# OPAS Porject のなりたち

青山希, 岸健司, 明石敬, 小林敦

2005 年 4 月 4 日

# 目次

# 第 1 章

# はじめに

(文責:青山　希)

## 1.1　背景

慶応大学大岩研究室では、これまでオブジェクト指向プログラミングの授業や、社会人向けの研修など多くの場でプログラミングの導入教育を実施してきた。それらのカリキュラムの中で常に主眼を置いてきてきたことに「わかりやすいプログラムを書く」ということがあげられる。プログラムはただ動くだけではなく、人にも自分にもわかりやすいものでなければ、保守管理や機能追加をすることができず、結果として役に立たないプログラムとなってしまう。

しかし、初学者や、長く我流でプログラムを書いてきた学習者にはわかりやすさが重要であるということを理解するのは難しい。その一番の理由は今まで人が書いたプログラムを読むという経験がほとんどなかったためである。そこでカリキュラムではペアでプログラミングをするなどのアクティビティを取り入れ、人のソースコードを読む機会を与えている。しかし、より多くの人が書いたソースコードを読み、わかりやすさの重要性を実感するためには、もっと多くの人のソースコードを読みそれについて議論できる環境が必要である。

## 1.2　OPAS とは

背景で述べたような問題を解決するために、我々はプログラミング教育課題提出システム「OPAS(Open Programming Assignment System)」を開発した。OPAS はプログラミング教育のカリキュラムで出される課題を快適に提出し、それを受講者全員でお互いに見合うことができるシステムである。

このようなシステムを導入することで、他人のソースコードを読み、それに対してレビューをするような授業を行うことができる。それにより、わかりやすいプログラムを書く意義を実感し、能力を伸ばすことができる。また、OPAS は課題提出の環境としても有用なものであり、講師の課題回収の労力も軽減することができる。

# 第 2 章

# 分析

## 2.1　要求の分析

**(文責:岸　健司)**

このプロジェクトの企画は揺れ動き、仕様はなかなか定まらなかった。「プログラミング授業課題提出採点システム (オブジェクトプログラミング用)」プロジェクト。これが、当初、我々が配属されたプロジェクトだった。

ところが、オブジェクトプログラミングの過去のワークフローを把握する中、課題関連以外にも問題が存在することに気づき、オブジェクトプログラミング授業準備のワークフロー全体の改善に企画は拡大された。またその後、ワークフロー改善は見送られ、より世の中の役に立つ物にしようと、オブジェクトプログラミングへの束縛がはずされた。「プログラミング教育での課題提回収採点システム開発」が扱う対象になった。さらに、実際の今期の開発は、の関係上、「プログラミング授業の課題回収**閲覧**システム開発」へ限定された。採点は見送られ、「受講生同士で課題の公開された解答を見合う」特徴などが足された。

この節ではこれらの企画変遷と仕様の定義ができるまでの様子を書く。

仕様自体は、

- シナリオ
- ユースケース図
- 画面遷移（主要画面のイメージ及び画面間の遷移）

によって、定義した。

### 2.1.1　ワークフローの改善提案

我々はまず、プロジェクトマネージャ (以下 PM) 青山の意見をふまえ、

- オブジェクトプログラミングのスタッフのワークフローを考える
- それを支援するアプリケーションを作る

という方針で進めることにした。PM 青山はオブジェクトプログラミングのティーチングアシスタント (以下 TA) でもあり、また、メンバー明石も、スタッフを務めたことがあることがわかった。この 2 人に、過去の授業における毎週の授業スタッフ仕事の流れをインタビューした。そして、これを リスト 2.1 のようにまとめた。

リスト 2.1: 現状のワークフロー

```
月曜日
  課題提出者リストへのリンクをはる。
  資料を Web にアップ
  課題提出メーリングリストのチェック
  告知とかあればする。
月-水
  大岩先生にテキスト（PPT）を見せて議論
水曜
  先生の指摘を受けて、テキスト（PPT）の修正
  資料が変わったら更新。
木曜
  ☆授業→課題が出る

  メール (課題提出リンクつき) が objprog0x-kadaixx@crew.sfc.keio.ac.jp に届く
  木曜?月曜にかけて SA チームに登録作業をしてもらう
  次の次の週までに採点を行い、メイン TA に採点講評を伝える

最終課題
  メール (課題提出リンクつき) が objprog0x-final@crew.sfc.keio.ac.jp に届く

  プログラム点、レポート点それぞれを採点し、合計して 100 点満点とかできめる。

評価
  大岩先生に採点結果を Excel で渡す

□登場人物
大岩先生
メイン TA:明石
サブ TA(CreW):りんたろう
サブ TA(CreW):青山
サブ SA:小林　敦
サブ SA:小林　明日香
サブ SA:有田

＊このシナリオはフィクションです。
実在の人物とは関係ありません。
```

我々は、このワークフローをもとに、 システムを用いて改善されたシナリオを考えてみた。次のミーティングまでに用意した各メンバーのシナリオ案は次のとおりである。

リスト 2.2: シナリオ明石案

シナリオ明石案

シナリオ
口登場人物
大岩先生
メイン TA:明石
サブ TA(CreW) :りんたろう
サブ TA(CreW) :書山
サブ SA:小林　救
サブ SA:小林　明日香
サブ SA:有田

*このシナリオはフィクションです
実在の人物とは関係ありません

メイン TA 明石は日曜までに捷出された課題の捷出者リストのリンクをはり
次回授業分の資料を Web にアップした
メイン TA 明石は大岩先生にテキスト (PPT) を見てもらい議論をし
修正すべき点があったので修正を加え修正した資料を web にアップした
メイン TA 明石は TA SA チームに資料の修正があった旨を伝えた
授業:大岩先生が授業を行い TA SA はがんばります
生徒がメールで課題を提出し届いたメールから提出課題を SA チームが採点システムに登録した
TA チームは採点システムで各生徒の課題を見て採点を行いメイン TA 明石に採点講評を伝えた

最終堺題:
生徒がメールで最終課題を提出し届いたメールから最終課題を SA チームが採点システムに登録した
TA チームは採点システムで覚醒との最終課題を見てプログラム点レポート点それぞれ採点を行い
各生徒の点数を採点システムに登録した
大岩先生は採点システムに登録された点数をもとに生徒たちの成績をつけた

リスト 2.3: シナリオ岸

シナリオ
[Web で URL 登録版]

授業前 :
SA 小林君はシステムが稼働していることを確認し、
教材 Web にシステムへのリンクを貼る。
第 1 回目の課題をシステムに追加 (※) する。

※課題名、締め切りなどを入力。管理パスワードを用いる。

第 1 回目の授業 :
TA 明石君はプログラミング課題提出は課題用の Web を作くり、
システムにそのページの URL を登録することを伝える。
課題用の Web にはテンプレート (※) があることを伝える。
ただし、システムはユーザ登録が必要であることも伝える。

※) java ファイルへのリンクや感想がまとめられている (XML が良い？)

第１回目の授業終了後：
生徒岸君は第１回のプログラミング課題（商品種類を管理するプログラム）を作成する。
テンプレートを用いて、課題を Web ページにし、
ちゃんと表示されることを確認する。

教材 Web からリンクをたどりシステムの説明を読む。
システムにユーザ登録する（※ 1）。
システムにログイン（※ 2）する。
課題の Web ページ（※ 3）の URL を登録する。
回収された Web ページをシステムに示され、
ちゃんと回収されたことを確認する。

※ 1）学籍番号，氏名，学年，ログイン名，メールアドレス等の必要を入力する。（履修者名簿より早く…）
　１度システムがメールを返して、パスワードを得る。
　パスワード紛失・ログイン名重複（いたずら）には SA が対処し、
　システムの登録データを変更する。
　ユーザ登録は学期で１回のみ。

※ 3）java ファイルへのリンクや感想がまとめられている XML を想定。
　　この時点でシステムは関連ファイルをすべて回収する。
　　重複登録は最新版いがいも回収物を一応取っておく…

第１回課題の締め切りを 3 日すぎた日
TA りんたろう君は
システムに管理者としてログインする
第１回の課題を採点する（※）。
次の日もシステムに管理者としてログインし、第１回の課題を採点する。
その３日後
第１回目課題の採点を終える。

※）採点対象の課題と遅刻提出かどうかなどを見ながら、得点を入力する。
フレームで区切られた採点をコントロール（次の人，点数入力欄など）する領域の存在を想定。
インタフェースは Java スクリプトなどでキーボードにも対応できると…

第 3 週の授業-1 日：
TA りんたろう君はメイン TA 明石君に採点講評を伝える。
SA 小林君はシステムに第１回課題を受付終了させる。

第 3 週の授業：
TA は授業で第１回目課題の講評を生徒に伝える。

第 3 週の授業後：
TA りんたろう君はシステムの第１回課題の採点リストをチェックする。
未採点の課題が登録されていいるに気づき、採点をする。

同様に第 n 回の課題も処理。

学期末：
SA 小林君はシステムから採点表を出力（※）し、大岩先生に渡す。

※）出力は CSV を予定。

---

[E-MAIL で URL 登録版]
e-mail で web のアドレスを送る版も考えられる。メールとして参照先が残るメリットもある。
しかし、作りづらそう。

リスト 2.4: シナリオ小林案

シナリオ小林案
オブジェクトプログラミングを履修している学生 A くんは. 木曜日の授業に出た
その日の課題は, メソッドを使うことの利点を答える記述問題と,
これまで作ったプログラムをメソッドを利用して書き直すというものだった
A くんは授業中にこの課題をこなし, 記述問題を txt 形式で, プログラム問題を
Exercise. Java の形式で作成した
それから. 授業サイトから課題提出フォームへ行き, ログイン名を入れて, 作成した
課題の答えをテキストボックスに貼り付け, 更に授業の感想を書き, 提出ボタンを押した
こうして提出された課児を元に. 決められたフォーマットの html に組み込まれて, html として
CreW のサーバに保存されるそしてそれぞれの html へリンクが作成され, -つの html にまとめられる
また提出した学生のログイン名と提出時間は CSV として保存される
TA は, 次の集の月曜から. 作成された CSV を元に課題の採点をしていく
CSV には捷出された時間が妃入されているため, 遅延かどうかはすぐに判断でき
またプログラムによって作成された課題 html は. それぞれの html をまとめた html から
クリックで飛ぶことができるため. 採点してから次の課題を参照するまでの時間も短縮される
こうして提出者一覧の CSV は, 採点結果一覧のリストとなり, 別名で保存される.

これらのシナリオを読み比べ、まとめたワークフローを作成したものが、リスト 2.5 である。

リスト 2.5: ワークフロー分析

■初回の授業のフロー
受講者：ユーザ登録
TA：課題登録システムの説明
■ 1 週間のフロー
月曜日
明石 (TA)、先生：今週の資料について議論
明石：資料を Web にアップ、opas に課題を登録する
武田 (TA)、青山 (TA：先週の課題の採点
火曜日
水曜日
明石：先生の指摘を受けて教材の修正
木曜日
☆授業
SA 小林：opas で課題提出開始
受講者：課題作成
受講者：課題提出
(感想、ソース　複数問題、複数ソース（必要なものだけ）)
＊課題受付中止
適切なタイミングで SA がやる
■最終レポートのフロー
毎年違うだろうから出力した Excel に足す

### 2.1.2　企画書

ここまでで作成したシステムを用いたワークフロー改善案をもとに、 企画書を作成することとなった。

企画書を考えるこの段階での企画目的は、「オブジェクトプログラミングのスタッフが行う授業準備全体のワークフローの改善」であった。

特に、課題の収集・採点の作業は、受講生とスタッフの両方に負担が大きかったため、サポートシステムを作ろうと言うことになっていた。

この目的に対しすぐには賛同しなかった岸は、 「今回のシステムをきちんと動かすのは難しいので、負担は少ない方がよい。ワークフローも課題関連に絞って、システムに専念した方がよい」という意見を出した。 しかし、

- ワークフロー自体の作成はそんなに大きな負担ではない
- 課題以外にも授業準備に問題点 (予習をしなかったり、質問に答えないスタッフ) があり、それらを来年は防ぎたい

などの意見が出た結果、ワークフロー全体の改善をやはり主眼に置くことになった。そして、企画の目的として、「オブジェクトプログラミングの運営を円滑にし、授業の質を向上させる」となった。

実際の企画書の作成は、 ミーティングで、記載内容の大枠を決め、レビューを繰り返すことで作成した。。図 2.1〜図 2.4 に発表で使った企画書を示す。

研究会では次の質疑応答があった。

- 質問:授業の質問は今までのようにメールで受け付けるのか？
  - 反応：メールを考えてる
  - 反応：プロジェクトのスケジュール上課題に関する問題解決が優先
  - 反応：掲示板での解決方法も挙がっているがうまい方法とは言えない
  - 反応：FAQ があったら良いかもしれない
  - 反応：質問のメールの対応はしっかりしなくては。対応が遅れてたら管理者が知る機能がいるかもね
- 質問:課題の講評や感想に対する返答はどうするのか？-反応：これから検討
- 質問:解答の遅延提出は受け付けるか？-反応：(先生の意見) 受け付けない

企画の内容に未定事項が多かったが、研究会で、この企画は次へ進む了承を得ることができた。

# OPAS 企画書

~Object Programming Assignment System~

2003.10.16

## 1 目的

オブジェクトプログラミングの運営を円滑にし、授業の質を向上させる。

## 2 背景(現状の問題点)

2003 年度のオブジェクトプログラミングの授業に関して次の問題がある。

● TA について

仕事が遅いことがあった。教材を先生に見せることが直前になってしまったり、プログラミング課題の採点がなかなか終わらない。

● SA について

仕事をしっかりとしていないことがあった。授業の予習をしていなかったり、受講者からの質問メールにほとんど返信しないことがある。

●プログラミング課題について

・採点の準備(提出課題と提出者と採点欄との対応表づくり等)が SA の負担になっている。
・採点の際の課題閲覧と点数入力の作業にストレスを感じる(閲覧と点数入力の切り替え操作など)。
・受講者は課題用の Web ページを用意する必要があった。
・受講者から提出された課題を保管していない。

我々は、これらの問題の原因は「SA/TA の一連の仕事の流れが定まっていない」「課題に関わる作業の負担が多い」にあると考えた。

## 3 概要

[背景]で指摘した問題を解決するために、次のことをする。

●授業準備、授業、課題提出/採点を円滑に行うためのワークフローを提案する
●課題の提出/採点のワークフローをサポートをするシステムをつくる

対象(影響を与える人々): オブジェクトプログラミングTA・SA・受講者



図 2.1: 企画書 1 頁

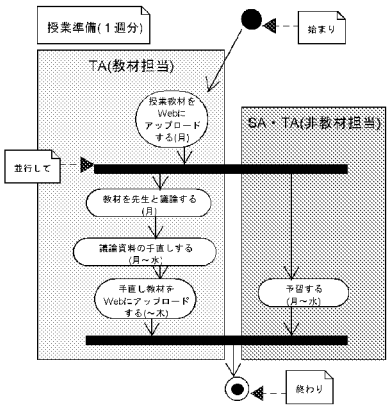
## 4 内容

ワークフローの提案をする。また課題の提出/採点サポートシステムの制作をする。

### 4.1 ワークフローの提案

以下を原型としたワークフローを提案する。「授業準備」「授業外質問」「課題提出採点」の3つについて示す。これらは現状案であり、今後の検討で改善の余地があれば改善していく。

★「授業準備」のワークフロー↓




★「授業外質問」のワークフロー↓

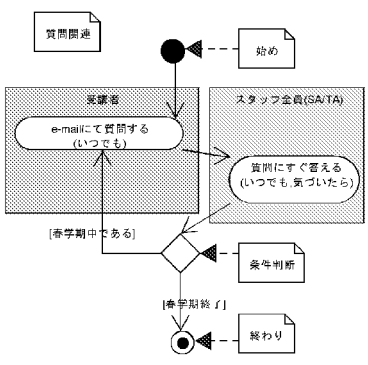





図 2.2: 企画書 2 頁

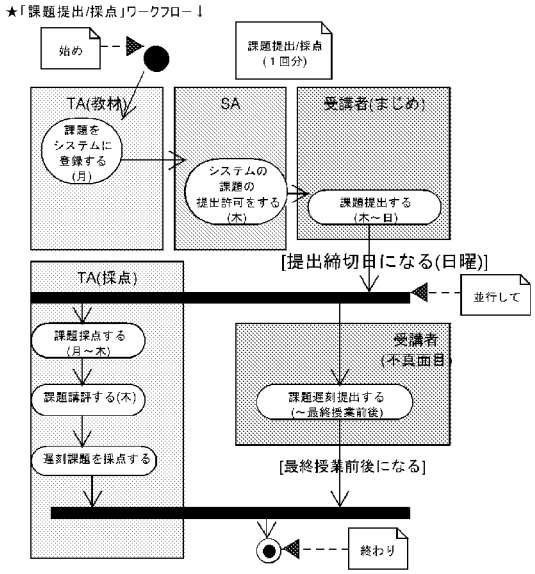



図 2.3: 企画書 3 頁

### 4.2　課題提出/採点サポートシステムの制作

オブジェクトプログラミングのプログラミング課題について、提案する「課題提出/採点」ワークフローを前提とし、以下の特徴をもつ Web アプリケーションを制作する。

●採点準備の労力をほとんどなくす
…システムが採点用の課題閲覧ページを作成してくれる。従来のように SA がコピー&ペーストを繰り返して採点用の表を作成しなくてよくなる。

●提出労力を低減する
… Web アプリケーションを通じて、受講者が課題を直接提出できる。従来のように受講者は Web ページを作らなくてよい。

●採点作業のストレスを軽減する
…従来の採点では課題閲覧と点数入力との移行に、ウィンドウの切り替え作業が必要であった。これを不要にする。

●全採点結果の表を Excel 処理が可能なファイルとして出力する
… 2003 年度では各回の採点結果の手作業マージを余儀なくされた。これを不要とする。

●課題を保管する
…受講者の提出課題の保管を特別な作業無しで提供する。

## 5　実施体制

### 5.1　メンバー

政策メディア研究科修士1年　青山　希(Project Manager)
環境情報学部3年　明石　敬
環境情報学部4年　岸　健司
環境情報学部3年　小林　敦

### 5.2　スケジュール

| 日付(～まで) | | システム関連 | ワークフロー関連 |
|---|---|---|---|
| 第4回 | 10月16日 | 企画決定 | |
| 第5回 | 10月23日 | ユースケース作成 | ワークフロー検討 |
| 第6回 | 10月30日 | 分析・設計 | |
| 第7回 | 11月6日 | 実装 | |
| 第8回 | 11月13日 | CUI完成 | |
| 第9回 | 11月27日 | GUI作成 | |
| 第10回 | 12月4日 | GUI作成 | |
| 第11回 | 12月11日 | GUI作成 | |
| 第12回 | 12月18日 | テスト・バグ取り | |
| 第13回 | 1月15日 | テスト・バグ取り | |

- 4/4 -

図 2.4: 企画書 4 頁

### 2.1.3 方針転換-プログラム教育環境の改善へ-

企画書段階では、オブジェクトプログラミングのワークフロー改善をすることで、運営を円滑にし、授業の質向上へつなげようという目的であった。ところが、その後の PM 同士の話し合いで、

- オブジェクトプログラミングについてでなく、プログラミング教育全般に焦点を当てた方がよい
- 授業運営と直接関わり合いのない、このプロジェクトに、授業のワークフローを決めて欲しくない
- 他人のソースコードを読む機会があるとよい。わかりやすいソースコードにもつながる

などの意見が出てきた。我々はこれに賛同し、プログラミング授業の課題の収集、採点を支援するシステム開発に企画を絞ることにした。絞ることで、単なるレポートシステムで終わらない、「新たな何か」をシステムに加えられるとも考えた。この時点で、課題関連のシステムには、様々な機能も期待がされていた。コンパイルチェック、提出プログラム実行機能、操作性の良い採点・閲覧、ダウンロード機能、印刷機能、アーカイブ機能、フォルダ提出機能など数多く挙げられる。我々は、最終的な特徴として、

- 公開閲覧…受講生同士も提出プログラムを見合える
- フォルダ指定提出…複数ファイルを一括で提出する

などの特徴を盛り込むことができた。

### 2.1.4 ユースケースとシナリオ

企画がほぼ固まった段階で、我々は、システムを用いた課題提出・採点のシナリオを考え、さらにシナリオから必要な機能を抽出しようと考えた。ここで、個人ベースで作成したシナリオを次に示す。

リスト 2.6: シナリオ岸案

```
シナリオ by 岸
[ユーザの登録]
初回の授業で、
TA 明石君は「課題提出のためにシステムにユーザ登録することが必要です。
履修する予定の人は次回までに登録してください。
Web ページからたどれます。」との説明をする。
あわせて登録の実演もしてくれた。

受講者岸くんは
さっそくユーザ登録を始めた。
```

システムにログイン名・学部・学年・氏名・メールアドレス (CNS のアドレスが指示されていた)・パスワードを入力すると
メールアドレスに確認メールを送った旨のメッセージを受け取った。
メールを受信するとユーザ登録完了するための URL が記載されていた。
その URL にアクセスすると、ユーザ登録の完了メッセージが表示された。

第２回の授業のとき、
TA 明石君はシステムに管理者としてアクセスし、暫定的な受講者リストを得た。

[課題の登録]
第２週の月曜日、
TA 明石君は、第 2 回課題をシステムに登録するため、管理者ログインする。
課題名 (第２回課題)、課題のフォーマット (感想、問題１のソース、問題２のソース)、
提出締め切り日 (次の日曜日 23:59) をシステムに伝える。

[課題の受付]
第２週の木曜日の授業開始直前、
SA 小林君は、システムに管理者ログインし、第２回課題を受付開始をシステムに伝える。
試しにテスト用のアカウントを用い、課題を提出し、問題がないことを確認する。
TA 明石君に受付を開始したことを伝える。

[課題の受付中止]
第２週の木曜日授業中、
TA 武田君は第１回課題の講評をする。
それを受けて SA 小林君は武田君から渡されていた模範解答を授業 Web に載せると共に、
システムの第１回課題を中止する。

[課題の提出]
第２週の授業中
TA 明石君は「今日の課題もシステムを使って提出してください。第２回課題が提出できるようにしてあります。」と受講者に伝える
受講者岸君は、授業中にに課題を終え、システムにログインする。
第２回課題の提出を選択し、感想、問題１のソースコード、問題２のソースコードをそれぞれシステムに伝え、提出する。
システムから受け取った内容とメールを送った旨が示さる。
岸君は提出証拠メールを受け取る。
メールは内容に提出課題内容を含んでいて、CC で obj-prog-04-assignment@crew.sfc.keio.ac.jp にも送られた。

