remarks
Yamamoto
Technical
Manager
Nothing
Update
Export options: CSV | Excel | XML | PDF
Help

図 5-11

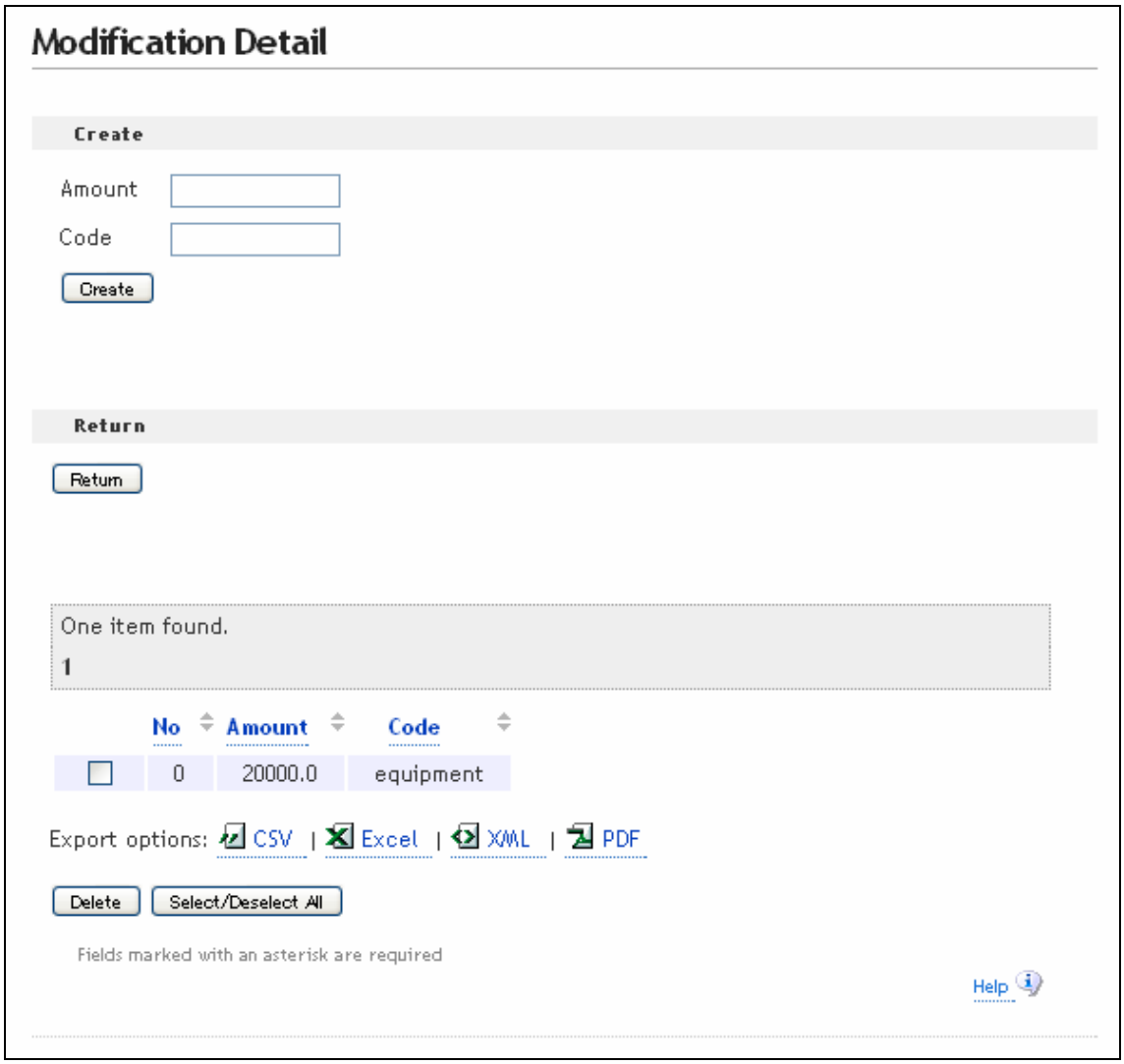

図 5-12

図 5-12 は、詳細編集画面である。新たに取引が行われたとき、「取引科目」と「金額」を入力し、「Create」ボタンを押すとデータベースに書き込まれる。また、「No」は、取引の数を表し、それらは生成時に自動で付けられる。チェックボタンをチェックし、「Delete」ボタンを押すと削除される。

次に、ログイン機能を追加する。

ここで、name、code、department、status、password の 5 つの要素を持つ Staff データベースを作成する。管理者はこれを管理し、メンバー登録された社員は、code と password を入力して顧客管理システムにログインする。図 5-13 にログイン機能のクラス図を図 5-14 に状態遷移図を示す。