[課題の提出状況の確認-受講者個人]
第２回課題を提出したか忘れた受講者岸君は、
システムにログインする。
すると、過去の提出履歴と採点履歴が表示される。
その中に第２回課題を期限内に提出したことが表示されていたことを確認する。

[採点者のコメントをみる]
第２回課題の課題の全体の講評を授業中に聞いた受講者岸君は、
自分の課題がどう見られたのか気になる。
システムにログインする。
すると、過去の提出履歴と採点履歴が表示される。
第２回の課題を選択すると、提出した課題の内容と採点者のコメント表示される。

コメントには「20 行の条件式は'<' ではなく'<=' を使わないと…」と書かれてた。
そのコメントに納得した岸君はそれ以後条件式の記述には気をつけるようになった。

[課題の提出・採点状況の確認-全体]
第２週の授業で
TA 明石君は第１回課題について講評する。
「今回の採点の分布と提出状況はこんな感じでした。
ほとんどの人が、〇〇の条件分岐のところがきちんとできていなかった…」
その際、課題の期限内提出者/遅刻者の人数やリストや採点分布の示されたページが示されたページが示された。
このページはシステムで自動生成されるもので、授業ページからリンクが張られている。

[閲覧・採点]
月曜日の朝、
TA 武田君はシステムに管理者ログインし、第２回課題を採点することをシステムに知らせる。
ログイン名、得点、提出日時、学年、学部、採点日時の一覧表が表示される。
まずは、提出された課題のレベルを把握するため、
学年で表を並び替え、無作為に５個ほど表から選び、課題を閲覧する。
そこから採点基準を決め、採点基準メモらんに、採点者名とともに入力する。
連続閲覧/採点することをシステムに伝える。
システムは先ほどの表の１番目にあった岸君の課題を表示する。
武田君は感想、問題１、２のソースコードを順に読む。
それぞれの移動にはキーボードのキーを用いた。
採点コメントに「20 行の条件式は'<' ではなく'<=' を使わないと…」と書く。
また、得点を４点と決めたので、「４」キーを押す。
システムは「この期限内提出者を４点に決め、次の提出課題に移って良いか」確認してくる。
リターンキーにより次の佐藤くんの課題に移り、同様に採点を続けた。
同様に３０人ほど採点を続けたところで、おなかが減り、採点を打ち切った。

注：課題閲覧
閲覧ページ：
提出日時に関する情報（期限内・何日遅れか）
課題の内容（感想・ソース）
提出者の情報
採点情報 (未採点/得点)
(採点基準メモ, 採点者コメント欄…別ウィンドウ？)
などが表示される
・よく使う機能にショートカットキーを用意

{ ソース閲覧補助機能 }
行番号
コンパイルチェック (ボタンによりコンパイル結果を表示)
ハイライト
メッソッドアウトライン
注目メソッドを強調
実行（クライアントの Java アプリケーションにクラスファイルをソケット通信で持ってくる）

注：最終課題採点
技術点、〇〇点…の複数配点にも耐えられるようにする

[採点つづき]

火曜日の朝、
TA 武田君は前日の採点の続きをするため、システムに管理者ログインする。
第２回課題を採点することをシステムに知らせる。
ログイン名、得点、提出日時、学年、学部、採点日時の一覧表が表示される。
採点日時にの欄に未採点と書かれたものが連続するように並んでいる。
未採点と書かれた中で一番上の所を選び、連続閲覧/採点を開始する。
採点基準を忘れかけていたので採点基準メモらんで確認し、
提出されているすべての課題を前日と同様に採点する。

[採点結果出力]
学期末、TA 明石君はシステムに管理者ログインして、
課題採点結果表を CSV 形式で出力した物をダウンロードする。
先生に成績をつける資料として提供する。

[アーカイブ]
2003 年秋
来年以降の参考資料として残すため、
TA 明石君は、システムに管理ログインし、アーカイブすることを伝える。
システムは提出課題や採点結果を閲覧できる静的な Web ページを構成するファイルを出力する
(html にすべてが埋め込まれているのではなく、データ { テキスト、java,xml} などとそのレイアウトを指定するファイル)
システムの置いてあるサーバにルート権限でログインし、システムの出力した Web ページのファイルをコピーし、適当な Web サーバに設置する。
また、印刷コマンドを実行することにより、提出課題と採点結果をすべて印刷する。

2005 年春
去年の提出された課題のできを確認し、教材の改良を加えるために、
去年の提出課題のファイルを参照したり、印刷物を閲覧したりする。

リスト 2.7: シナリオ小林案

シナリオ

受講者

オブジェクトプログラミングの受講者である A くんは，いつものように木曜 3，4 限にオブジェクトプログラミングの授業に出た。その日の課題は，授業で学んだメソッドを利用して，前回までに作った課題を改良することだった。
A くんは，授業中になんとか課題用の java ファイルを作成し，OPAS のトップページへ行きログインをした。そしてその回の課題提出ページへ行き，java ファイルを添付し，感想を書きアップロードをした。
翌週，授業の最初に TA 明石さんが課題の講評をしてくれた。
明石 :「今回の課題ではメソッドを利用するということでしたが・・・」
A :「ふむふむなるほど」
先週分の課題の講評を忘れないうちに聞けて，メソッドについてより理解の深まった A くんなのでした。

SA

SA 小林くんは，授業のある木曜日に OPAS を利用して課題の受付を許可した。OPAS のトップページへ行き，SA としてログインをし，課題を選択して受付許可をする。この課題は月曜日に事前に TA 明石さんによって登録されている。

こうして受付許可された課題は，その日の授業の間から日曜までに（受付自体は次週木曜まで）受講者によってどんどん登録されていく。
そして木曜日に次の課題受付を開始する際に，前回の課題受付を終了する。その時点で，提出していた学生のリストが公開され，OPAS を通じて学生も見ることができる。

TA

TA 青山さんは日曜になると課題の採点を始める。OPAS のトップページへ行き，TA としてログインをする。採点する回を選択すると，提出者の名前，ログイン名，提出時間，採点結果（この時点では空白，採点したかどうかも分かる）などがリストアップされる。
青山さんはリストの上から順に採点していくことにした。一番上にある受講者のログイン名をクリックすると，今回の課題の java ファイルの中身が iframe で表示されている html が開く。また，そのファイルがコンパイルを通るかどうかも表示されている。（コンパイル：〇などのように）
今回の課題はメソッドを利用することなので，新しく定義されたメソッドの部分の背景色が変わっていて，主にその部分を見れば採点できるようになっている。そしてその画面上で数字キーを押すと，採点が直接できるようになっている。採点が終わると，自動的にリストの次の受講者の課題へと飛び，引き続き採点ができるようになっている。こうして TA 青山さんはストレスを感じず，採点をすることができるのでした。

---

シナリオは、考えられる様々な機能を盛り込んだ物になっている。これらをもとに、現実的なものにまとめたユースケース案が図 2.5, 図 2.6 及びリスト 2.8 である。

リスト 2.8: シナリオ

---

*採点結果を確認する
ＴＡの青山さんは第 3 回の課題の採点を確認しようと思った。
ＯＰＡＳにログインしメニューから採点結果確認を選択し、第 3 回を選択した。
すると採点表が表示され、採点結果を確認することができた。

*授業を管理する
ＴＡの明石は 2004 年度のオブジェクトプログラミングでＯＰＡＳを使いたいと思った。
ＯＰＡＳにログインしＴＡ・ＳＡメニューから授業登録を選択した。
年度、学期、授業名を入力し送信ボタンを押すと「授業が追加されました」というメッセージが表示された。

*ＴＡの明石は登録した授業名が「オフジェクトプロクラミング」となっているのに気付いた。
ＯＰＡＳにログインしメニューから授業情報閲覧を選択し、授業情報編集ボタンを押した。
「オブジェクトプログラミング」と入力しなおして送信ボタンを押すと
「授業が編集されました」というメッセージが表示された。

*提出履歴を確認する
３回目の授業の際に A くんは前週に提出した課題がしっかり受け取られているかが気になったため OPAS を利用して確認することにした。OPAS にログインし提出履歴確認を選択すると提出履歴が表示され、その中に第二回があったため A くんは安心した。

*ユーザーを管理する
メイン TA の明石は受講者の A くんがだぶって登録されていることに気づいた。
OPAS にログインしユーザー管理を選択し明らかに使われていないアカウントのほうの A くんを削除した。

*解答を採点する
ＴＡの青山さんは第 6 回の解答を採点しようと思った。
ＯＰＡＳにログインしメニューから解答閲覧を選択し、

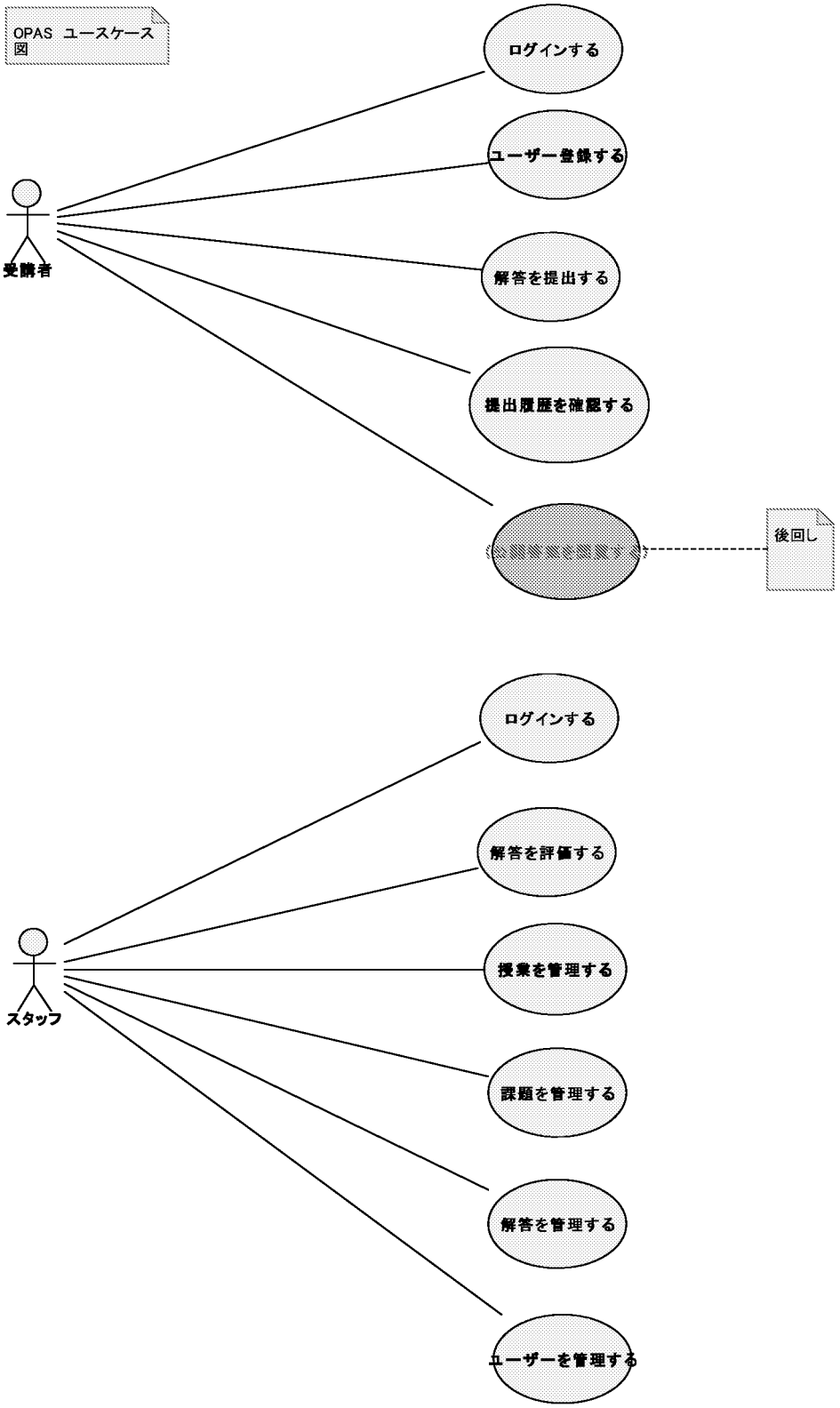



図 2.5: ユースケース図 1

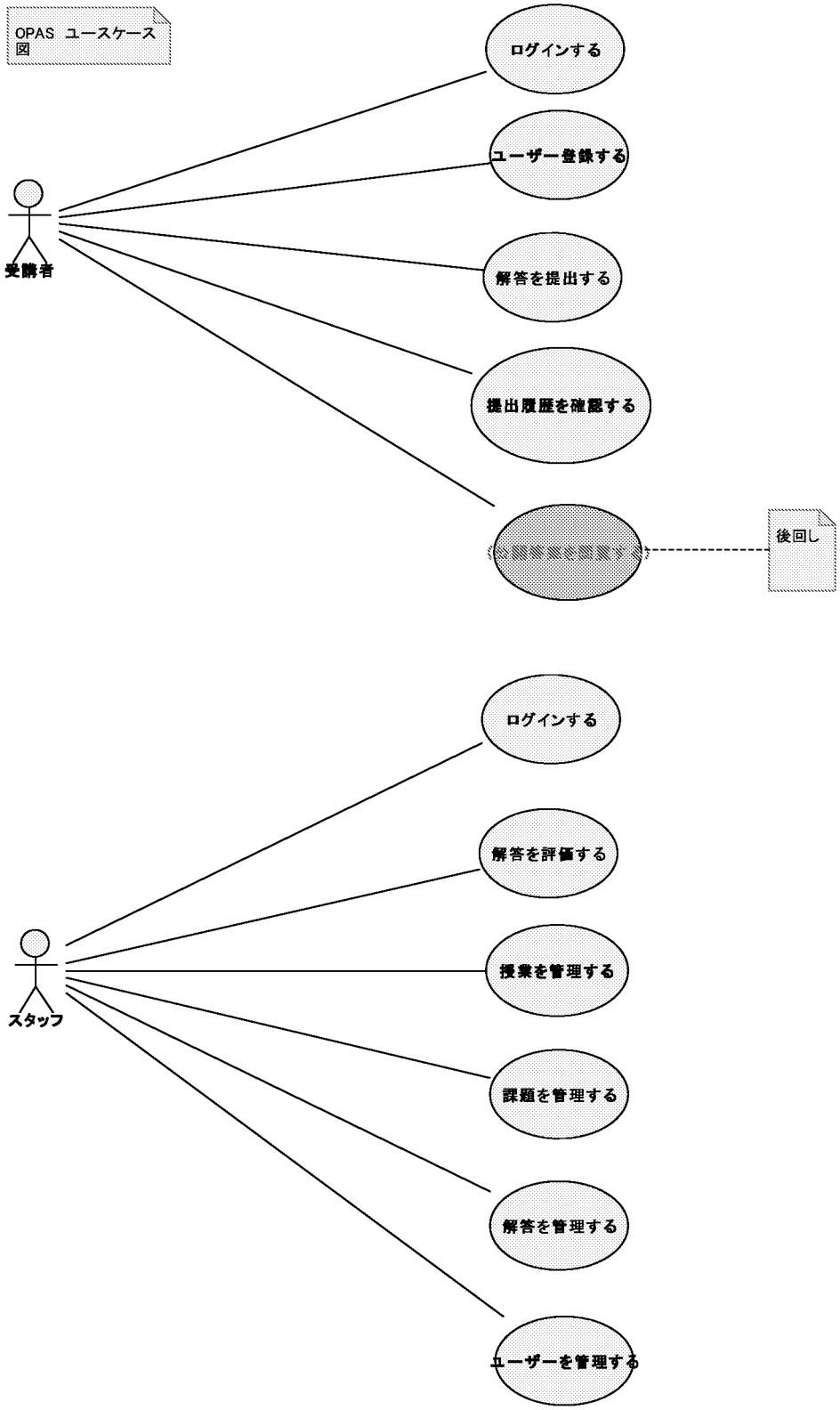



図 2.6: ユースケース図 2

第 6 回を選択し、採点ボタンを押した。
最初の提出者の解答が表示され、青山さんはアウトラインを用いて getHoge() メソッド
の中身を確認し、問題なかったのでラジオボタンで５を選択し、次へボタンを押した。
すると点数がつけられ、次の提出者の解答が表示された。

青山さんは間違って次へボタンを押したことに気付き、直前の解答へ戻ろうと思った。
戻るボタンを押すと直前の解答が表示され、４を選択し、次へボタンを押した。

全ての解答を採点し終わると「採点が終了しました」というメッセージが表示された。

*解答を提出する
ログインすると，メニューに課題提出の項目があり，選択すると第 1 回課題が受付中になっている。提出ページへ移動し，今回の課題ファイルを選択し，感想を書いてアップロードをした。

*解答を閲覧する
ＴＡの青山さんは受講者Ａ君の第 4 回の解答を見たいと思った。
ＯＰＡＳにログインし採点確認から第 4 回を選択した。
採点表が表示されたので、Ａ君の解答へのリンクをクリックすると
Ａ君の解答が表示された。

*解答ファイルを取得する
TA 青山さんは最終課題で提出されたアプリを実行してみたいと思い、解答ファイルを取得したいと思った。
ＯＰＡＳにログインしメニューから答案ファイル一括取得を選択した。
するとファイルが置いてあるフォルダが表示され、2003 年度のフォルダごとそのままダウンロードし、コンパイルをしてから実行してみた。

*課題を出題する
メインＴＡの明石は第二週目の課題を出題しようと思った。
まず、OPAS にログインしＴＡ・ＳＡメニューから課題の出題を選択した。
そして、課題のタイトル、課題内容、開始時間、期限を入力し、送信ボタンを押した。
すると課題の確認画面が表示された。ＯＫボタンを押すと課題が登録された。

*課題を締め切る
ＳＡの小林君は第二週の課題の受付を終了しようと思った。
まず、OPAS にログインしそしてＴＡ・ＳＡメニューから課題閲覧を選択し、
第二週の課題を選択し、受付終了ボタンを押した。
すると「第二回課題の受付が終了されました」というメッセージが表示された。

---

これらに対して、研究会では、

- 「解答を評価する」がわかりにくい
- ユースケースの粒度がバラバラである。アクタの協調関係を意識してユースケースを考えるとよい
- ユースケースを絞る必要があるのではないか

などの指摘をいただいた。

研究会での指摘と、並行して作成していた画面遷移（別節で説明）による機能詳細の変更を受け、最終的にユースケース図を図 2.7 のようにまとめた。ユースケースはまとまり

を意識しおかげで、開発の分担や範囲を決める単位としてユースケースが使われ、有用であった。

### 2.1.5 画面遷移

スタッフの採点及び受講生の提出関連の画面は、本システムの重要な部分であるため、メンバー各人の手書きの画面イメージ図を比較しあいながら作成した。シナリオを決め、各人が思い描く、理想の画面イメージをその場で書こんだ。そして、より良い画面を決定していった。このとき作成した、画面イメージ図の一部を図 2.8 から図 2.16 に示す。

受講生提出の画面づくりでは、松澤氏に協力頂いた。このことで、より具体的なシナリオを設定できた (具体的な課題問題設定や再提出、エラーシナリオ)。この作業により、提出１つをとっても、様々な場合があることがわかり、このプロジェクトの開発規模の大きさを我々は実感することになった。

残りの画面については、各人に宿題として分担したものをお互いにレビューしあいながら決めた。また、画面作成には、一昨年使われた提出課題登録をする松澤氏作のレポートシステムのインターフェースや TA を務めた川村氏へのインタビューなども参考にした。

こうして作成された画面遷移の主要部分を図 2.17 から図 2.20 に示す。主にユースケース毎に分け、シナリオを載せることで、理解しやくなるように工夫した。

### 2.1.6 開発の範囲

今期開発は、当初決めていた仕様の範囲全体から、ユースケース「採点する」をのぞいた部分になった。これは、開発期間の不足が原因であった。期間不足と判断した理由は、

- 仕様決定までに、2003 年 10 月初旬から 11 月中旬までと長期を要してしまった
- 課題という重要情報を扱うので信頼性・安定性必要である

などが挙げられる。採点部分は、今期の研究会後に追加される予定である。

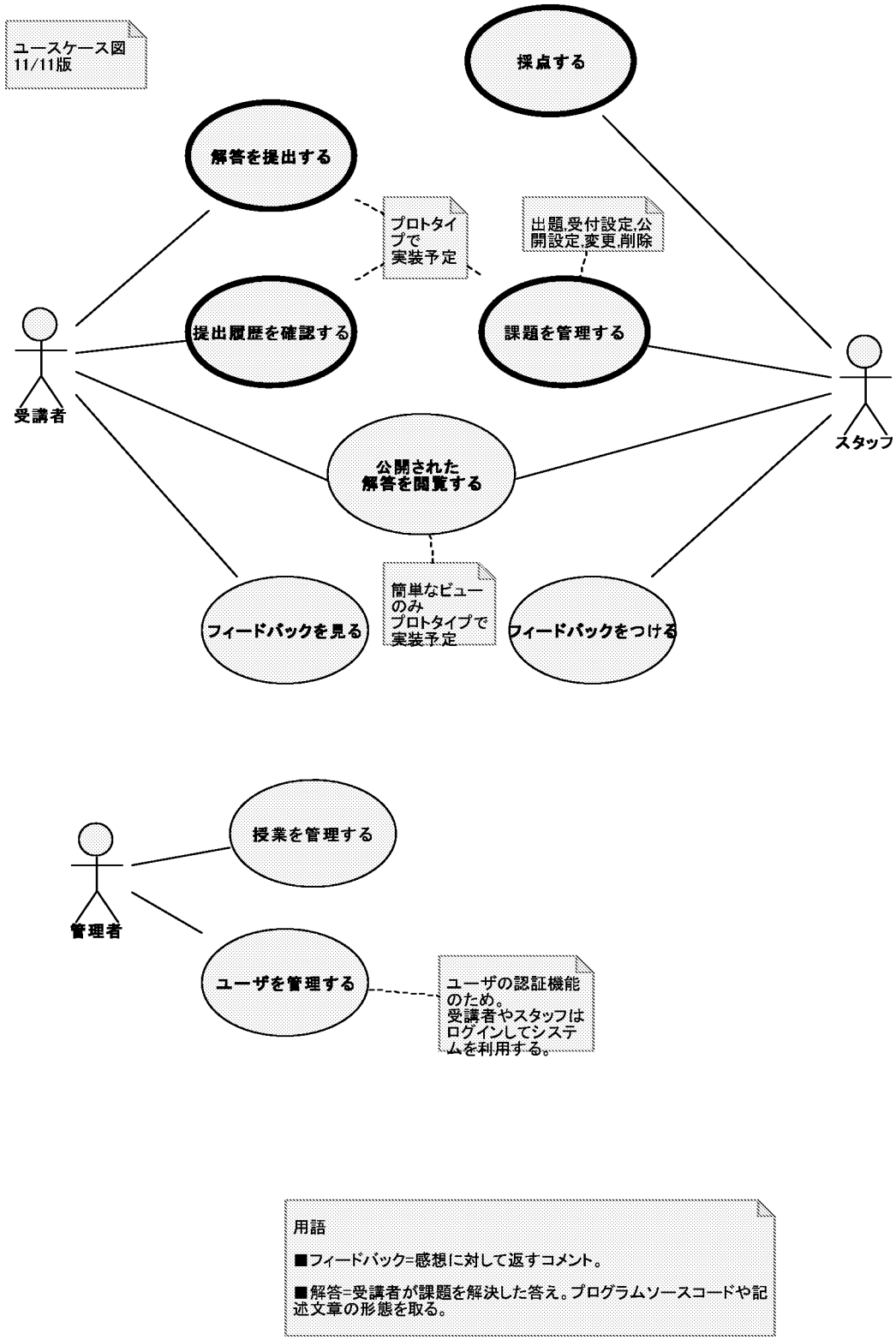



図 2.7: ユースケース図 (11 月 11 日)

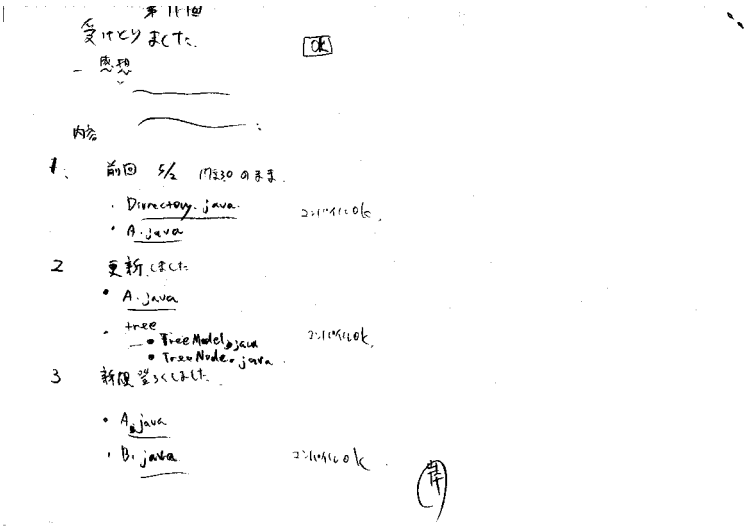



図 2.8: 提出画面メモ 1 頁

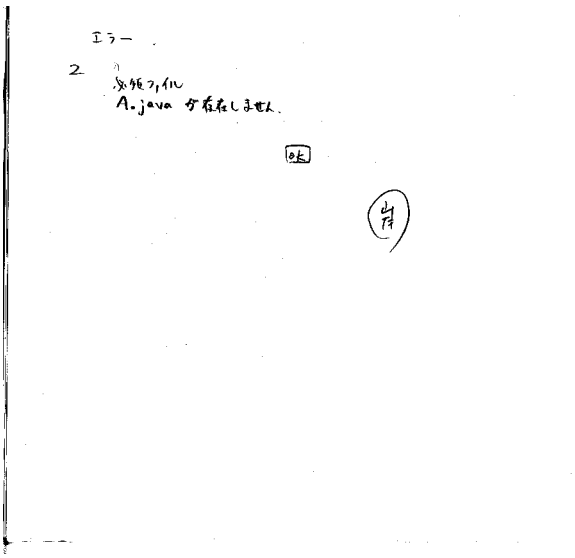



図 2.9: 提出画面メモ 2 頁

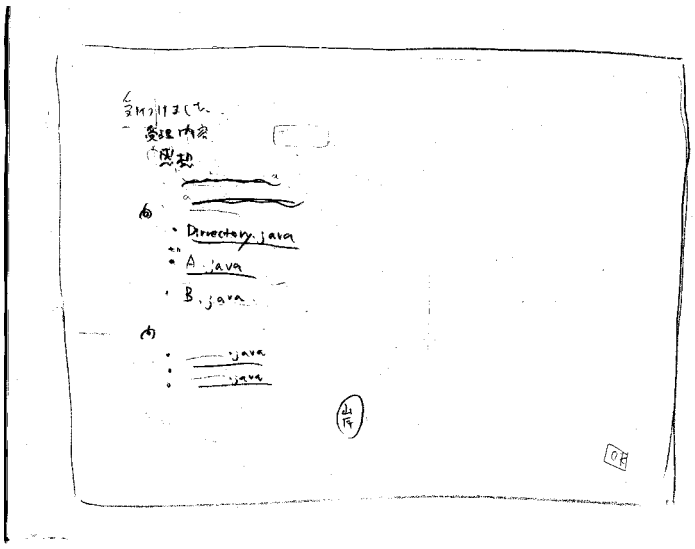



図 2.10: 提出画面メモ 3 頁

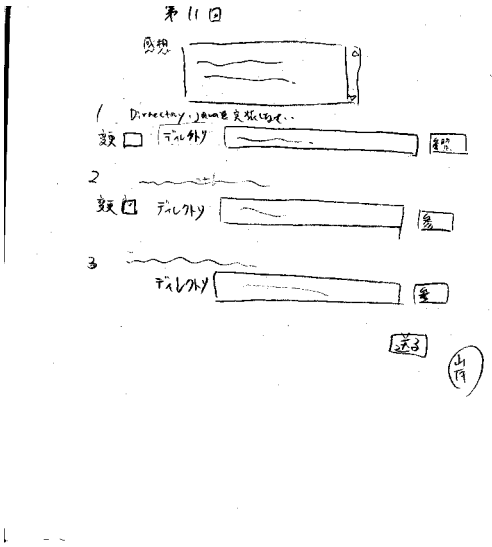



図 2.11: 提出画面メモ 4 頁

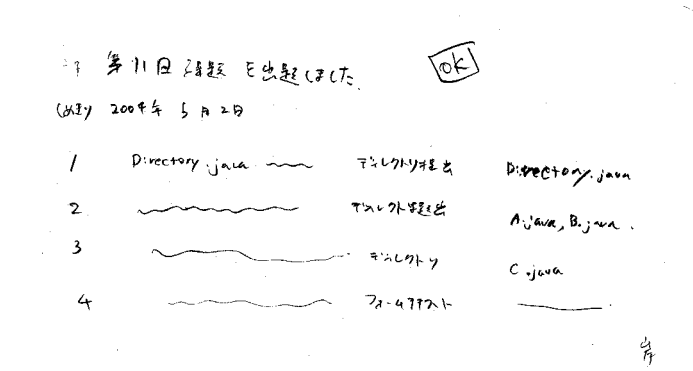



図 2.12: 提出画面メモ 5 頁

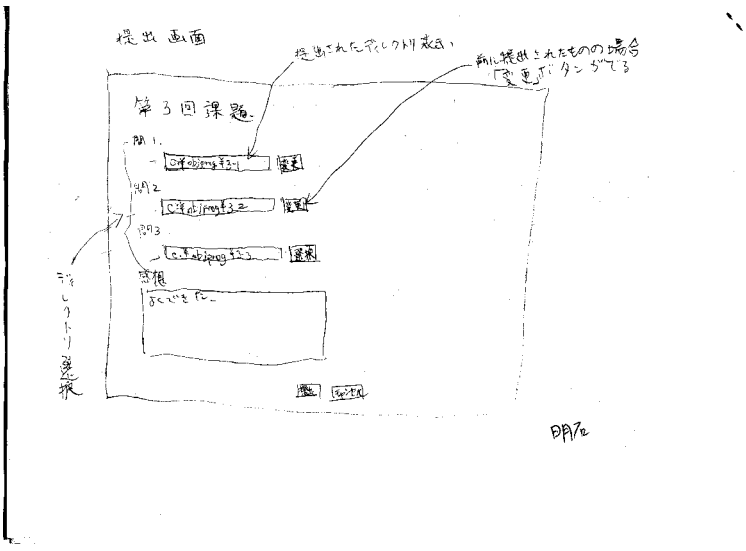



図 2.13: 提出画面メモ 6 頁

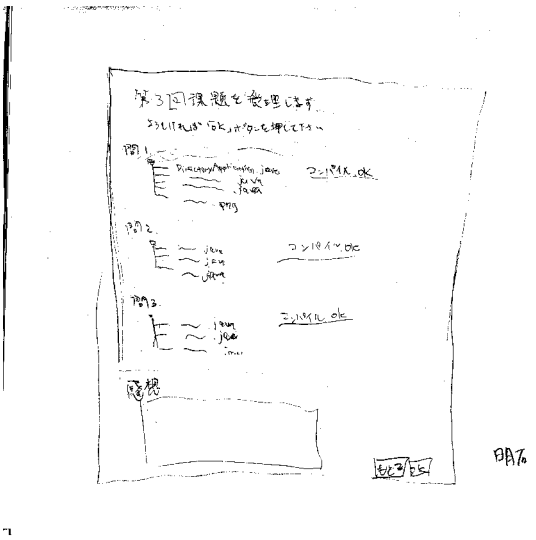



図 2.14: 提出画面メモ 7 頁

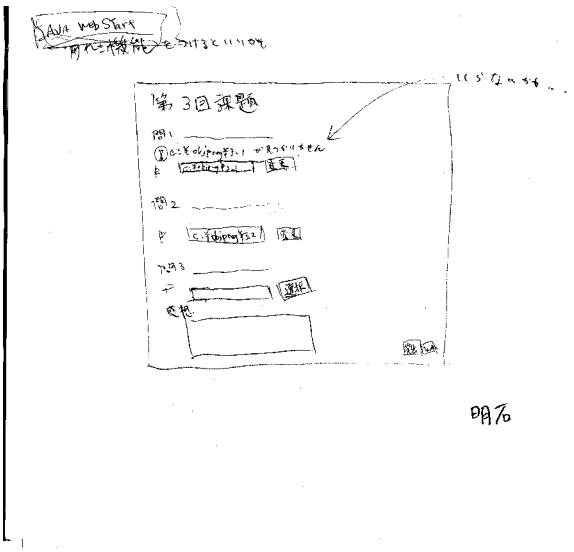



図 2.15: 提出画面メモ 8 頁

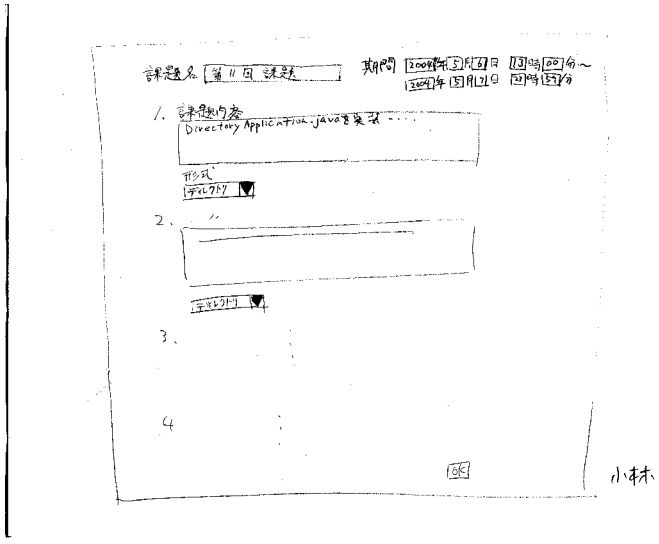
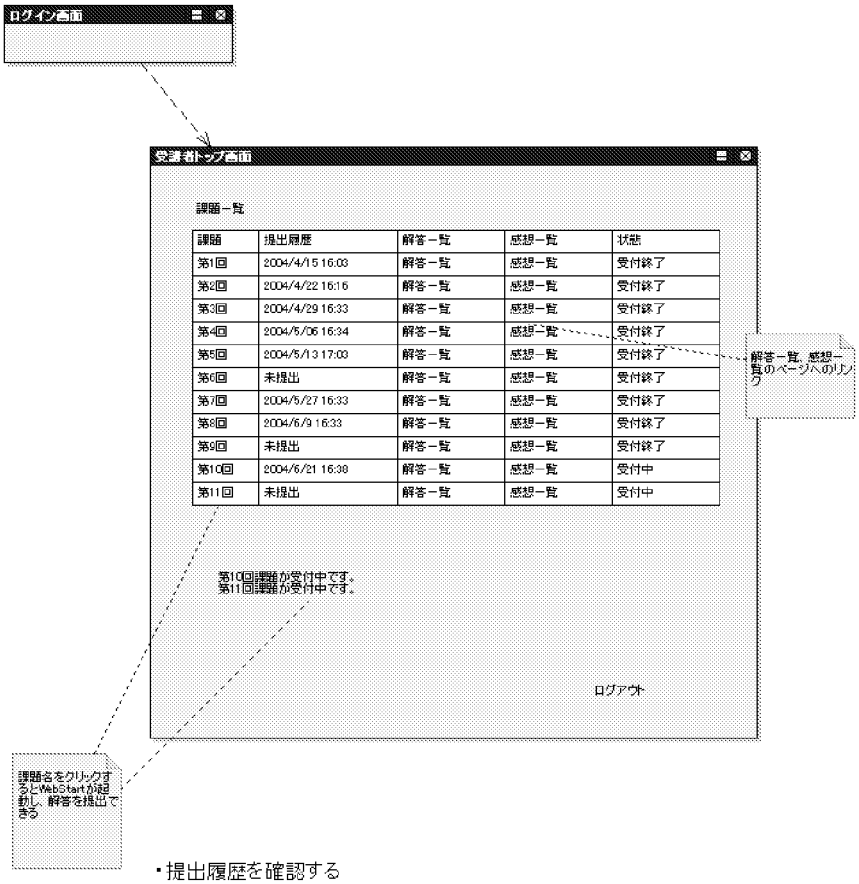


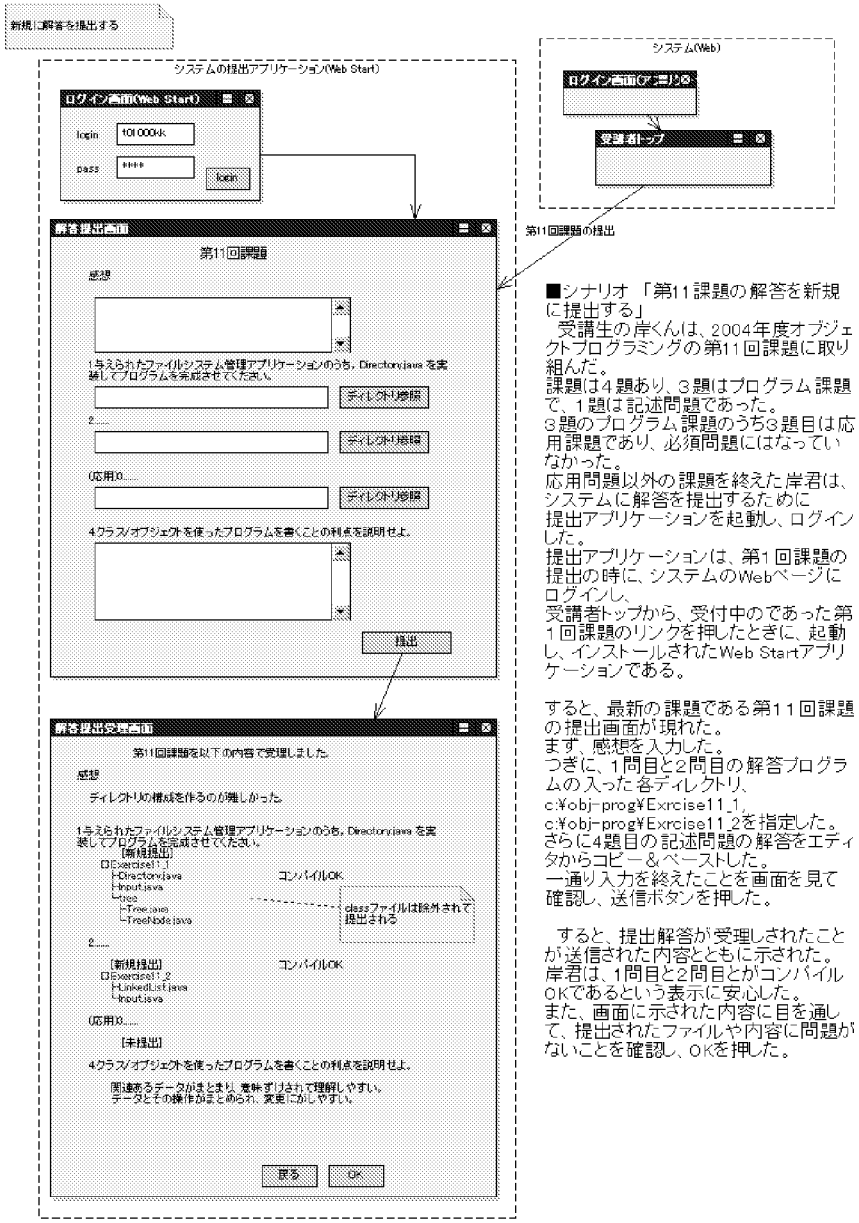

図 2.16: 提出画面メモ 9 頁




・提出履歴を確認する

オブジェクトプログラミングを履修中の小林君は月曜日が締め切りの第10回課題をきちんと期間中に提出できたかどうか不安に思った。そこでOPAS(仮)を利用し確認することにした。ログインページでログイン名とパスワードを入力しOPAS(仮)にログインするとOPAS(仮)の受講者トップ画面が表示された。受講者トップ画面には課題一覧が表示されてあり、第10回課題の提出時刻を見ると「2004/6/21 16:38」とあったので期間中に提出されており、小林君は安心した。

図 2.17: 画面遷移 1 頁

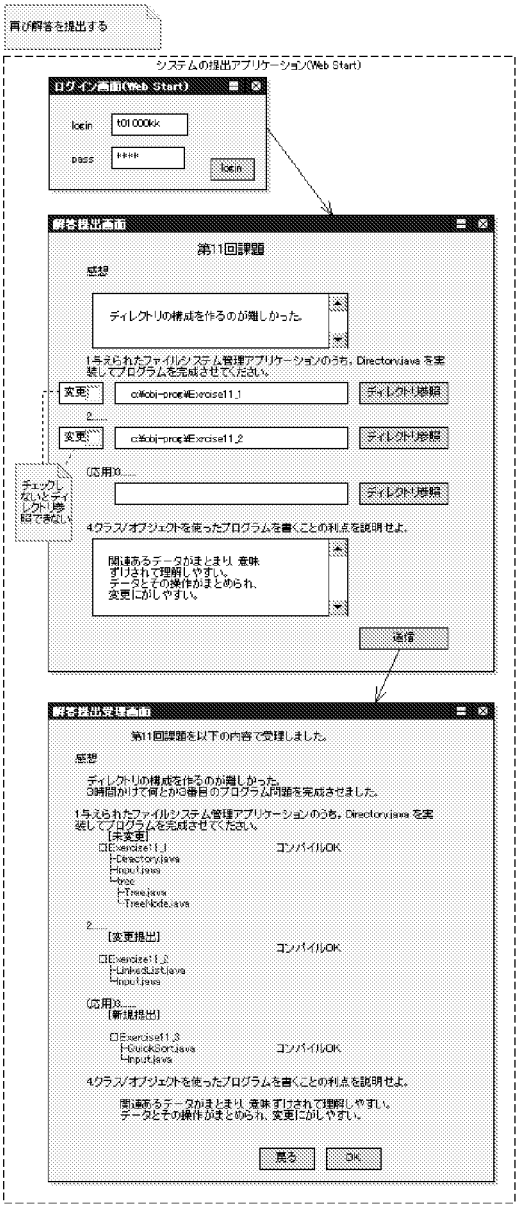



■シナリオ 「第11課題の解答を新規に提出する」
受講生の岸くんは、2004年度オブジェクトプログラミングの第11回課題に取り組んだ。
課題は4題あり、3題はプログラム課題で、1題は記述問題であった。
3題のプログラム課題のうち3題目は応用課題であり、必須問題にはなっていなかった。
応用問題以外の課題を終えた岸君は、システムに解答を提出するために
提出アプリケーションを起動し、ログインした。
提出アプリケーションは、第1回課題の提出の時に、システムのWebページにログインし、
受講者トップから、受付中のであった第1回課題のリンクを押したときに、起動し、インストールされたWeb Startアプリケーションである。

すると、最新の課題である第11回課題の提出画面が現れた。
まず、感想を入力した。
つぎに、1問目と2問目の解答プログラムの入った各ディレクトリ、
c:¥obj-prog¥Exrcise11_1、
c:¥obj-prog¥Exrcise11_2を指定した。
さらに4題目の記述問題の解答をエディタからコピー＆ペーストした。
一通り入力を終えたことを画面を見て確認し、送信ボタンを押した。

すると、提出解答が受理されたことが送信された内容とともに示された。
岸君は、1問目と2問目とがコンパイルOKであるという表示に安心した。
また、画面に示された内容に目を通して、提出されたファイルや内容に問題がないことを確認し、OKを押した。