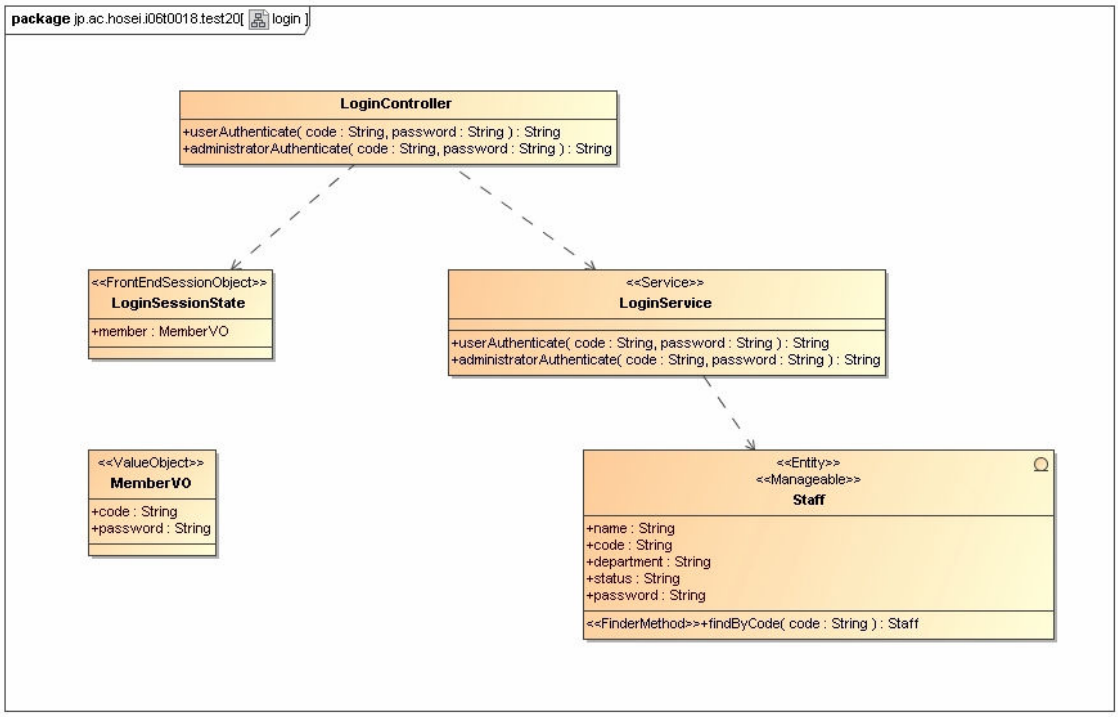

図 5-13

- LoginSessionState
  ステレオタイプ<<FrontEndSessionObject>>を加える。
  ページをまたいで同じデータを保持する。