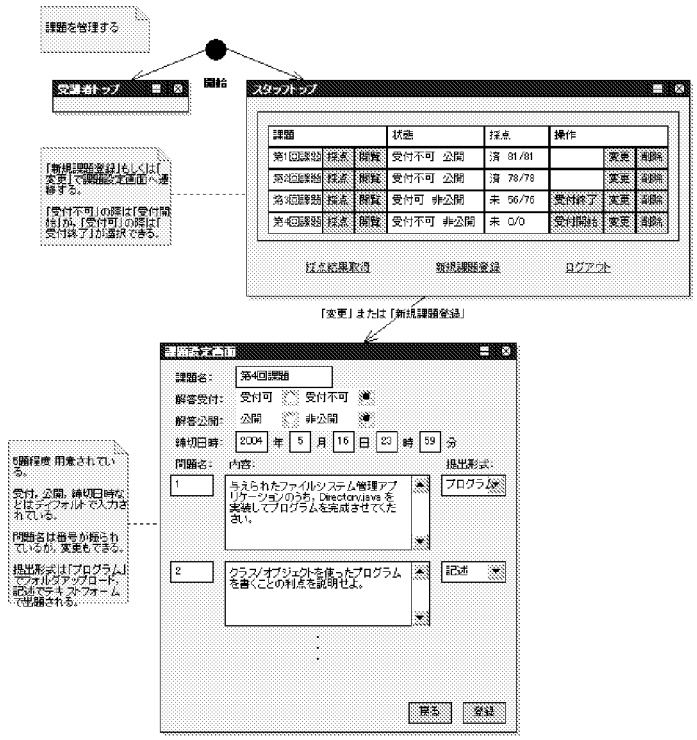
図 2.18: 画面遷移 2 頁




■シナリオ「再提出をする」
受講生の岸くんは、昨日、2004年度オブジェクトプログラミングの第11回課題に取り組み、必須でない3番目の問題以外を提出した。
今日、3番目の応用問題を終え、さらに、2番目の問題の手直しを行った。これらを提出するため、システムにログインした。

すると、提出を受け付けている第11回課題の提出画面が現れた。
前回提出の時に入力した内容(感想、1番目、2番目、記述問題)に関する情報は既に入力されていた。
感想に「3時間かけて何とか3番目のプログラム問題を完成させました。」を追加した。
さらに、2番目の問題の変更をチェックし、3番目の問題ののディレクトリc:¥obj-prog¥Exercise11_3を指定し、送信をした。すると、提出解答が受理されたことが送信された内容とともに示された。
岸君は、2問目と3問目が変更されたことを理解し、OKを押した。

図 2.19: 画面遷移 3 頁




2004年度のオブジェクトプログラミングのTAをしている明石くんは課題を出題することにした。

5/10(月)
第4回の授業をひかえた月曜日，明石くんは授業の資料をアップロードするとともに第4回の課題を出題することにした。
まずシステムにスタッフとしてログインし，「新規課題登録」を選択した。
すると「課題設定画面」が表示されたので，課題名に「第4回課題」と入力した。
その他はディフォルトのままなので，明石くんは内容を入力することにした。内容には
「与えられたファイルシステム管理アプリケーションのうち，Directory.java を実装してプログラムを完成させてください。」
というプログラム問題と
「クラス/オブジェクトを使ったプログラムを書くことの利点を説明せよ。」
という記述問題の2題を入力し，登録ボタンを押した。
登録ボタンを押すと，スタッフトップが表示され，そこに第4回課題が追加されていることを確認し，明石くんは満足した。

5/12(水)
第4回の授業前日，TA明石くんは課題の問題数が少ないと考え，1問追加することにした。
スタッフとしてログインした明石くんは，「第4回課題」の変更ボタンを押した。
すると前回登録した内容が入力された「課題設定画面」が表示された。
そこで，明石くんは3問目にプログラム問題を追加し，登録ボタンを押して終了しました。

5/13(木)
第4回の授業日，SAの小林くんは3限が終わったところで，第4回課題の受付を開始することにした。
スタッフとしてログインした小林くんは，第4回課題の受付開始ボタンを押した。
また，同時に第3回課題の受付を終了するために第3回課題の受付終了ボタンも押した。
そしてそれぞれの状態が「受付可　非公開」，「受付不可　公開」となっていることを確認し，ログアウトした。

こうして，第4回課題は受け付けられ，受講者は解答を続々と提出していくのでした。

図 2.20: 画面遷移 4 頁

## 2.2　概念モデルの分析

(文責:小林　敦)

前節で記された経過を経て、OPAS の仕様は決定した。その後、それに基づきどういったモデルが必要となるか、分析がなされた。この節では、まず概念モデルを作成するために、シナリオに沿ったオブジェクト図を考え、それを元に概念モデルを抽出していった過程を述べる。

### 2.2.1　オブジェクト図の作成

ここまでで分析された仕様を元に概念モデルを考えていった。概念モデルを考える上で、オブジェクト図を先に考え、その上で概念モデルを考えていくこととした。オブジェクト図を考える時点で、ミーティングに出席していたのは岸、小林の 2 名で、それぞれの考えるオブジェクト図を作成、比較した。図 2.21、図 2.22 が 2 人のオブジェクト図である。

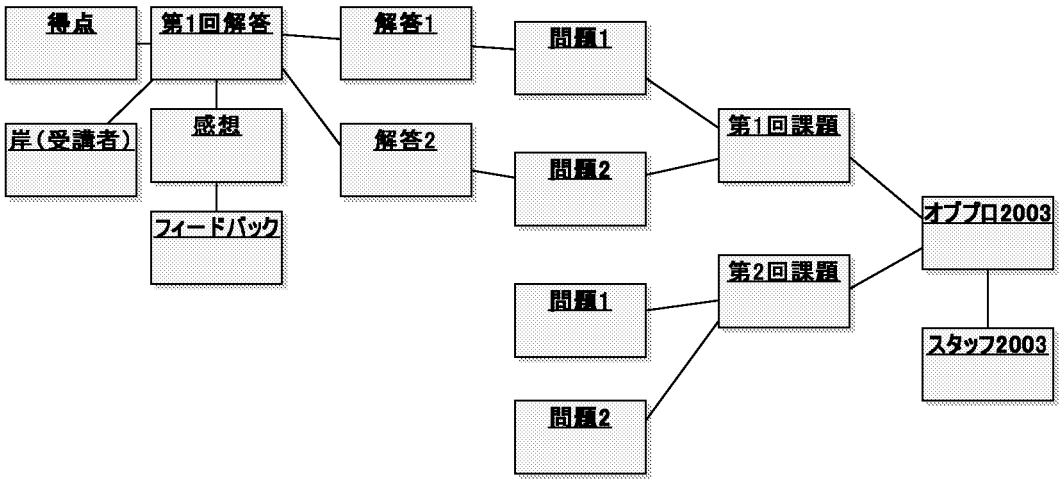



図 2.21: 仮オブジェクト図（岸案）

これらの図を元に話し合いながら、修正しながらオブジェクト図を完成させていった。注意点を以下に挙げていく。

- システム内には複数の授業が存在する
- スタッフという概念は必要ない
- 1 つの課題が複数の問題を持っていて、それぞれに対して受講者は解答を提出する
- 1 つの課題に対して受講者は 1 つの提出物をもつ
- 提出物は、1 つの課題に対する複数の問題をもつ
- 問題は記述問題と、プログラム問題のどちらかであり、両者は問題を継承する

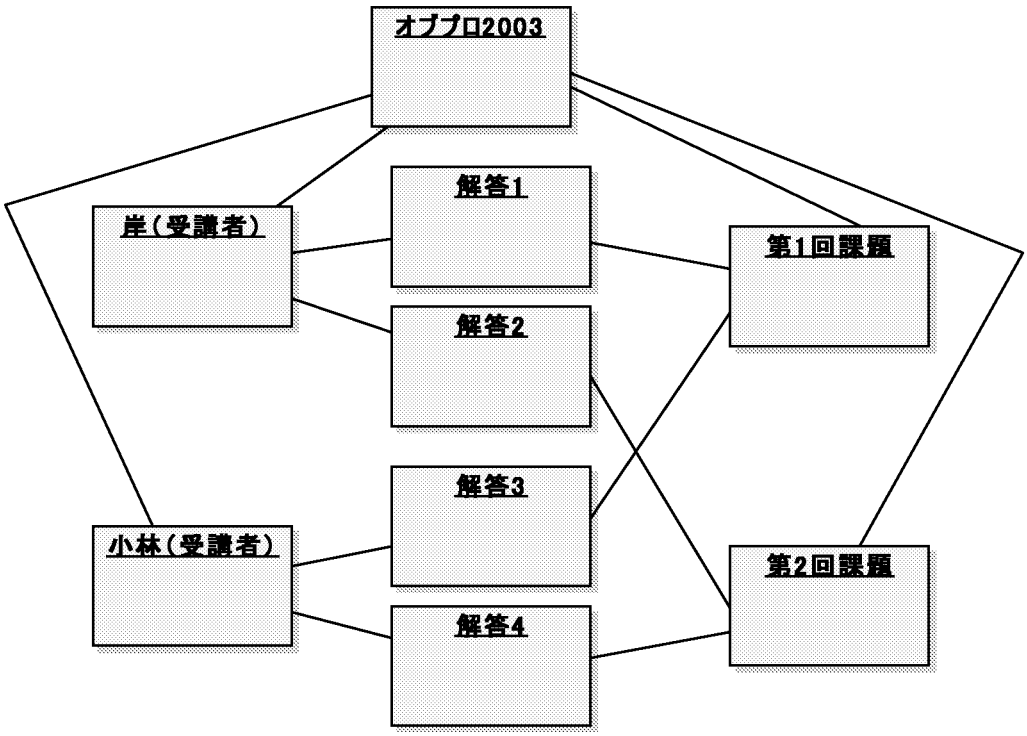
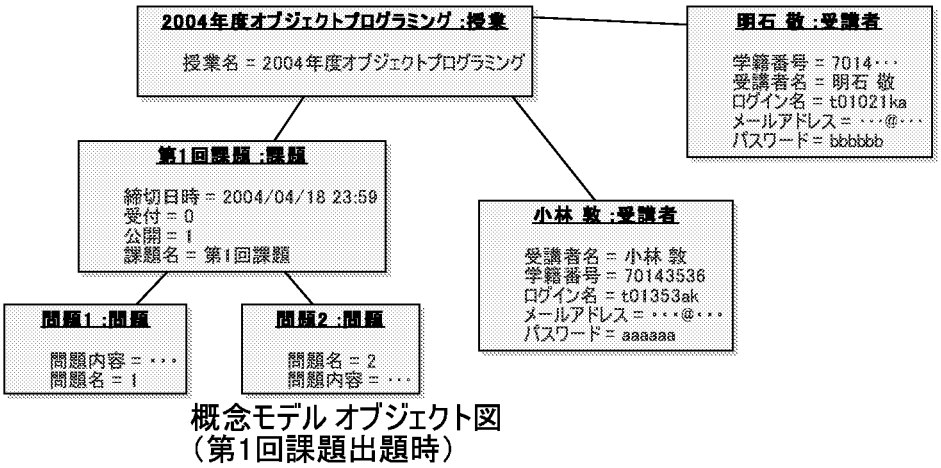



図 2.22: 仮オブジェクト図（小林案）

ここで、それぞれの根拠について触れていく。まず、複数の授業が存在することにしたのは、当初 OPAS はオブジェクトプログラミングにのみ、利用する予定であったが、プログラミング授業全般に使うこととなったためである。それぞれの授業に対して受講者が登録され、運用することが可能となっている。次に、スタッフという概念が必要ないことは、スタッフのユースケースから考えて、スタッフは共通で課題の出題や採点などをするだけであり、複数の人物を分ける必要がなく、スタッフ共通のログイン名、パスワードを授業の属性とすることで解決した。課題関連については、課題に問題が付属しているという出題形式から、受講者はそれに対応する問題を提出することになるが、それに対する点数は解答ごとに付けるわけではない。そのため、1 つの課題ごとに対応させた提出物という概念を作り、そこに点数を付けるために提出物という概念を作ることになった。これらの点に注意し、シナリオに沿って用意したオブジェクト図が図 2.23 から図 2.26 である。

### 2.2.2 クラス図の作成

次に、オブジェクト図から共通する概念を抽出しクラス図を作成した。これは上記のオブジェクト図についての議論と重複するが、各概念を決定し、その属性について考えていった。ここでは、オブジェクト図の作成の際の議論を利用したため、あまり手間を取らなかった。そこで完成したクラス図が図 2.27 である。




概念モデル オブジェクト図
（第1回課題出題時）

2004年度のオブジェクトプログラミングの初回授業が，4/15におこなわれた。
TAの説明にしたがって小林くんと明石くんは課題の提出に利用するシステムのユーザ登録をした。

また，早速第1回の課題が出題され，提出期限は4/18いっぱいとなっていた。
第1回の課題には問題が2つあり，2人は課題に取り組みました。

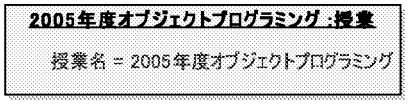



図 2.23: オブジェクト図（第 1 回課題出題時）

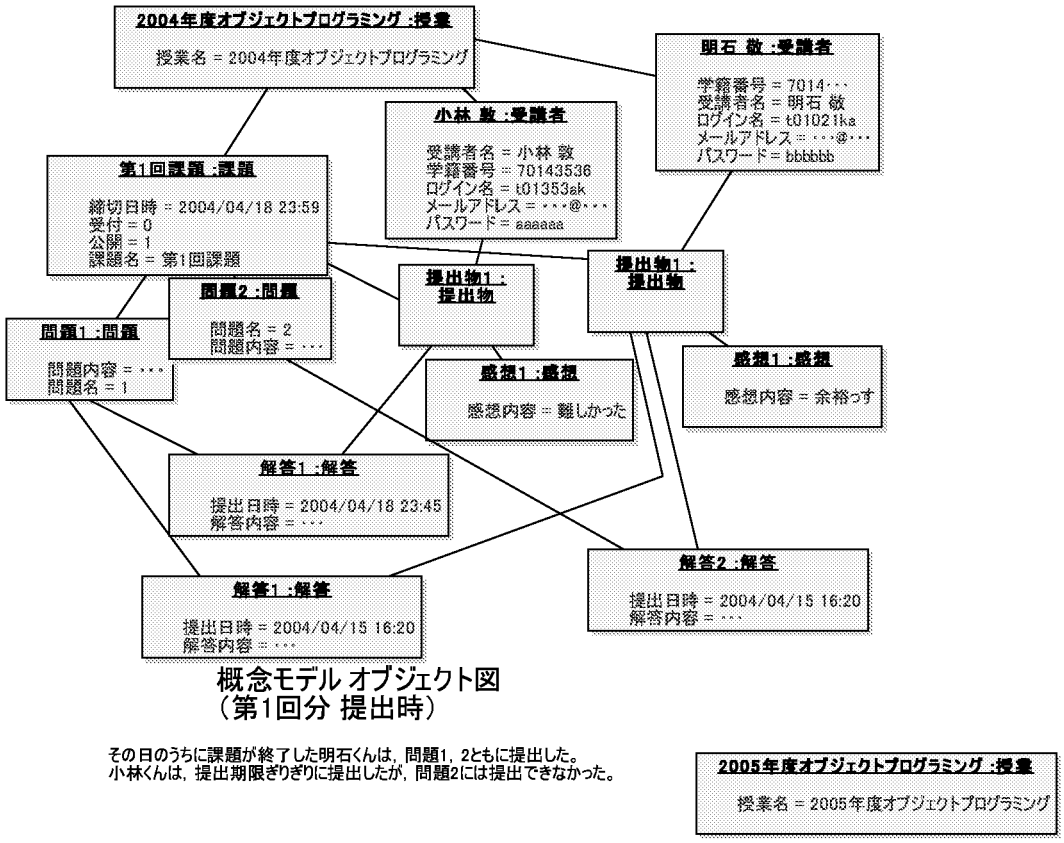



図 2.24: オブジェクト図（第 1 回課題提出時）

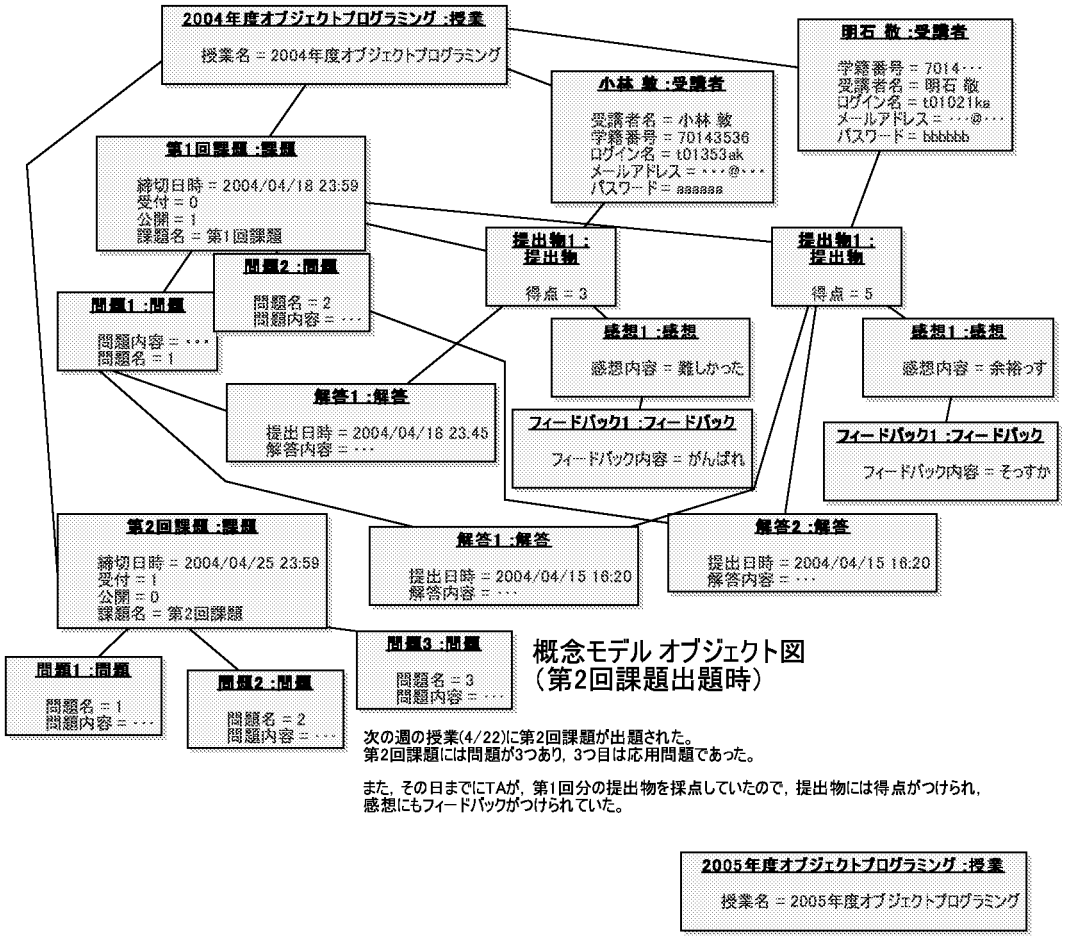



図 2.25: オブジェクト図（第 2 回課題出題時）

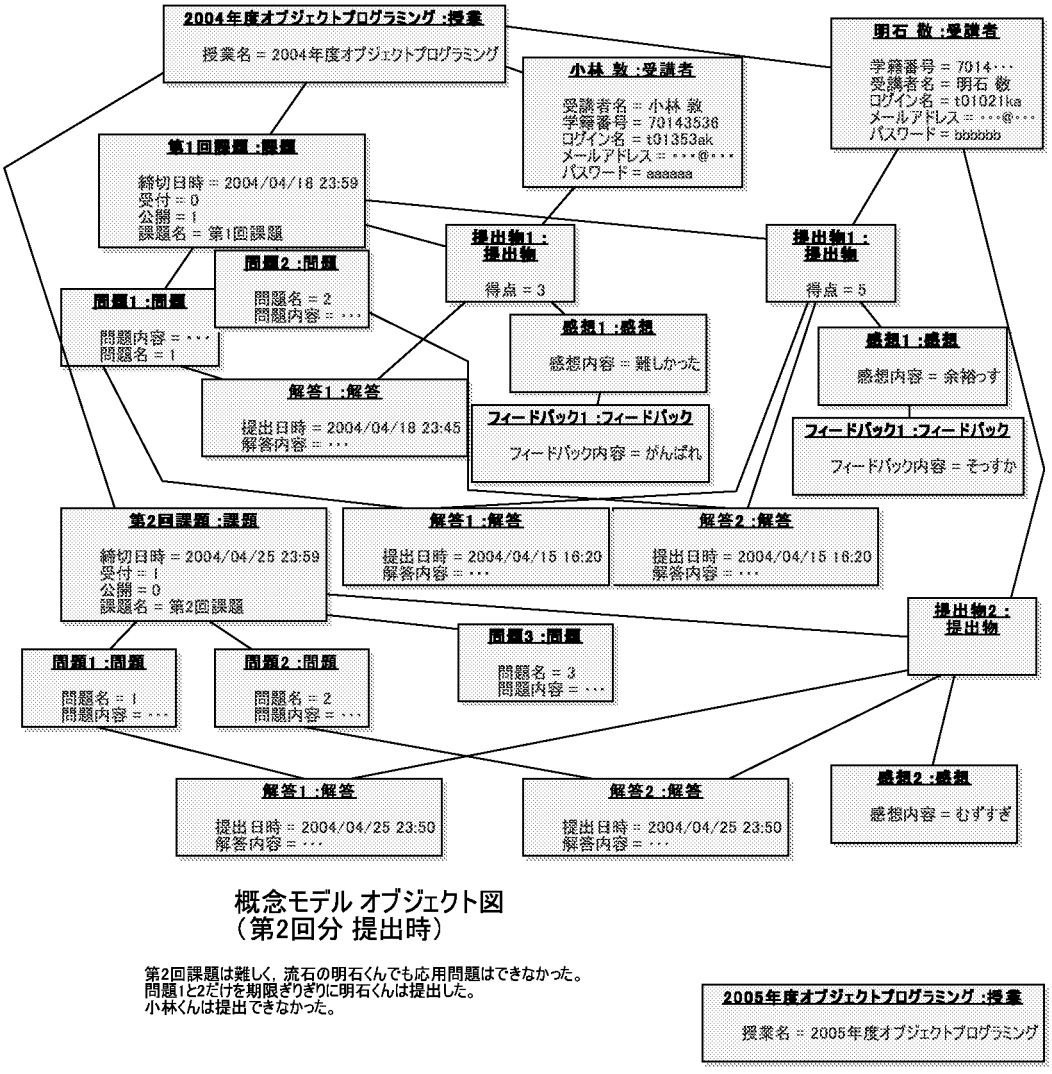



図 2.26: オブジェクト図（第 2 回課題提出時）

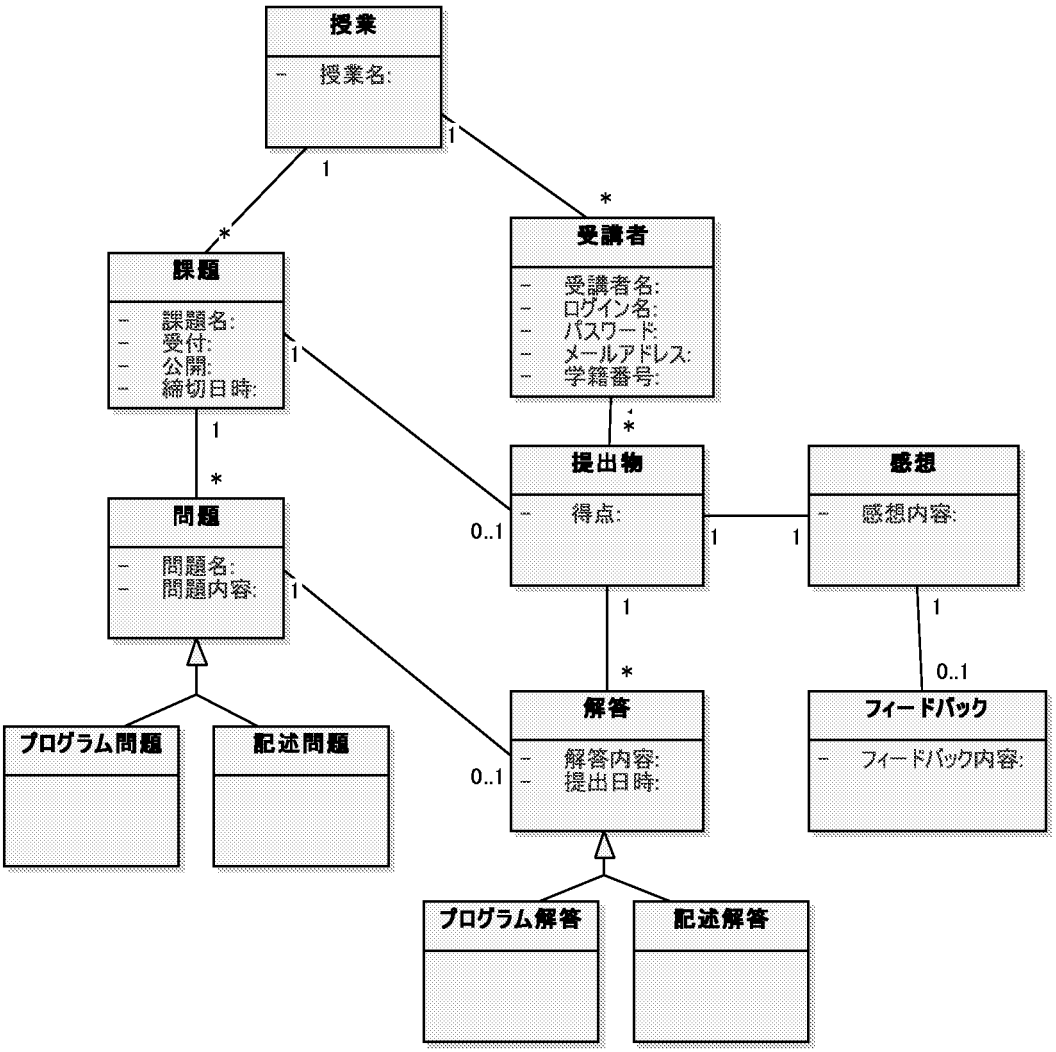



図 2.27: クラス図（合意版）

# 第 3 章

# 設計

## 3.1　アーキテクチャ

(文責:明石　敬)