- MemberVO
  ステレオタイプ<<ValueObject>>を加える。
  Controller と Service 間でやり取りする値を格納する。

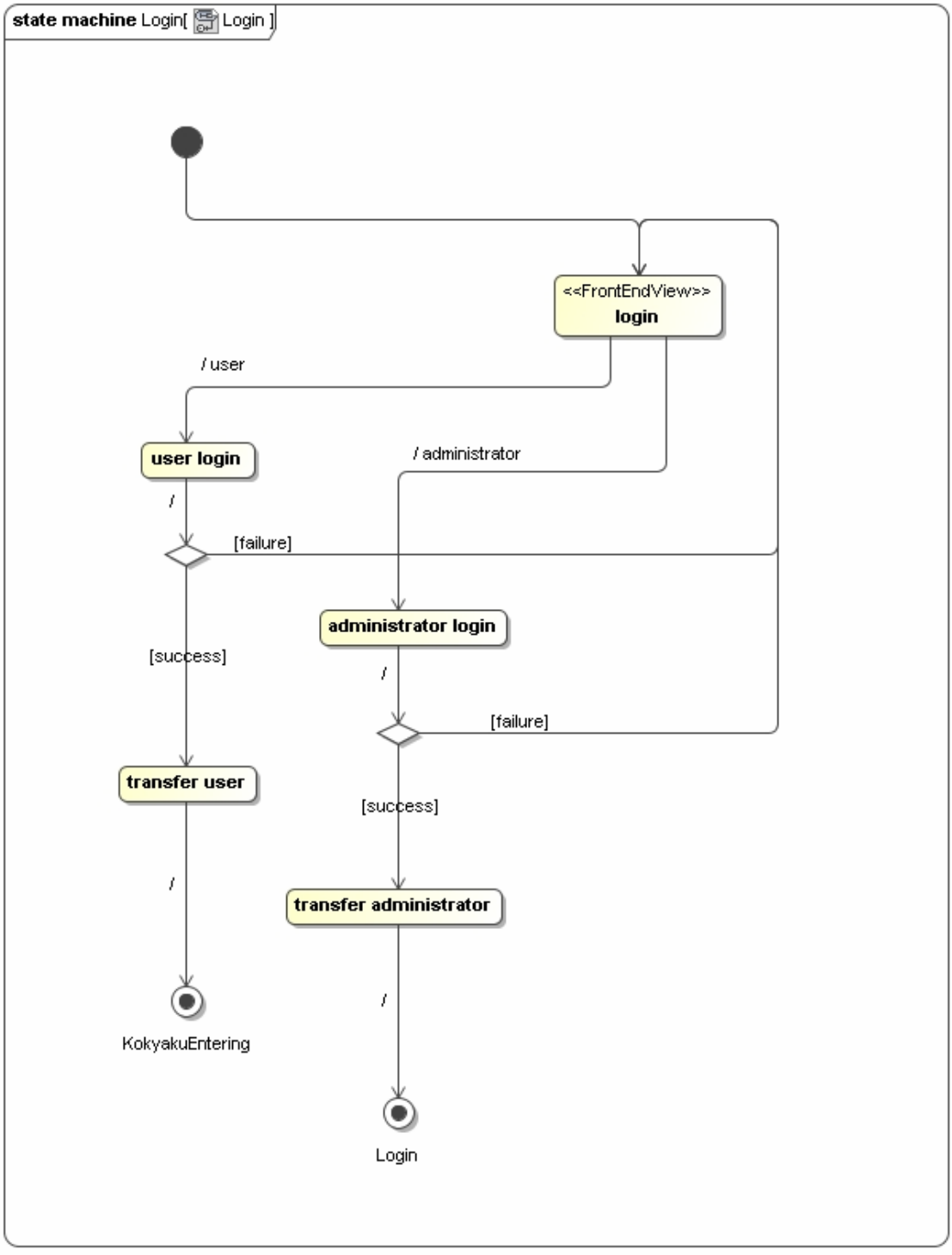
state machine Login[Login]
<<FrontEndView>>
login
/ user
/ administrator
user login
/
[failure]
[success]
administrator login
/
[failure]
transfer user
[success]
/
transfer administrator
/
KokyakuEntering
Login