### 3.1.1　必要とされるアーキテクチャ

仕様から次のような機能を実現可能なアーキテクチャが必要とされていると考えた。

- 多くの環境で実行することができる
- ディレクトリごとサーバーにアップロードできる

このため、まず、授業や課題の管理や提出物の閲覧などは、どのプラットフォームからもアクセスできるブラウザを通して行うことを考えた。そのため、Web アプリケーションとして OPAS を実装することに決定をし、Java Servlet を使うことに決定した。

しかし、問題は提出物の提出である。受講者が提出物を提出する際、受講者は解答のファイルをひとつずつ選択するのではなく、解答のファイルが入ったディレクトリを選択し、そのディレクトリごとサーバーにアップロードする。ブラウザで表示される html 形式のフォームではその仕様上ディレクトリごとサーバーにアップロードすることはできない。そのため html フォーム以外のアップロード方法が必要になった。

考えられた方法は以下のようである。

表 3.1: 考えられるアーキテクチャとその特徴

| | |
|---|---|
| ASP(Active Server Page) | microsoft が提供する IIS で動くスクリプト。windows ローカルマシンのディスクにアクセスできる。 |
| Java Applet + Servlet | ブラウザからプログラムを起動することができる。ブラウザにより動作が異なる場合もある。 |
| Java WebStart + Servlet | ブラウザからスタンドアローンの Java アプリケーションを実行させることができる。 |

ASP は html ページにスクリプトや ActiveX コンポーネントを組み合わせて強力な Web アプリケーションを作ることができ、windows ローカルマシンのディスクにもアクセスすることができる。しかし、サーバに Microsoft IIS を使用しなければならないことや windows 以外のマシンのディスクにはアクセスすることができないので採用しなかった。

次の Java Applet は自分たちでディレクトリにアクセスするためのプログラムを作ってしまえば、ディレクトリごとアップロードすることは不可能ではなかった。しかし、Applet はユーザーが InternetExplorer や Netscape,Opera など使用するブラウザによって Java VM が異なり、セキュリティ制限もまちまちであった。

JavaWebStart はブラウザから Java アプリケーションを実行する仕組みである。ブラウザ上の html ページから JavaWebStart の記述ファイルである jnlp ファイルにアクセスすることにより、サーバからアプリケーションをダウンロードし、実行することができる。実行するアプリケーションは Java で記述されているためプラットフォームに依存することなく、ローカルマシンの Java VM で実行されるため、ブラウザの種類により挙動が変わることはなくなる。

上記の 3 つを比較し、やはりブラウザから起動できること、プラットフォームに依存することなくローカルマシンのディスクにアクセスできることから 3 番目の Java WebStart+ Servlet という選択肢に至り、OPAS 提出アプリは Java で実装し、実行は WebStart から、という方式を採用した。

#### 3.1.1.1　ソケットサーバ vs サーブレット

提出アプリは WebStart と決まったが、提出されたファイルを受け取るサーバのアーキテクチャは確定していない。考えられた方法は次の二つだ。

表 3.2: 受け取りサーバの選択肢

| | |
|---|---|
| ソケットサーバ | ファイルの送受信が比較的楽にできる。管理が難しい。 |
| http サーバ (サーブレットを利用) | RFC で規定された形式でファイルの受信が可能。管理は比較的易しい。 |

ソケットサーバの使用はコストがかかる。アプリケーションから送られてくるソケットを的確に管理する必要もあり、また、WEB アプリケーションとの連携が難しく、実装もかなり難しい。

一方、http サーバ（サーブレット）で提出アプリを実装した場合、アプリケーションスコープのモデルにアクセスすることができる。さらに、RFC の規格にそっていればファイルをアップロードすることもでき、管理もソケットサーバに比べ、楽である。

アプリケーションスコープのモデルのアクセスとその管理コストから受け取りサーバは
http サーバ（サーブレット）で実装することになった。

#### 3.1.1.2 アーキテクチャの確定

最終的に決まったアーキテクチャは以下のようである。

表 3.3: アーキテクチャ

| | |
|---|---|
| OPAS サーバ | OPAS の Web アプリケーション部を担う。サーブレットで実装 |
| OPAS 提出アプリ | 受講者が提出物を提出するためのアプリ。WebStart でブラウザから起動。 |
| OPAS 提出アプリケーションサーブレット | OPAS サーバの一部。提出アプリに必要な情報を送信や、提出物を受信する |

アーキテクチャとユースケースとの対応をで示す。図 3.1

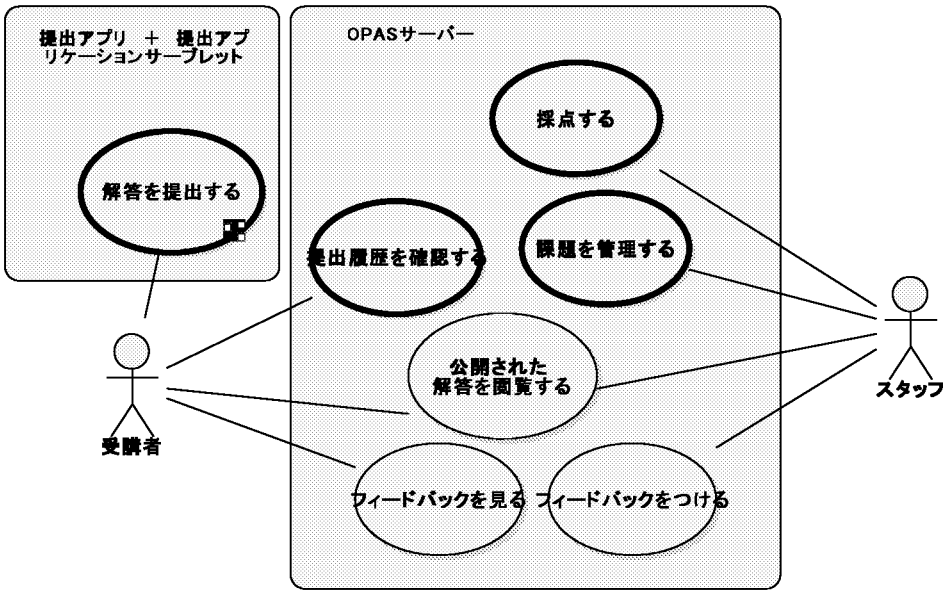



図 3.1: アーキテクチャとユースケース

## 3.2　モデルの設計

**(文責:岸　健司)**

前章において、概念モデルは既に示されている。この節では、実装に向けて、概念モデルをどのようにして設計モデルへ変換していったかを示す。

### 3.2.1　モデル要素

今回の設計モデル作成の第１歩は、概念モデルのクラス図をそのまま設計モデルのクラス図として見なすことから始まった。１つ１つの授業、受講生、課題などのクラスがそのまま設計のクラスであると考えた。

ただし、日本語名では、環境によって問題になる場合があるので、英語名に変換することにした。英語名への置き換えは、辞書を用いてもなかなか難しい。例えば、「問題」クラス１つしても英語訳をある人は Problem と言い、またある人は Question と訳す。さしあたって、開発者間で分かり易い言葉遣いであればよいと判断し、無難で聞き覚えのある単語または多数決でそれぞれの英訳を決めた。図 3.2 に英語訳した設計モデルを示す。

関連については、実装の容易さを考え、当初、概念モデルでクラス同士の双方向関連も単方向で実装可能な部分は単方向の関連とした。しかしながら、最終的には、実装が困難な箇所が発生したため、概念モデル通りの双方向に戻した。筆者は同研究会の別プロジェクト「みるみる」においても同様に後から双方向関連に設計を戻すことを経験している。もしかしたら、概念モデルの双方向関連が元々妥当である場合、実装の都合で、無理矢理変更すべきでないのかもしれない。

継承についても、そのまま設計モデルに引き継いだ。しかし、問題に、プログラム問題と記述問題の継承関係を残したことは、実装に都合が悪かった。プログラム問題と記述問題２つは、所有する情報の違いがほとんどないが、２つの問題に対する処理は大きく異なった (表示などが大きく異なる)。継承のメリットであるポリモーフィズムの恩恵があまり受けられないかった。

また、この２つの問題の種類間での種類変更があった。継承による設計であると、クラスの変更につながるが、クラスの変更を行うのは Java での実装はコストがかかる。

結局、概念としてこの継承はわかりやすかったが、実装に大きなコストを掛けさせるだけで、メリットが余り得られなかった。

私は、この継承問題に実装の段階で気づいたのだが、そのまま実装を続けた。設計変更をメンバーに了承を得ることと実装をやり直す手間を恐れて、そのままにした。しかし反省点として、気づいた時点で、すぐに設計を変えるべきだったと考えている。なぜなら、この問題は後々の永続化の実装等にも大きなコストを費やしたからである。長期的に見て、目先にとらわれない変更が適当であったと考える。

属性については、概念モデルをそのまま引き継いだが、実装の上の都合で若干追加した

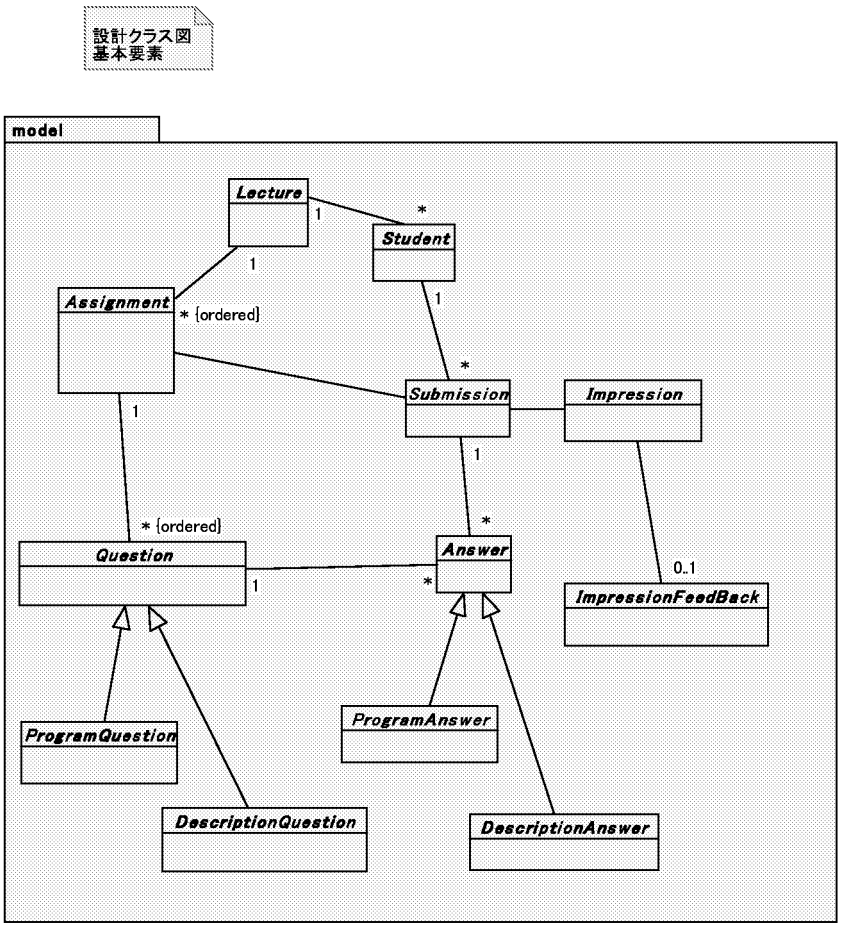



図 3.2: 設計モデルの基本要素

物もある。

### 3.2.2 実装のための要素

#### 3.2.2.1 Facade

本システムの設計には、「The Gang of For」のデザインパターンの 1 つ Facade パターン (詳しくは「『デザインパターン』ソフトバンク,Erick Gamma,Richard Helm 他 2 名著」を参照) を用いた。

今回このパターンを使った目的は、モデル要素の使い方を規定し、開発のスピードを上げることとバグを減らすことである。今回の開発はクラス数も多くその使い方が複雑である。モデル群の使い方をすべて Facade クラスにまとめ、外部からはこのクラスを通してモデルにアクセスしてもらっている。このクラスの利用者は細かいモデル要素の使い方を気にすることなく、また、間違った使い方をしないで、抽象度の高い処理が行える利点がある。今回、この Facade は提出アプリケーションやその他の Web アプリケーション部

分の開発で大きな利益をもたらしてくれた。

この Facade の重要な処理には、HCP チャートによる設計を行い、これをもとに実装コードに落とし込んだ。HCP チャートの記述は、不慣れであった私には、手間のかかる物であったが、その目的や粒度を意識してた処理記述は、理解の容易性では優れていると感じた。用いた HCP チャートを図 3.3 から図 3.6 に示す。

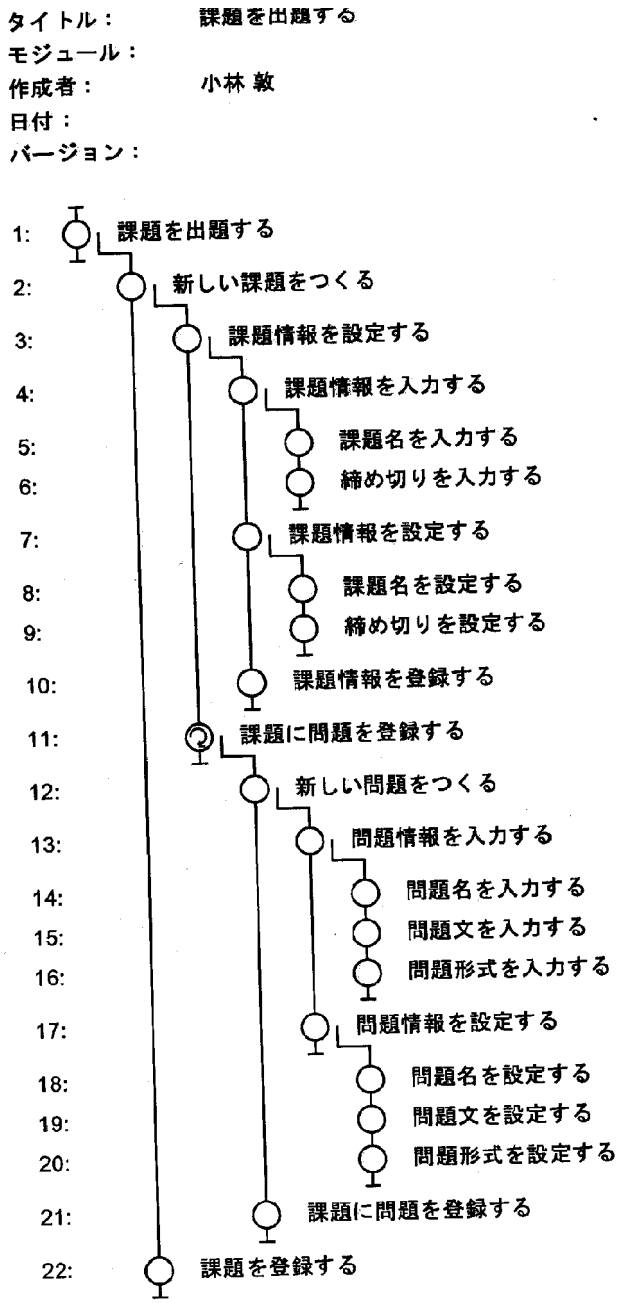



図 3.3: HCP1

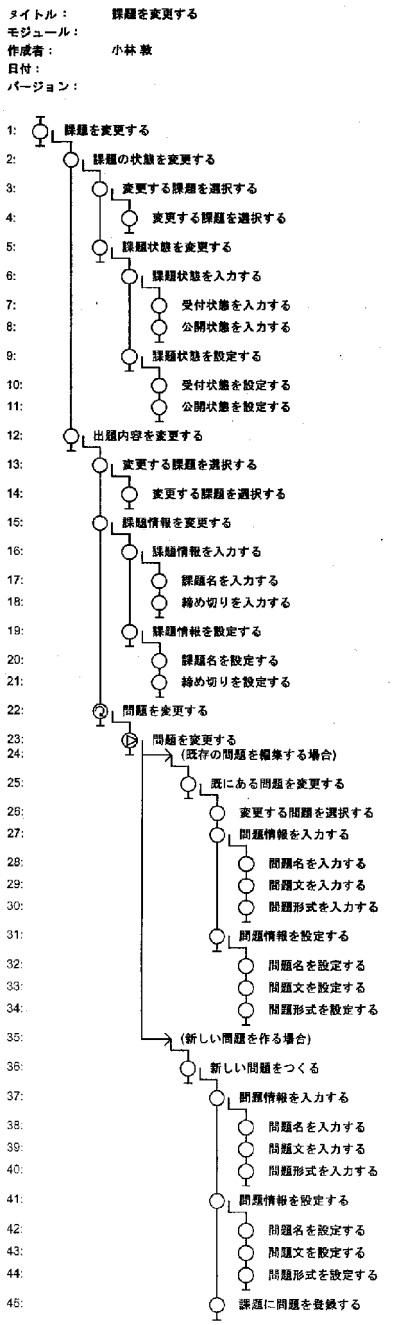

タイトル： 課題を変更する
モジュール：
作成者： 小林 敦
日付：
バージョン：
1: 課題を変更する
2: 課題の状態を変更する
3: 変更する課題を選択する
4: 変更する課題を選択する
5: 課題状態を変更する
6: 課題状態を入力する
7: 受付状態を入力する
8: 公開状態を入力する
9: 課題状態を設定する
10: 受付状態を設定する
11: 公開状態を設定する
12: 出題内容を変更する
13: 変更する課題を選択する
14: 変更する課題を選択する
15: 課題情報を変更する
16: 課題情報を入力する
17: 課題名を入力する
18: 締め切りを入力する
19: 課題情報を設定する
20: 課題名を設定する
21: 締め切りを設定する
22: 問題を変更する
23: 問題を変更する
24: (既存の問題を編集する場合)
25: 既にある問題を変更する
26: 変更する問題を選択する
27: 問題情報を入力する
28: 問題名を入力する
29: 問題文を入力する
30: 問題形式を入力する
31: 問題情報を設定する
32: 問題名を設定する
33: 問題文を設定する
34: 問題形式を設定する
35: (新しい問題を作る場合)
36: 新しい問題をつくる
37: 問題情報を入力する
38: 問題名を入力する
39: 問題文を入力する
40: 問題形式を入力する
41: 問題情報を設定する
42: 問題名を設定する
43: 問題文を設定する
44: 問題形式を設定する
45: 課題に問題を登録する

図 3.4: HCP2

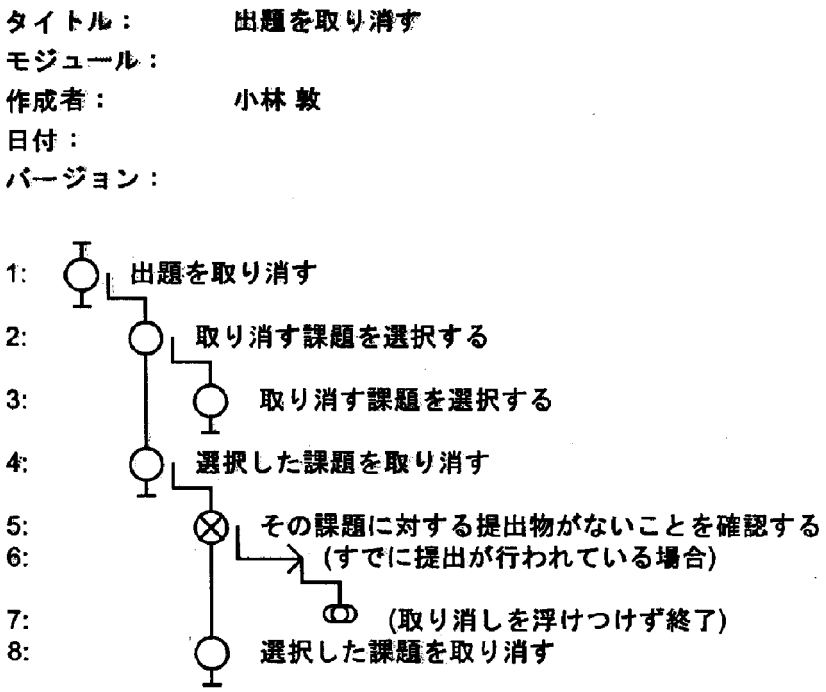



図 3.5: HCP3

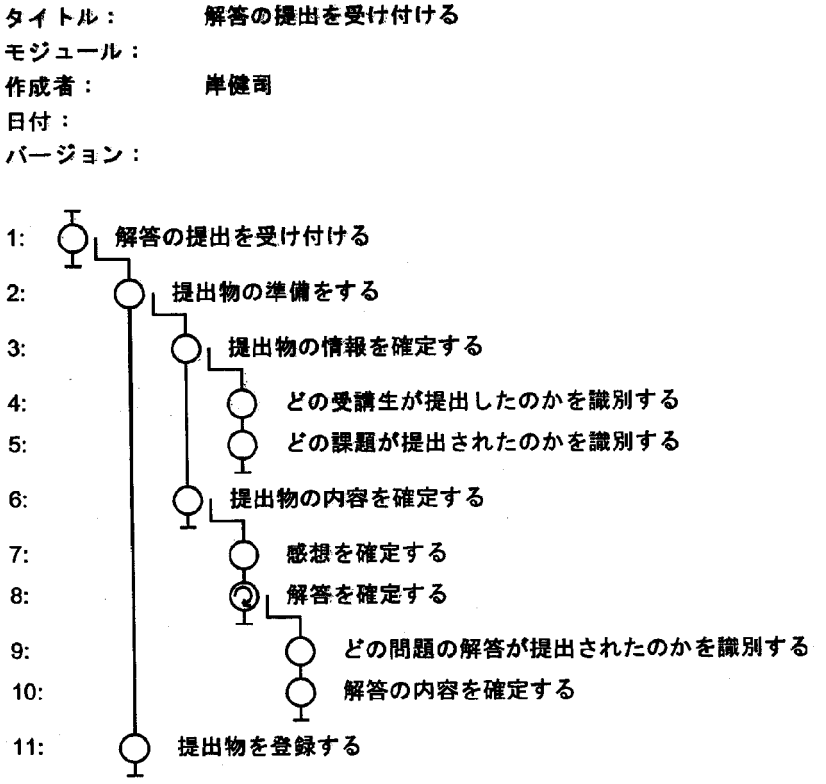



図 3.6: HCP4

#### 3.2.2.2 その他

実装上都合がよいということで、つぎのクラスを追加してある。

- SourceFile…提出されたフォルダ (パッケージ)、ソースファイルを表す
- OpasException…OPAS 独自の例外を表す
- Point…提出物の得点を表す
- ElementManager…モデル要素の生成、永続機構やメモリから取得などを担当する

など。

### 3.2.3 シリアライズでの永続化

初期の開発では、データの保持は一時的にはメモリにすべてのデータを展開し、永続化には、オブジェクトをシリアライズ (直列化) してファイルに書き込む方法をとった。これは、実装の容易さのためである。ただし、将来的に、ファイルへ書き込む保証がなかったため、ファイルへの依存性はおおよそ分離しておいた。

### 3.2.4 RDB ベースの永続への移行

本システムは、課題という重要データを扱う。そのため、データ永続においても信頼性を確保する必要があった。データは更新されるたびに速やかに確実に永続化される必要があった。

そこで我々は、データストアとして、リレーショナルデータベース（PostgreSQL）を選んだ。

#### 3.2.4.1 O/R マッピング

オブジェクトで表現されたデータをリレーショナルデータベースに保存するには、テーブル表現に変換する必要がある。我々はまず、保存すべきモデルの要素に対応するスキーマを考えることになった。

図 3.7 が成果物になる。

スキーマはなるべく簡潔に表現するようにした。原形はクラス図である。１クラスを１リレーションで表現する方針でほとんどの部分は、そのまま置き換えた (ただし、hidden_files は例外)。

オブジェクトのＲＤＢへのマッピングで問題になるのは、関連と継承である。関連については、今回幸運にも多対多の関連がなかったので、関連だけを表すテーブルを用意する必要はなかった。関連はすべて、関連するどちらかのクラスを表現するリレーションに外部参照キーをもうけた (同時にすべてのリレーションに主キーを設けた)。また、継承は

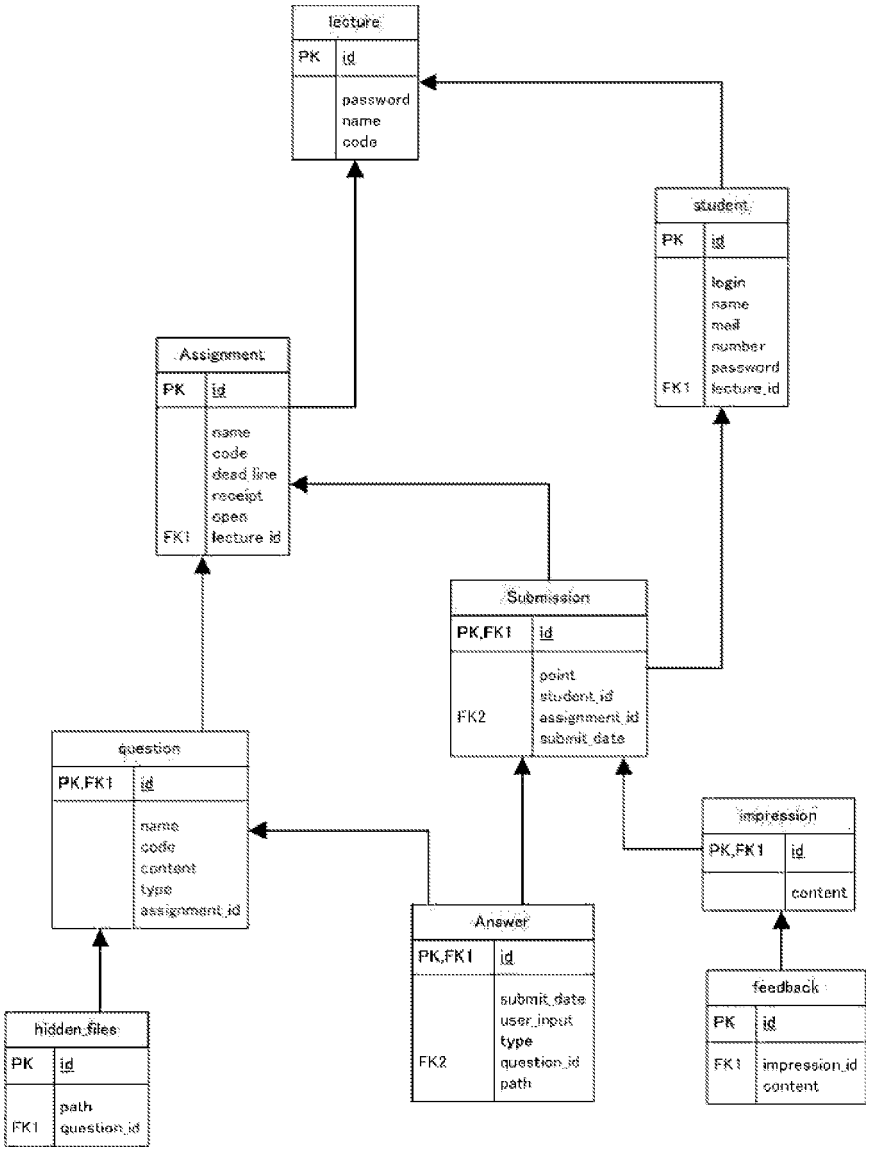



図 3.7: データベーススキーマ

スーパークラスを表現するリレーションに、サブクラスの属性も含ませ、リレーションを減らした。この継承の解決方法はアクセススピードが得られる反面、無駄な保存領域が生じたり、新たなサブクラスに対する拡張性に乏しいデメリットもあったが、どちらも問題はないと判断した。

スキーマを作ってみて, オブジェクトとリレーションのマッピングコードを書くのは、かなりの作業量になると考えられた。ちょうどその時、中鉢氏からのアドバイスがいただけた。「最近よく使われる「Torque」という O/R マッピングのフレームワークがある」とのことだった。PM の判断としては、「技術調査をして、すぐに使えるのであれば使ってみよう」とのことだった。すぐに、岸中心に技術調査が行われ、結果、Torque のフレームワークを利用することになった。

Torque の利用によって以下のメリットが生まれた。

- SQL など低次元のプログラム記述が不必要になる (自動生成や高次元の記述が可能)
- 特定の RDB への依存しない
- データベースのコネクションプリーリング、メモリキャッシュなど性能的機能が簡単に利用できる

逆にデメリットとしては、

- フレームワークの概要を学ぶ必要がある (まだ、マニュアルが整備しきっていない)
- プログラムの可読性が下がる（継承や依存関係が複雑）
- 設計の自由度が下がる (永続化するオブジェクトの継承の自由度が下がる)

Torque フレームワークにモデル要素を当てはめた設計は図 3.8 のようにした。これは受講生クラスだけを示した物だが、他のクラスも同様である。

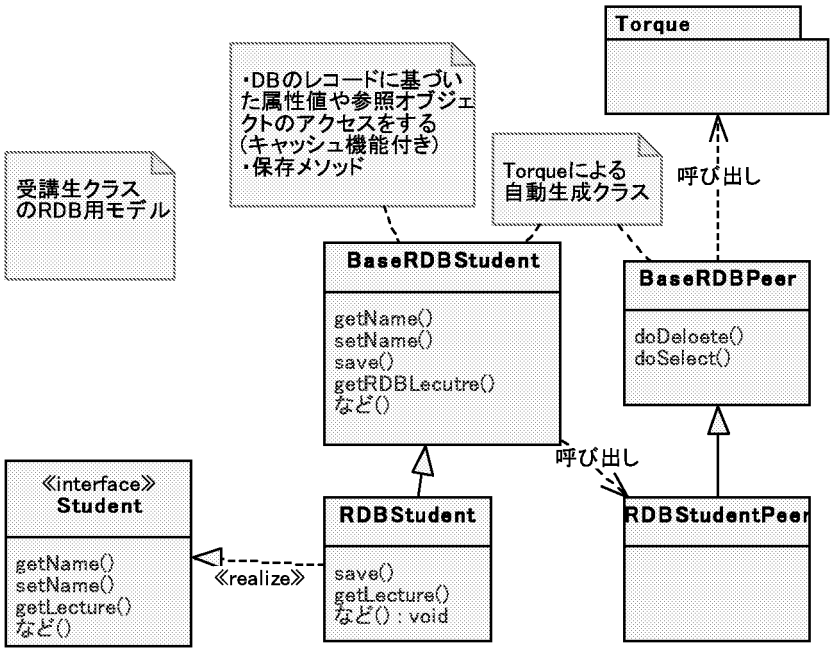



図 3.8: RDBStudent の例

後から、Torque フレームワークの利用を決め、ソースコードの可読性や、シリアライズ永続化との両立を考え、やや複雑になっている。

#### 3.2.4.2　移行問題

今回、永続化の RDB への変更作業と同時期に別の Web アプリケーション部分の開発も行われていた。スムーズにシリアライズ永続化のモデルから移行する必要があった。したがって、容易に切り替え可能な作りにした。モデル関連のパッケージの関係は図 3.9 のようにした。これは受講生クラスだけを示した物だが、他のクラスも同様である。

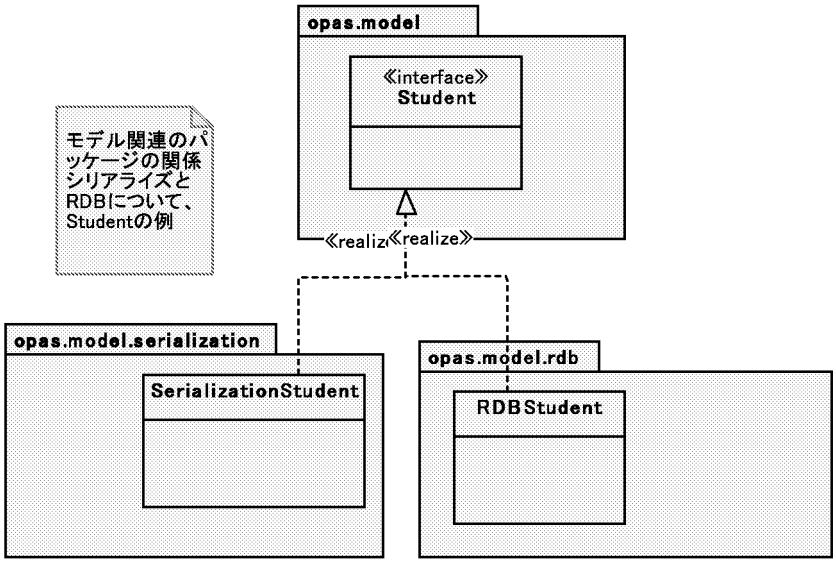



図 3.9: serialization と rdb パッケージの関係 (RDBStudent の例)

#### 3.2.4.3　メモリと RDB の同期

今回、バグの巣となっていた箇所があり、そこにはメモリ上のオブジェクトと RDB の情報同期の問題がある。オブジェクトをメモリで一元管理せずに、必要なときに必要なだけ、RDB のレコードからインスタンスが作られるため、状況は複雑である。同じ ID を持つオブジェクトがメモリ上に２つ存在する状況もあり得るのである。

現在の同期の方針の特徴的な部分を簡単に示す。

- オブジェクトの内容変更をする更新処理
  - 変更対象が RDB に過去保存されていない一時的なオブジェクトなら、その内容を変更して処理を終了する (RDB には保存しない)
  - 変更対象が RDB に過去保存されていてたら、パラメタとして渡されたメモリ上の変更対象および、RDB のレコード情報を変更する
- DB レコードの更新処理の際、渡されたパラメタは RDB レコードから取得し直した物を使う

- RDB のレコードの削除は保存時に他のオブジェクトとの参照関係がとぎれたときに行う

#### 3.2.4.4 参考文献、情報


- 小林 (孝) 氏の Torque サンプル
- Jakarta プロジェクト徹底攻略 (技術評論社) 2002 年
- Java World 2003 年 9 月号 (アイ・ディ・ジー・ジャパン)
- The Apache DB Progect torque[http://db.apache.org/torque/
- 実践 UML 第 2 版　34 章パターンを用いた永続化フレームワークの設計 (ピアソン・エデュケーション) クレーグ・ラーマン著、依田光江訳

## 3.3　Web アプリケーションの設計

**(文責:小林 敦)**

この節では、前節までで設計されたモデルを利用して、どのように Web アプリケーションを設計したかについて触れていく。まず、サーブレットと JSP の遷移について触れ、それから、特に OPAS の中核となる OpasServlet についてを中心に記していきたいと思う。

### 3.3.1　遷移に関して

OPAS は主にサーブレットと JSP によって構成され、Smalltalk などで広く使われている MVC 構造を、Web アプリケーションに応用した MVC2 構造と呼ばれるアーキテクチャを採用している。OPAS では、ある操作を実行する際には MVC 構造でいう View（表示）の役割を果たしている JSP から、Controller（処理）の役割であるサーブレットを呼び出し、サーブレットが処理をした結果を、また View である JSP に渡すという遷移を基本としている。この遷移の設計については 3.3.2.2 節で述べる。

### 3.3.2　OpasServlet に関して

OpasServlet とは OPAS の中核となるサーブレットである。他の多くのサーブレットが OpasServlet を継承しており、一般のサーブレットが共通してするべき作業をこなしている。OpasServlet に関する継承図は図 3.10 となっている。

その主な役割としては

- モデルへのアクセス
- マッピング

の二つが挙げられる。これらについて述べていく。

#### 3.3.2.1　モデルへのアクセス

OPAS のモデルでは、サーブレットがモデルを取得したり操作する際には、Facade を通じてモデルにアクセスする。その Facade を利用するために、ほとんどのサーブレットは Facade のインスタンスが必要である。OpasServlet にはそういったサーブレットに共通する操作を、他のサーブレットに代替して行う働きがある。OpasServlet を継承しているサーブレットは、OpasServlet で生成されている Facade のインスタンスをなにも気にせず getModel() というメソッドを利用することで扱うことができる。

また、モデルを利用する操作として Session についての操作も果たしている。具体的に

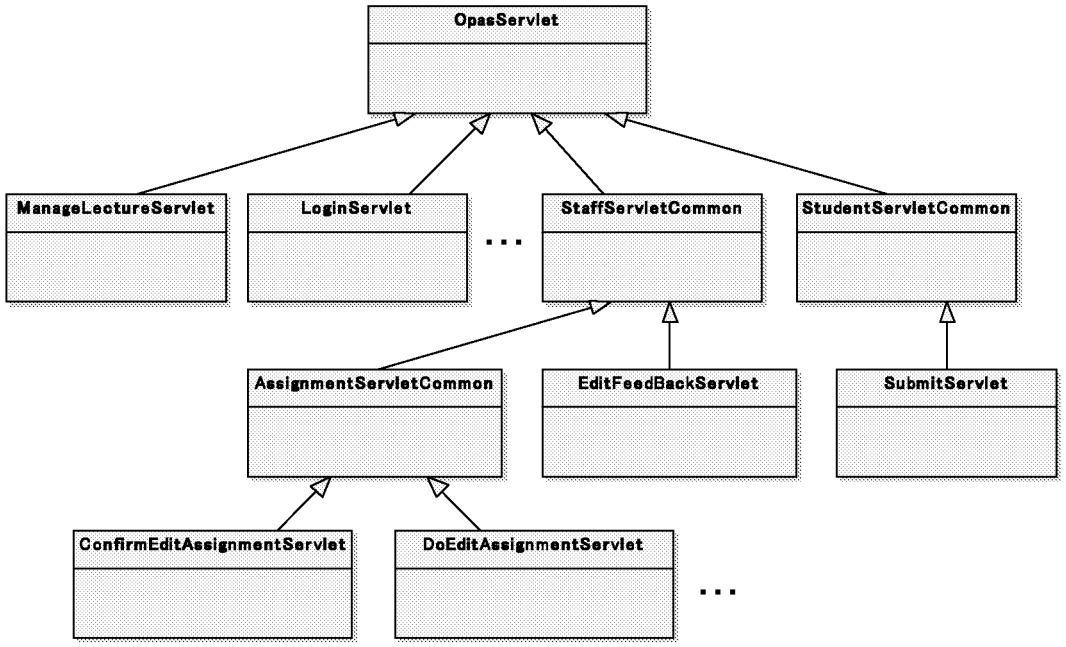



図 3.10: OpasServlet の継承図

は、ユーザに関連する操作である。OPAS のユーザには受講者とスタッフの 2 種類があり、それぞれの権限によって可能な操作が分かれる。そこでサーブレットは、ある操作を実行しようとする際には、ユーザが受講者であるかスタッフであるかを判別しなければならない。この判別などについても多くのサーブレットが利用するものであり、OpasServlet 内で行われているため、継承しているサーブレットは、わざわざ判別をする必要がない。更に、判別だけでなくユーザのログイン、ログアウトについても OpasServlet 内に実装されており、ログインやログアウトの操作は、各サーブレットで定義する必要なくそのまま利用することができる。

これらの、モデルを利用する操作が OpasServlet に実装する設計となっているため、各サーブレットは、容易にモデルにアクセスすることができるようになっている。

#### 3.3.2.2 マッピング

OpasServlet のもう一つの大きな役割としてマッピングがある。3.3.1 節でも触れたように、OPAS の構造では、View と Controller の間の遷移情報を、いかに効率よく管理するかが開発を行う上で大きな問題となる。そこで遷移を管理するための設計に関して、当初 2 案が挙げられた。1 つは Java で Web アプリケーションを開発する際に利用されるフレームワークである Struts（ストラッツ）を利用する案である。Struts を利用することで、その設定ファイルに遷移情報を記述しておき、それに従って JSP に遷移するというものである。もう 1 つが上記 Struts を擬似的に再現し、遷移先の情報をサーブレットと

は別に保持し、その key をサーブレット内で指定するというものである。結果的に後者を選択したが、この理由としては Struts を導入することの手間と、メンバーの小林が規模の大きい Web アプリケーションの開発に不慣れであったことなどから後者を選択することとなった。

この遷移を管理するメソッドが forward() であり、OpasServlet 内で定義してある。これまで記述してきたとおり、OpasServlet はほとんどのサーブレットが継承しており、どのサーブレットでもそのまま利用できるようになっている。そして、遷移先 JSP のパスなどに変更があった際は、サーブレットとは独立している mapping.properties を修正することで、遷移先を変更できる設計となっている。

OpasServlet には以上のような特徴があり、これによって各サーブレットの実装の負担が大幅に削減できた。

## 3.4　提出アプリケーションの設計

(文責:明石　敬)

### 3.4.1　OPAS 提出アプリ

OPAS 提出アプリは受講者が作成した提出物をサーバに提出するためのアプリケーションである。その実装は Java WebStart で起動する Java スタンドアローンアプリケーションである。受講者は Web の課題ページの提出ボタンを押すことで OPAS 提出アプリを起動する。その提出アプリのフォームに解答を入力し送信することで提出物を提出することができる。提出された提出物は OPAS サーバが受け取り、解答ファイルをサーバディスクに保存する。

#### 3.4.1.1　OPAS 提出アプリの設計モデル

OPAS サーバと OPAS 提出アプリはそれぞれユーザーが微妙に違う。OPAS サーバのユーザーは授業のスタッフ、受講者であり、OPAS 提出アプリのユーザーは受講者だけである。そのため OPAS 提出アプリと OPAS サーバでは設計モデルに違いがあると考えた。次に OPAS 提出アプリの設計モデル図を示す。