図 5-14

以下に、ログイン機能の実行画面を示す。

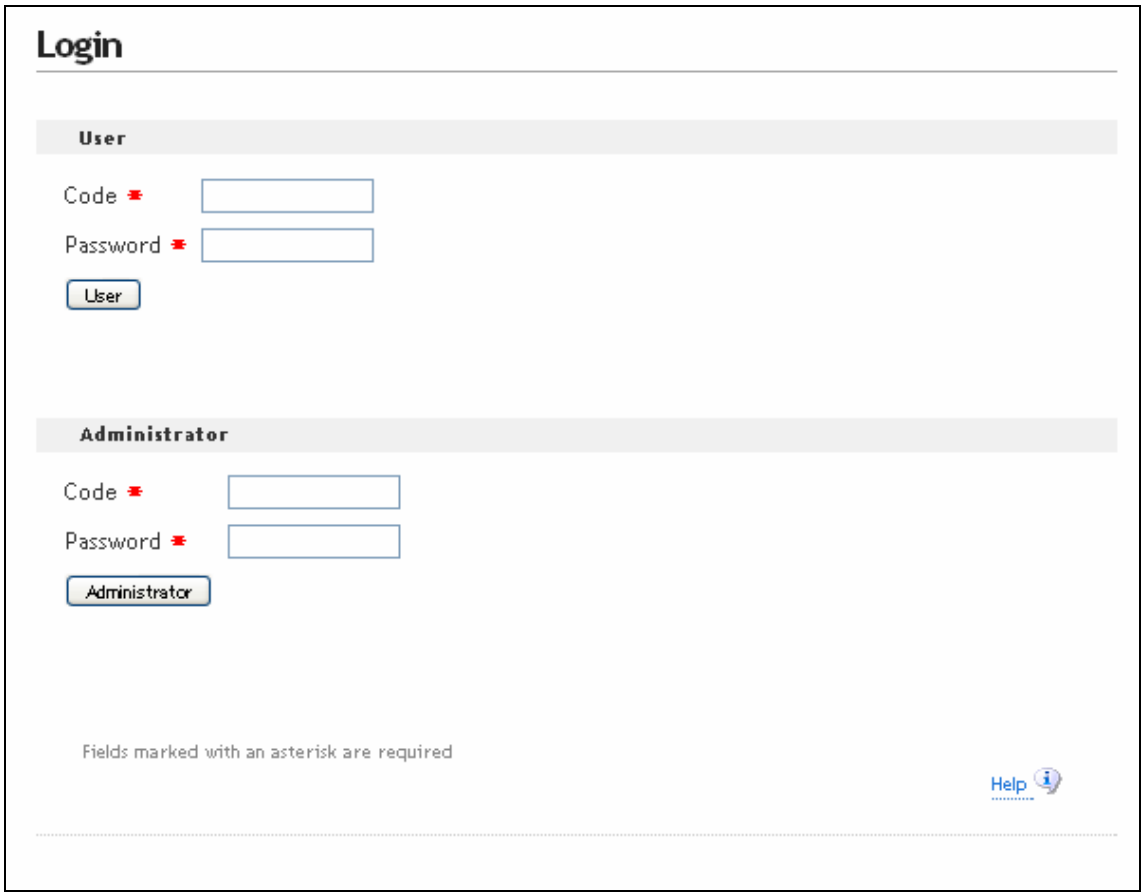

図 5-15

更新者は、コードとパスワードを入力し、「User」を押すと顧客管理システムのメイン画面（図 5-6）へ遷移する。また、管理者の管理画面としては、第 4 章の可視レベルで示した自動生成画面を使用する。

第６章　考察

UML 図から生成された java のファイル数を表 6-1 に示す。自動化率は、約 93.6%だった。自動生成されたファイルについては、デッバグの必要はなかった。半自動されたファイルは、ビジネスロジックの部分である。この部分においては、プログラムの記述が必要である。ここで、プログラムの記述間違いによるバグが発生したため、デバッグが必要だった。

表 6-1　UML 図から生成されたファイル数(java ファイル)

	自動	半自動	合計
WEB	94	3	97
EJB	61	9	70
共通	21	0	21
合計	176	12	188

また、自動生成された JSP のファイル数を表 6-2 に示す。

表 6-2　UML 図から生成されたファイル数(JSP ファイル)

	JSP
WEB	154
EJB	0
共通	0
合計	154

第 7 章　まとめ

本研究では、ソフトウェア開発手法として、IMAH と MDA を用いる方法で実験結果を示した。

それぞれの段階で、ビジネスロジックの記述におけるバグの発生のみで順調に開発することができた。IMAH を用いることで、モデリング段階での情報の整理が正しく行えるようになる。正しいモデリングが行われれば、AndroMDA により信頼性の高いコードを多くの部分で自動生成することができ、バグの発生を減らすことができる。

顧客管理システムの開発を通して、限られた時間一人で開発を進めたことを考慮すると、本研究で提案した開発手法は有効であるといえるだろう。

今後の課題として、ビジネスロジックのプログラム記述におけるバグの発生についてであるが、AndroMDA はオープンソースで開発されており、その開発ペースは非常に迅速である。将来的には、ビジネスロジックにおいても自動生成することができるようになり、プログラムの誤記によるバグを削減することができるだろう。それによって、より大規模なソフトウェアの開発が容易になり、今後はより多くの分野で使われるようになるだろう。

謝辞

この“線形モジュール追加型抽象階層およびモデル駆動型アーキテクチャを用いた顧客管理システムの開発”について研究するにあたり、多くの方々の協力とアドバイスを賜りました。研究全般においてご指導いただいた法政大学情報科学研究科の大森健児教授に深く感謝します。また、手助けしてくださったみなさま、アドバイスを下さったみなさま、心より厚く感謝いたします。

文献

[1] Ohmori K, Kunii T L, “Development of an Accounting System”, 9th International Conference on Enterprise Information System (ICEIS2007), Madeira, Portugal (June 2007) pp. 437-444.

[2] Ohmori K, Kunii T L, “An Incrementally Modular Abstraction Hierarchy for Linear Software DevelopmentMethodology”, CW 2006. IEEE Computer Society 2006, pp. 216-233.

[3] Kunii T L, Ohmori K, “Cyberworlds: Architecture and Modeling by an Incrementally Modular Abstraction Hierarchy”, The Visual Computer, Springer-Verlag, 2006, 22(12): pp. 949-964.

[4] Ohmori K, Kunii T L, “The Mathematical Structure of Cyberworlds”, CW 2007. IEEE Computer Society, Hannover (Oct. 2007), pp100-107.

[5] W. Li and K.Ohomori, “Hierarchical Visualization Of 3-Dimensional Objects Using, Cellular Structured Spaces”, Workshop on ITS and Image Processing, Sapporo Japan (2002) (in Japanese)

[6] Yuichi Okuno, “Online Accounting System The Development Using Incrementally Modular Abstraction Hierarchy and Model Driven Architecture”, in *Graduation Thesis 2007*, Faculty of Computer And Information Sciences, Hosei University, pp. 11-14.

[7] http://www.andromda.org/