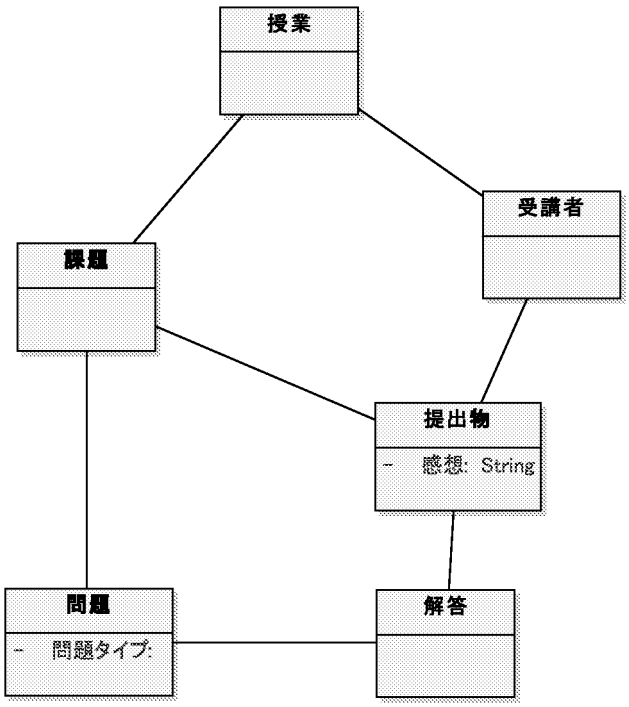



図 3.11: 提出アプリのクラス図

OPAS サーバ側では考えられていたクラスを提出アプリではいくつかなくしている。プログラム問題、記述問題の判別は問題クラスの属性としている。

#### 3.4.1.2　OPAS 提出アプリの処理の流れ

提出アプリの HCP チャートは図 3.12 のように考えた

#### 3.4.1.3　Servlet へ提出物の提出

提出物のディレクトリごと Servlet に提出するために、OPAS 提出アプリは http リクエストをサーバに送らなければならない。http リクエストを利用してファイルをアップロードするためには RFC が定めた規約どおりにリクエストを送る必要がある。(リスト 3.1)

リスト 3.1: 提出物を提出するためのリクエストメッセージ例

```
 1: POST /opas/servlet/opas.servlet.upload/SubmissionServlet HTTP/1.1
 2: Accept: text/txt
 3: Accept-Language: ja
 4: Content-Type: multipart/form-data; boundary=---------------------------123456789
 5: Host: http://etude.crew.sfc.keio.ac.jp:8080
 6: Content-Length:
 7: Connection: Keep-Alive
 8:
 9: -----------------------------123456789
10: Content-Disposition: form-data; name="user"
11:
12: t01021ka
13: -----------------------------123456789
14: Content-Disposition: form-data; name="uploadfile1"; filename="C:\hoge\test.txt"
15: Content-Type: unknown
16:
17:   ・
18:   ・
19: ファイルの中身
20:   ・
21:   ・
22: -----------------------------123456789--
```

### 3.4.2　OPAS 提出アプリケーションサーブレット

OPAS 提出アプリケーションサーブレットでは、提出アプリの起動、提出物の提出に必要な情報の取得、提出物の受け取り、という提出アプリに必要な通信を担っている。

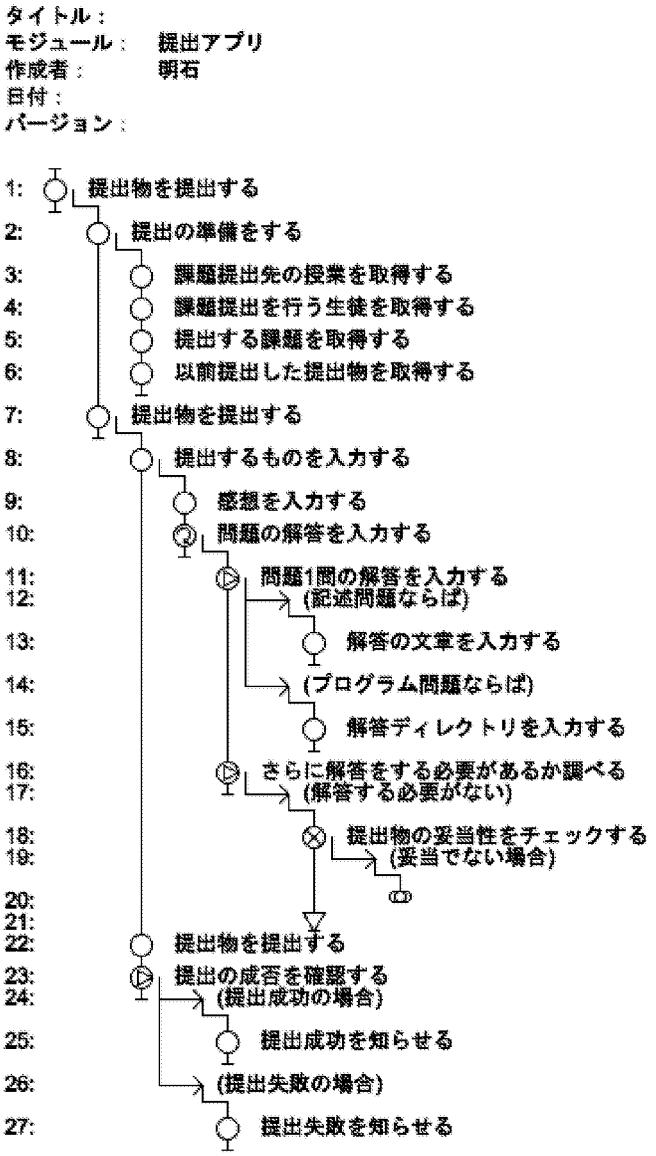



図 3.12: 提出アプリの hcp チャート図

#### 3.4.2.1　提出アプリの起動

OPAS 提出アプリは WebStart で実行される。WebStart を実行するには jnlp ファイルという起動ファイルが必要である。この起動ファイルに必要な情報を書き込むことでアプリケーションは起動することができる。その記述の中でアプリケーションの起動パラメータを設定することも可能である。

OPAS 提出アプリを使って提出物を提出するためには授業情報、受講者情報、課題情報を知る必要がある。これらの情報はサーバにアクセスし取得するのだが、取得するためのキーを提出アプリは起動時に知っていなければならない。OPAS 提出アプリでは起動パラメータにサーバのアドレス、授業コードを最低限必要とし、特に指定する場合があれば、課題 ID、受講者のログイン名、パスワードを設定することができる。課題 ID を指定しない場合は、サーバから最近出された提出可能な課題を自動で取得し、ログイン名、パスワードを指定しない場合は、アプリケーション起動時にログインをさせるよう考えた。

起動パラメータを OPAS 提出アプリに渡すためには jnlp ファイルを動的に生成しなければならない。そのために考えられたのが Servlet にリクエストを送り、Servlet はそのリクエストに基づき jnlp 形式の文字列をレスポンスとして返す方式である。

### 3.4.3　提出アプリとの Http 通信

ユーザーは OPAS 提出アプリを用いて授業で出された課題を提出する。OPAS 提出アプリはユーザーが選択したディレクトリ内のファイルをサーバにアップロードする。ファイルを受け取るサーバプログラムは Servlet で実装するため

OPAS 提出アプリでは提出物を提出するために次のようなサーバとのやりとりをすると考えた。

- 授業情報の取得
- 課題情報の取得
- 受講者情報の取得
- 前回提出した提出物の取得
- 提出物の送信

#### 3.4.3.1　Servlet のレスポンスの形式

やりとりを行う上でのレスポンスの形式を決め、それに基づき通信をするようにした。Servlet のレスポンスの形式には高い拡張性から XML 形式を利用することにした。OPAS 提出アプリはユーザー情報や課題情報などの提出物提出のために必要な情報を取得するために Servlet に http リクエストを送信する。Servlet からは求められた情報を

xml 形式で OPAS 提出アプリに対して返す。提出アプリでは Servlet からの xml 形式のレスポンスをパースしモデルに反映させる。以下に OPAS 提出アプリとサーバ間で行われる通信で用いられるプロトコルを記す。

リスト 3.2: 授業を取得する（サーバからのレスポンス形式）

```
1: <?xml version="1.0" encoding="Shift_JIS"?>
2: <lecture id="lecture_id" code="lecture_code" name="授業名"/>
```

リスト 3.3: 生徒を取得する（サーバからのレスポンス形式）

```
1: <?xml version="1.0" encoding="Shift_JIS"?>
2: <student lectureCode="lecture_code" login="student login" password="password" name="
生徒の名前"/>
```

リスト 3.4: 課題を取得する (サーバからのレスポンス形式)

```
1: <?xml version="1.0" encoding="Shift_JIS"?>
2: <assignment id="assingment_id" code="assignment_code" name="課題名">
3:   <question id="question_id" code="question_code" name="問題名" type="question_type">
問題文</question>
4:   <question id="question_id" code="question_code" name="問題名" type="question_type">
問題文</question>
5:   <question id="question_id" code="question_code" name="問題名" type="question_type">
問題文</question>
6: </assignment>
```

リスト 3.5: 前回提出した提出物を取得する (サーバからのレスポンス形式)

```
1: <?xml version="1.0" encoding="Shift_JIS"?>
2: <submission>
3:   <impression>感想</impression>
4:   <answer questionCode="question_code">ユーザーが入力した文字列</answer>
5:   <answer questionCode="question_code">ユーザーが入力した文字列</answer>
6:   <answer questionCode="question_code">ユーザーが入力した文字列</answer>
7: </submission>
```

リスト 3.6: エラーが発生した場合のサーバからのレスポンス

```
1: <?xml version="1.0" encoding="Shift_JIS"?>
2: <error type="error_type">エラーメッセージ</error>
```

#### 3.4.3.2　提出物の受け取りと保存

提出アプリから提出物を受け取るとサーバのに提出物を保存しなければならない。そのディレクトリ構成は次のようである。図 3.13

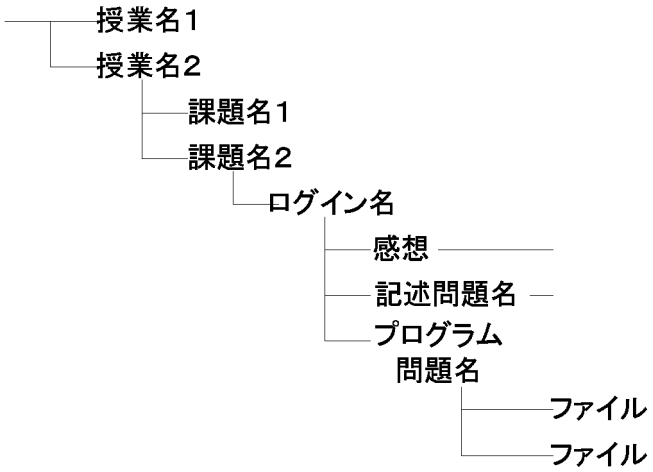



図 3.13: 提出物の保存ディレクトリ構成

# 第 4 章

# 実装

## 4.1 環境の整備

(文責:小林　敦)

この節では、Web アプリケーションの実装に入った後に行った環境整備について記述する。

### 4.1.1 環境整備の経緯

今回、あえて環境整備について触れることとなったのは、実装が始まってからしばらく経過して、Tomcat が起動しなくなったことに起因する。最初は単に server.xml の記述ミス程度に考えていたが、原因と考えられるものを調べても不具合は見あたらない。そこで、Tomcat そのものを再インストールすることとなり、その際に環境の整備をし、全員が統一した環境で開発を行うことにしたのである。

### 4.1.2 環境整備の概要

ここで具体的に統一した環境について挙げておきたいと思う。以下のような環境で統一し、マシンによる不具合を極力減らすよう努めた。

- JDK Version:1.4 以上
- Tomcat Version:4.1.24
- Eclipse Version:2.1.0
- Sysdeo Eclipse Tomcat Launcher

これらの環境をメンバー内で統一することにした。それ以外の開発環境としては、CVS の利用や、テスト用のデータベースの用意などが挙げられる。こうして開発環境を整えてからは、メンバー間での不具合の差が減った。開発環境の整備は実装に入る前の段階で行うべきであることが、改めて感じられた。

## 4.2　モデルの実装

**(文責:岸　健司)**

ここでは、設計モデルから Java コードへマッピングする際、特に注意した点、特徴的な箇所について、明記する。

本システムの Web アプリケーション部分は、モデルに依存している。モデルが動かなければ、Web アプリケーションも動かない。注意深く、実装したつもりである。

### 4.2.1　モデル要素

#### 4.2.1.1　インスタンスのクラス変更

問題クラスはその種類 (プログラム問題と記述問題) が仕様上、変更可能である。つまり、Question クラスのサブクラス ProgramQuestion と DiscriptionQuestion のインスタンスは、そのライフサイクルでクラスチェンジするかもしれないのだ。Java では直接そのようなコードは書けない。

解決として、変更先のクラスの別インスタンスを新たに作り、必要情報をコピーすることにした。ただし、変更前との同一性が取れるように、オブジェクトの ID のコピーや、Java で同一性判定に使われる equals メソッドの書き換えを行った。

この解決方法は、面倒な物であったので、設計を変更して、Question の継承関係を排除して実装すべきだった考える。

#### 4.2.1.2　属性の追加変更

今回の開発で、属性の追加が行われた。開発の進んだ段階での属性追加の影響力は、思いの外大きかった。オブジェクト指向を意識して作っていたので、変更の影響力は小さいと思っていた。しかし、属性が増えるというのは、モデルにとって本質的な変更となるため、実装変更する箇所は多かった。

変更の影響を狭めてくれたのは、多層構造の設計であった。モデルと Web アプリケーションが分離していることで、モデル内だけで変更は済みモデル外にほとんど影響を与えずすんだ。

### 4.2.2　Facade

モデルは、外部からマルチスレッドでアクセスされる。Web アプリケーションがマルチスレッドで動くためである。更新の処理が同時に起こる可能性がある。更新系の処理は同時に処理されないように、モデル群への外部からのアクセス口である Facade のメソッドは同期化させた (更新系メソッドは同時に 1 つしか実行できないようにロック)。

### 4.2.3 永続化

#### 4.2.3.1 Torque フレームワークによる変更容易性

Torque フレームワーク (今回は Torque3.1) を利用したことで, 変更容易性が感じられた。

Torque は永続化関連のコードを自動生成する。自動生成の手順を簡単に表現すると次のとおりである。

- データベースのスキーマを XML で記述する（今回は/torque-gen-3.1/schema/opas-schema.xml）
- Torque ジェネレータの設定をする (build.properties)
- Torque ジェネレータのビルド (build-torque.xml) を実行する (main,create-db,id-table-init-sql,insert-sql の順に)

これによって、Torque ジェネレータの src フォルダにテーブル定義の SQL 及びレコードに対応したクラスの Java コードが生成されるまた、新規 DB やテーブルの登録、ID 生成の準備まで行う。初めは設定が面倒であると感じたが、後にスキーマ変更時に利便性を感じた。スキーマの XML の必要箇所だけ変更して、再ビルドするだけでよかったのだ。Java コードを 1 行も変更する必要がなかった。

また、対象データベースをローカルホストの MySQL から、リモートホストの PostgreSQL へ変更する場面でも、設定ファイルを変更するだけですんだ。

### 4.2.4 運用環境への配慮

開発環境と運用環境は別物である。したがって、開発環境に依存しないように作らなければいけない。以前のプロジェクトみるみるの経験から、ファイルパスやアクセス権限には注意して開発した。Windows で開発していると、ついファイルへのアクセス権限を忘れがちである。データやログを保存する箇所のアクセス権限やカレントディレクトリにも気をつけなければいけない。今回、これらは問題は発生しなかった。しかしながら、テキストファイルの文字コードの問題は発生した。いくら開発環境で動いても、運用環境で正常に動かなかったら、どうしようもない。目先の動く、動かないでなく、しっかりと先を見据えた開発が重要である。

### 4.2.5 ログ出力

問題発生時に、追跡調査やデータ復元が行えるように、更新処理にはログ出力を付加してある。実用システムには必要な要素であろう。

### 4.2.6　共同開発への配慮

モデルは他の多くのモジュールから頻繁に使われる部分であった。モデルが動かなければ、Web アプリケーション部分も動かない。なるべく、モデルのエラーによって他の開発を妨げないように気遣った。

特に気を遣ったのが、永続化を RDB ベースに移行させるときである。頻繁にテストをするだけでなく、元々のシリアライズ化ベースの永続化コードと RDB ベースの永続化コードを両立させた。そして、設定ファイルを変更するだけで、永続化方法を切り替えられる様にした。このことによって、映像化方法を一斉に切り替えられたし、問題があったときにすぐさま元に戻せる安心感も得られた。

また別の工夫として、モデルのインターフェースの変更をするとき、以前のメソッドには破棄予定の印を付け、しばらくの残したあとに、削除する方針をとった。これによって、共通で使っている eclipse という開発ツールの機能を通して、他の開発者に破棄予定のメソッドを使用しているかが伝わる。そして、気づいたときに直してくれるということが可能になった。

また他に、問題の切り分けがしやすいように、モデル群への入り口である Facade のメソッドでは事前条件を表明する様にしたり、本システム独自の例外 OpasException を用意するなどもした。

### 4.2.7　デバッグ

今回、特に、Facade クラスには単体テストを当初から用意した。もっとも、テスト項目はすべての公開メソッドに対して、思いつく様々な状況を並べただけで、機能の正当性を十分保証する物ではないが、それでも、デバッグに大いに役立った。

JUnit というツールを用いて、テストを用意したが、ある程度変更する度に、単体テストを実行することにした。このことで、ある程度の有効性を、いつも検証できた。

Facade は他のメンバーが開発したコードに頻繁に使われるプログラムであり、Facade コードが信頼できないと、他のコードの開発にも大きな影響が出る。単体テストによる検証は、精神的な安心感も与えてくれた。

また、単体テストは、該当モジュールの持っている全機能の使用例でもあり、他の開発者に使い方を教える面でも役に立つ。

## 4.3 Web アプリケーションの実装

(文責:小林　敦)

この節では、Web アプリケーションの設計に従って実装をしていく上での経過について触れていく。

### 4.3.1 各部実装の分担

Web アプリケーションを実装するにあたって、各メンバーの実装担当を決める必要があった。この時点で、明石は提出アプリケーションの開発を担当しており、主に実装にあたることができたのは、岸、小林の 2 名であった。Web アプリケーション自体の実装は、この 2 名を中心に行っていった。そこで各メンバーの担当を明確にするために、遷移表を利用した。図 4.1 が中間発表までに実装した範囲の遷移表である。ここまでで、提出アプリケーションが仮完成しており、また課題の出題のみ対応していた。

| 画面 | 遷移 | タイプ | フォーム | 遷移先 | 遷移先JSP |
|---|---|---|---|---|---|
| login.jsp | LoginServlet | フォーム | login,passwd | toStaffTop | staffTop.jsp |
| | | | | toStudentTop | studetTop.jsp |
| | | | | loginfailure | error.jsp |
| | MappingServlet | リンク | – | toRegistStudent | registStudent.jsp |
| staffTop.jsp | RegistAssignmentServlet | リンク | – | toEditAssignment | editAssiginment.jsp |
| | ModifyAssignmentServlet | フォーム | assignmentID | toEditAssignment | editAssiginment.jsp |
| | RemoveAssignmentServlet | フォーム | assignmentID | toConfirmRemoveAssignment | confirmRemoveAssignment.jsp |
| | | | | toRejectRemoveAssignment | error.jsp |
| editAssignment.jsp | MappingServlet | リンク | – | toStaffTop | staffTop.jsp |
| | ConfirmEditAssignmentServlet | フォーム | たくさん | toConfirmEditAssignment | confirmEditAssignment.jsp |
| | | | | toInvalidEditParameter | error.jsp |
| confirmEditAssignment.jsp | DoEditAssignmentServlet | リンク | – | toStaffTop | staffTop.jsp |
| confirmRemoveAssiginment.jsp | DoRemoveAssignmentServlet | リンク | – | toStaffTop | staffTop.jsp |
| rejectRemoveAssingment.jsp | ConfirmEditAssignmentServlet | リンク | – | toStaffTop | staffTop.jsp |
| studetTop.jsp | MappingServlet | リンク | assignmentID | toSubmit | submit.jsp |
| registStudent.jsp | ConfirmRegistStudentServlet | フォーム | たくさん | toConfirmRegistStudent | confirmRegistStudent.jsp |
| | | | | toInvalidEditParameter | error.jsp |

図 4.1: 遷移表（中間発表時）

中間発表後、更に必要となると考えられる遷移を加え、その後の担当なども追加した遷移表を追加した図 4.2 を作成した。その後の開発については、この表を中心に作成していった。この表は自分の担当部分がよく分かり、実装の面で大きな意味があったといえる。

### 4.3.2 実装上の抽象クラスのメリット

Web アプリケーションの設計の節でも触れたように、OPAS では OpasServlet という抽象クラスを利用している。この抽象クラスを継承した各サーブレットの実装はについて述べていく。一言でいってしまうと全体として、大きな問題は発生しなかった。これは OpasServlet の設計がよかったことが大きかったと考えられる。私は今回のような規模の大きいプロジェクトに参加した経験がなかったため、比較対象がなく、あまり感じられなかったが、OpasServlet の設計は大変良かったようである。

| | 画面 | 説明 | 遷移 | タイプ | フォーム | 遷移先 | 遷移先JSP | 現状 | 権限 |
|---|---|---|---|---|---|---|---|---|---|
| → | login.jsp | ログイン | LoginServlet | フォーム | login.passwd | toStaffTop | staffTop.jsp | ○ | スタッフ |
| | | | | | | toStudentTop | studetTop.jsp | ○ | |
| | | | | | | loginFailure | error.jsp | ○ | |
| | | | MappingServlet | リンク | – | toRegistStudent | registStudent.jsp | ○ | |
| | | | RegistAssignmentServlet | リンク | – | toEditAssignment | editAssignment.jsp | ○ | スタッフ |
| | | | ModifyAssignmentServlet | フォーム | assignmentId | toEditAssignment | editAssignment.jsp | ○ | スタッフ |
| | | | RemoveAssignmentServlet | フォーム | assignmentId | toConfirmRemoveAssignment | confirmRemoveAssignment.jsp | | スタッフ |
| | staffTop.jsp | スタッフトップ | | | | toRejectRemoveAssignment | error.jsp | | スタッフ |
| | | | ShowAssignmentServlet | フォーム | assignmentId | toShowAssignment | showAssignment.jsp | | スタッフ |
| | | | ShowImpressionsServlet | フォーム | assignmentId | toShowImpressions | showImpressionList.jsp | | スタッフ |
| | | | ShowSubmissionsServlet | フォーム | assignmentId | toShowSubmissions | showSubmissionList.jsp | | スタッフ |
| | | | UserTopServlet | リンク | – | toStaffTop | staffTop.jsp | △ | スタッフ |
| | editAssignment.jsp | 課題編集 | ConfirmEditAssignmentServlet | フォーム | たくさん | toConfirmEditAssignment | confirmEditAssignment.jsp | ○ | スタッフ |
| | | | | | | toInvalidParameter | error.jsp | ○ | スタッフ |
| | confirmEditAssignment.jsp | 課題編集確認 | DoEditAssignmentServlet | リンク | – | toStaffTop | staffTop.jsp | ○ | スタッフ |
| | | | | | | toEditAssignmentError | error.jsp | | |
| | confirmRemoveAssignment.jsp | 課題削除確認 | DoRemoveAssignmentServlet | リンク | – | toStaffTop | staffTop.jsp | | スタッフ |
| | | | | | | toRejectRemoveAssignment | error.jsp | | |
| | | | DownloadJnlpServlet | リンク | assignmentId | toGetJnlp | upload/getJnlp.jsp | △ | 受講者 |
| | | | ShowAssignmentServlet | フォーム | assignmentId | toShowAssignment | showAssignment.jsp | | 受講者 |
| | studetTop.jsp | 受講者トップ | ShowImpressionsServlet | フォーム | assignmentId | toShowImpressions | showImpressionList.jsp | | 受講者 |
| | | | ShowSubmissionsServlet | フォーム | assignmentId | toShowSubmissions | showSubmissionList.jsp | | 受講者 |
| | registStudent.jsp | 受講者登録 | ConfirmRegistStudentServlet | フォーム | たくさん | toConfirmRegistStudent | confirmRegistStudent.jsp | ○ | 受講者 |
| | | | | | | toInvalidEditParameter | error.jsp | ○ | 受講者 |
| | showSubmission.jsp | 提出物表示 | | | | | | 岸さん | |
| * | upload/getJnlp.jsp | JAVA WEBスタート 起動 | | | | | | | |
| * | showAssignment.jsp | 課題内容 | ModifyAssignmentServlet | フォーム | assignmentId | toEditAssignment | editAssignment.jsp | 岸さん | スタッフ |
| | | | DownloadJnlpServlet | リンク | assignmentId | toGetJnlp | upload/getJnlp.jsp | △ | 受講者 |
| | | | UserTopServlet | リンク | – | toStaffTop /toStudentTop | staffTop.jsp / studentTop.jsp | | |
| | showImpressionList.jsp | 感想一覧 | ShowSubmissionServlet | フォーム | submissionId | toShowSubmission | showSubmission.jsp | 小林 | |
| | | | UserTopServlet | リンク | – | toStaffTop /toStudentTop | staffTop.jsp / studentTop.jsp | | |
| | showSubmissionList.jsp | 提出物一覧 | ShowSubmissionServlet | フォーム | submissionId | toShowSubmission | showSubmission.jsp | 明石くん | |
| | | | UserTopServlet | リンク | – | toStaffTop /toStudentTop | staffTop.jsp / studentTop.jsp | | |
| → | manageLecture.jsp | 授業管理 | EditLectureServlet | フォーム | lectureCode | toEditLecture | editLecture.jsp | △ | |
| | | | MappingServlet | リンク | – | toRegistLecture | registLecture.jsp | △ | |
| | editLecture.jsp | 授業編集 | ConfirmEditLectureServlet | フォーム | たくさん | toConfirmEditLecture | confirmEditLecture.jsp | △ | |
| | | | | | | toInvalidParameter | error.jsp | | |
| | confirmEditLecture.jsp | 授業編集確認 | DoEditLectureServlet | フォーム | – | toEditLectureTop | editLectureTop.jsp | | |
| | registLecture.jsp | 授業登録 | ConfirmRegistLectureServlet | フォーム | たくさん | toConfirmRegistLecture | confirmRegistLecture.jsp | △ | |
| | | | | | | toInvalidParameter | error.jsp | | |
| | confirmRegistLecture.jsp | 授業登録確認 | DoRegistLectureServlet | フォーム | – | toEditLectureTop | editLectureTop.jsp | △ | |
| | error.jsp | エラー | | | | | | | |
| | exception.jsp | 例外 | | | | | | | |

※ログイン後はセッション切れからerror.jspへの遷移がある

図 4.2: 遷移表（決定）

ではこの抽象クラスを使うことのメリットとはどういったものなのだろうか。これは、主に 2 つの効果が考えられる。1 つ目は、実装の高速化が挙げられる。Web アプリケーションの設計でも触れたように、モデルにアクセスする際に必要となる Facade のインスタンス生成の省略などにより、実装がより素早くできる。2 つ目に変更の容易さが挙げられる。今回の開発の範囲では、仕様変更などは発生しなかったが、もし、各サーブレットで共通する操作に変更が必要となったとき、抽象クラスを利用していない場合は、各サーブレットを修正していかなければならない。これが、この変更される共通の操作が抽象クラスに含まれていた場合は、この部分だけを変更すれば、継承する各サーブレットを修正する必要がない。抽象クラスの継承には、このようなメリットがあると考えられる。

しかし、抽象クラスが良ければ問題が全く起こらないか、というとそうではなく、小さな問題は発生していた。以下の節でそれらの問題とその解決について触れていく。

### 4.3.3 実装上の諸問題

上記のように、サーブレット・JSP の実装において、小さな問題はいくつか発生した。以下にその例を示したいと思う。

- etude における文字コードの問題
- 不正な入力に対するエスケープの問題
- request と session を取り違える問題
- 「戻る」が未配置であるなどのインターフェイスの問題

上記のような問題が発生した。これらはそれぞれ小さな問題ではあるが、しっかりと対応しようとすると労力の掛かる問題でもある。実装に大きな問題がなかった分、これらの小さな問題に対応する時間が作れたことが幸いであったと思う。

## 4.4　提出アプリケーションの実装

**(文責:明石　敬)**

この節では提出アプリの設計から実装する際に起こった問題、工夫した点を述べる。

### 4.4.1　Servlet との http 通信

提出アプリから提出物を Servlet に送信する際、設計の段階では考え切れていなかった部分が発見された。

#### 4.4.1.1　サブディレクトリの対応

実装していくうちにサブディレクトリの問題が浮き上がった。提出されたディレクトリの中にディレクトリがあるとしたら、それは提出すべきか。議論した結果、サブディレクトリも提出することになった。

しかし、設計の段階で考えた提出物送信リクエストでは、サブディレクトリの中までは送信することができず、リクエストの形式を変更する必要があった。

結果、図 4.3 のようなリクエストの形式になった。サブディレクトリの始まりと終わりを知らせることで、サブディレクトリの中に含まれるファイルと含まれないファイルを区別する。”directory”パラメータでディレクトリ名を設定し、”directoryend”パラメータまでの間をすべてサブディレクトリ内とした。この方法を利用することでディレクトリの下のディレクトリ、さらにその下のディレクトリ、さらにその下の・・・と無限に最下層のディレクトリまで読めるようになった。

### 4.4.2　新たな問題と提出されるファイルへの制限

図 4.3 のようなリクエスト形式により無限に最下層のディレクトリを読み込めるようになった。しかし、新たな問題に気づかされた。大量のファイルをリクエストに乗せを送りつけられても許可をしてしまうことになってしまっていた。サーバーに大量のファイルを書き込めるようにしてしまったのである。さらに、ディレクトリ内のファイルはすべて送信してしまうようになっていたので、当然、無駄な画像ファイルや音楽ファイルなどもサーバーに書き込めるようになってしまっていた。

次のような解決策があった。

- 受け取るファイルの容量を制限する

```
POST /opas/servlet/opas.servlet.upload/SubmissionServlet HTTP/1.1
Accept: text/txt
Accept-Language: ja
Content-Type: multipart/form-data; boundary=---------------------------123456789
Host: http://etude.crew.sfc.keio.ac.jp:8080
Content-Length:
Connection: Keep-Alive

-----------------------------123456789
Content-Disposition: form-data; name="user"

tO1O21ka
-----------------------------123456789
Content-Disposition: form-data; name="uploadfile1"; filename="C:\hoge\test.txt"
Content-Type: unknown

  .
  .
ファイルの中身
  .
  .
-----------------------------123456789
Content-Disposition: form-data; name="uploadfile2"; filename="C:\hoge\test.txt"
Content-Type: unknown

  .
  .
ファイルの中身
  .
  .
-----------------------------123456789
Content-Disposition: form-data; name="directory"

sub
-----------------------------123456789
Content-Disposition: form-data; name="uploadfile3"; filename="C:\hoge\sub\test3.txt"
Content-Type: unknown

  .
  .
ファイルの中身
  .
  .
-----------------------------123456789
Content-Disposition: form-data; name="uploadfile3"; filename="C:\hoge\sub\test4.txt"
Content-Type: unknown

  .
  .
ファイルの中身
  .
  .
-----------------------------123456789
Content-Disposition: form-data; name="directoryend"

true
-----------------------------123456789--
```

図 4.3: 提出物を提出するためのリクエストメッセージ例 (サブディレクトリ対応版)

- 送信するファイルの種類を限定する

上のようにすることで上記の問題は解決することができた。デフォルトでは 1 メガバイトまでのファイルを受け取り、java ファイルと txt ファイルを送信できるようになっている。

### 4.4.3　ブラウザからの起動

web ページから OPAS 提出アプリを実行する際、サーブレットを利用することでインタラクティブに jnlp ファイルを生成することができた。しかし、それでも Java WebStart はうまく起動しなかった。そのわけは「content-type」である。最終的な jnlp ファイルの生成は jsp にまかせていたが、jsp はデフォルトで「content-type」を「text/html」または「text/plain」となる。これではせっかく生成された jnlp ファイルを WebStart の起動ファイルと認識せず、ただのテキストと判断されてしまうため、ブラウザに jnlp ファイルの中身が表示されてしまうだけだった。

これを回避する方法として図 4.4 のような記述をし，「content-type」を「application/x-java-jnlp-file」と設定しなおしている。こうすることでブラウザは WebStart の起動ファイルであると認識し、無事 WebStart を起動することができる。

```
//webstart として起動させる
response.setHeader("Content-type","application/x-java-jnlp-file");
```

図 4.4: WebStart の起動のための記述

### 4.4.4　セキュリティ制限

Java WebStart、Java Applet には Web からアプリケーションプログラムをダウンロードし実行させるため、不正にデータを読み取ったり、不正なデータを書き込んだりされることを避けるためにローカルファイルへのアクセスが禁止されていた。そのためユーザーのローカルディレクトリにアクセスをし、ユーザーのディレクトリごとアップロードすることができない。これらのセキュリティ制限をはずすためにデジタル署名証を発行する方法がある。プログラムの作成者がユーザーにプログラムの安全性を示し、ユーザーがプログラムの無制限のアクセスを承諾することで、プログラムはユーザーのローカル環境にアクセスすることができる。OPAS 提出アプリではデジタル署名証を発行し、プログラムの安全性を証明することでユーザーのローカルディスクにアクセスすることにした。

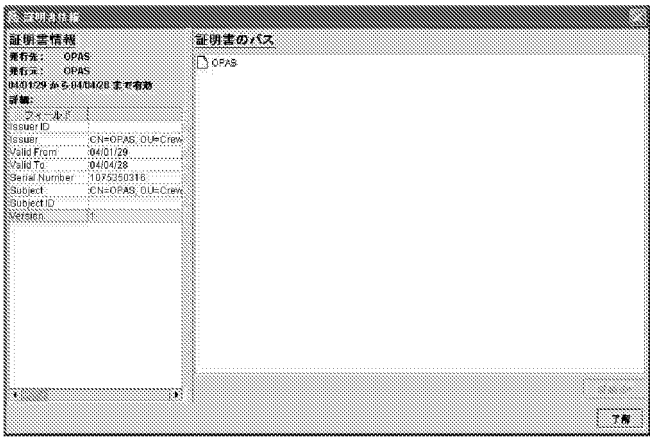



図 4.5: 発行された署名証

### 4.4.5 文字コード問題

はじめ、各自のローカル環境で開発を行っていた際、文字コードという問題はまったく起こらなかった。どうやら windows デフォルトの Shift_JIS をアプリ、サーバとも使用していたようである。しかし、テストサーバで実行した際、問題が起こった。文字化けが起こったのである。提出物を提出する際、提出物入力フォームには、以前入力した解答がセットされており、以前提出した提出物を知ることができるが、文字化け問題のせいでまったく読むことができなかった。これは、テストサーバの OS が linux であり、そのデフォルトの文字コードが Shift_JIS ではなく EUC-JP であったためである。

この問題が発生してから、サーバと提出アプリ間の文字コードを統一することを考えた。そこで、サーバ、提出アプリともに文字コードを EUC-JP にした。OPAS で利用している DB の文字コードが EUC-JP であるため、これにあわせるのが妥当であろうと考えたからである。

結果、文字化け問題は解決され無事以前入力した提出物を知ることができた。

### 4.4.6 GUI

提出アプリで一番苦労したのは GUI の実装である。

swing は java の GUI として一番普及している。今回選択した swing は、さまざまな機能をとりそろえた GUI コンポーネント群としてはば広く注目されたが、swing 以前の GUI コンポーネント awt との強い互換性を維持するためにきれいとは言えない設計がなされ、その扱いが非常に難しいものとなった。(awt は java 登場時に GUI コンポーネントが熱烈に望まれ、それに応えるべく慌てて作ってしまったためあまりよい設計とは言えないものになってしまった。)

提出アプリは swing で実装したのに加え、作成者明石が GUI に不得手なためあまりい

い実装はしていない。GUI の実装が遅れてしまったために他メンバーに迷惑をかけることもたびたびあった。

#### 4.4.6.1　マルチプラットフォームの GUI

提出アプリは Java で実装されているため、マルチプラットフォームで実行されることを想定している。しかし、その GUI はすべての環境で同一というものではなかった。微妙に Java の VM によって GUI の配置が違うようである。

また、swing のデフォルトのルック＆フィールの「metal」は非常にメンバーに不評だったため、それぞれの OS のデフォルトのルック＆フィールが動くように設定した。これにより、windows では windows ライクな GUI が起動し、Mac では「Aqua」と呼ばれる Mac 独自のルック＆フィールで起動するようになった。

#### 4.4.6.2　レイアウト

GUI の中で特に苦戦したのがレイアウトである。作成者明石はもともとレイアウトセンスに欠けるわけだが、それでも果敢に自分の思うようにレイアウトをしてみた。

swing には、「GridBagLayout」という自由度の高いレイアウトができるレイアウトマネージャがある。しかし、自由度の高さの反面、まったく扱いづらいものだ。細かい設定が可能な割には配置するコンポーネントによって挙動が違い、あまり自分の思い通りに配置することがうまくいかない。そこで、しょうがなく配置するコンポーネントのサイズを指定する方法をとってみた。こうすることでコンポーネントの大きさは一定のものとなり、コンポーネントの配置がしやすくなった。しかし、大きさを一定とすることは、その反面、インタラクティブにコンポーネントの大きさがかわる利便性を失ってしまった。さらに調査をしなんとか解決していきたい問題である。

# 第 5 章

# 評価

## 5.1　機能テスト

**(文責:岸　健司)**

開発したシステムは、評価する必要がある。今回、我々は開発関係者間でテストを行った。その内容をこの節では示す。残念ながら、上の都合で実際の授業などでユーザテストをすることは出なかった。本当に使えるシステムとなるには将来的に、エンドユーザの評価を受ける必要があるだろう。また、性能評価も必要となるだろう。

### 5.1.1　ロールプレイング

ユーザテストは、期間及び各人の役を決めて行った。役は次の通りである

- 先生スタッフ役：明石
- 普通受講生：岸
- 遅刻提出受講生：小林、青山

また、日時と実施内容は次の通りである。

- 1/27：メールで課題出題 (明石)
- 1/27：課題提出 (岸)
- 1/28 朝:遅延提出（小林）
- 1/28 昼：課題解答公開 (明石)
- 1/28 夜：課題受付中止 (明石)
- 1/28 受付中止後:遅延提出（青山）

バグを見つけるのがねらいであるが、実際に役割を決めて、それぞれの立場になって演じることで、使いやすさも判定も同時におこなおうというものであった。

### 5.1.2　テストの成果

得られたフィードバックをリスト 5.1 に示す。

リスト 5.1: 機能テスト

```
*ユーザ登録
**「登録してよろしいですか？」で OK しかない。ブラウザ戻るでないと戻れない。
→意味的に微妙かも。
**名前にタグが使えます。

*課題表示画面
**エスケープ文字が化けている。きしが間違って２重エスケープしたかも

*提出関連
**プログラム提出の確認画面で Root だけ表示されるがわかりづらい
(ディレクトリ空っぽ (新ためて提出しないの反応)、存在しないディレクトり)
**確認画面では感想や記述解答の改行がされていない
**感想に改行が含めない

**そういえば、部分再提出（変更する物としない物）ってできなくて良かったん
だっけ
(提出したものとしないものの区別がはっきりつくようなインタフェースにする旨を大岩先生がずいぶん前に
言っていたような)

**おまけ…提出ファイル制限があるといいかも Opas 全部挙げてみたけど…大変な
ことに。
**プログラム問題は未提出にできるが、記述問題を未提出にできない？
**感想未提出ができない


*感想一覧
**名前と内容のエスケープができていない
**名前から各人提出物へのリンクあるが、非公開中にでリンクがあり間違って押
してしまう。(とどってもみれないが)

*スタッフ関連
　・課題登録後のエラー表示で「ok」を押すと編集画面に戻るが、入力した値がすべて消えてしまっている。
残ってるとうれし
　・課題登録確認画面で戻るを押すと上と同じ現象
　・課題を新しく登録したが、採点の欄に「済 81/81」と出ていた。
　・課題を削除で OK を押す。削除後にブラウザの戻るで戻り、もう一度削除で OK を押すとえくせぷしょん画
面が・・・　当たり
　・同じく課題登録時にも登録して戻って OK をするとぬるぽ。
```

問題がいくつも発見され、これらの中で、致命的な問題はすぐに修正した。

### 5.1.3 テスト時の問題

今回のテストで１つ問題になったのは、テスト期間中に修正したコードをサーバに上げ直してしまったことだ。このとき、サーバへの配置に失敗し、一時システムの利用に支障が出た。テストの最中にテストしているコードが変わってしまうと、状況が複雑化してしまい、評価や後のデバッグ時の原因追及がしづらくなってしまう可能性がある。テスト中はテスト対象を変更してはいけない。テスト前に禁止事項として確認しておくべきだった。これが今回の教訓である。

# 第 6 章

# プロジェクトの運営

## 6.1 岸　健司

ここでは、本プロジェクトの経験を通して、学んだこと、感じたことを示す。

### 6.1.1 共同作業の難しさ

共同開発では他の開発者への配慮が必要で、手間がかかる。

今回、私は設計や実装においては、アプリケーション全体の基礎になるモデル部分の多くを担当した。この部分に変更やエラーがあれば、同時に進行する他のメンバーの別部分の開発に影響が出る可能性があった。開発での注意点として、

- 他メンバーにモジュールを提供する前に、主要部分に単体テストを行う
- 実装や設計が大幅にされても極力、外部インターフェースは変えない
- 大幅な設計・実装変更のスムーズ化 (リレーショナルデータベースへの移行の際、一斉に移行でき、不具合時に備えすぐに元に戻せる設計にした)

などを実践することで、大きな問題なく、開発を進めることに貢献した。ただ、このような開発への配慮を実践することは、各所で工夫を必要として、多くの労力が必要であった。このことから、改めて、共同開発の大変さを実感した。

また、「手間がかかっても、不必要なものは、使えないするべきである」という教訓を得た。今回、テストを済みと思っていた機能を利用した外部モジュールの問題が 2 件発生した。

原因はどちらも、外部から利用する必要のない (利用を想定していない) インターフェースが利用されていることだった。利用を想定していないので、テストが不十分であったり、誤った使い方がされていた。不必要な情報や操作へは、アクセスできないようにする。オブジェクト指向のカプセル化がしっかり実装されていれば、未然に防げた問題である。しかし、実装の際に、熟考しなかったり、手間を惜しんだことによりこれらの問題は起こってしまった。

### 6.1.2　新しいツール利用の難しさ

ツールはマニュアルが整備されていないと使えない。

今回オブジェクトとリレーショナルデータベースとのにマッピングする Torque というフレームワークを使用した。これは、単純なコーティングの作業を大幅に削減してくれたが、その使い方習得に多くの時間を費やしたのも事実である。情報が少なく、マニュアルも完備されているわけではなかったからだ。サポートサイトにも説明が省かれている設定項目があったし、動作の表面的な説明はあっても詳細が説明されてないものもあった。今回は、ソースコードの一部の読解や、動作実験することで対応しできた。

新しいツールを使うには、その概要を学ぶ必要があるが、マニュアルが整備されていないことも多いだろう。幸い、Torque はある程度の認知度の認知度とオープンソースという特徴を持っていたため、何とか開発の大きな支障が出ずにすんだ。しかし、いつもこういった状況とは限らない。ツール自体にバグがある場合もあるだろう。

開発において、新しいツールを使うの際は、きちんと使える物であるのか、十分な検討がすることが重要だと感じた。

また、自分たちが開発物自体も、開発者以外が使える物にすることも重要だろう。

残念なことに、現時点では、開発メンバー以外が本システムを使う場合のマニュアルが整備されていない。本システムが使えるシステムになるためには、もエンドユーザ向けのマニュアルが必要である。

### 6.1.3　教育と開発規模

本プロジェクトのメンバーの小林敦は開発経験の少なかった。彼は、１つ１つの知識や技術を学びながら、開発を進める必要があった。私は、彼にツールの使い方、図の読み書き、実装の仕方、技術知識など、必要あるいは彼が興味を示したことを数多く教えた。

しかし、開発物の規模が大きく、開発の作業自体に大きな時間をかかかり続けたため、我々は教育に対して大きな時間は割けなかった。彼に対して、本プロジェクトを通して教えるべきだが、教えられなかった事柄がたくさんあっただろうと感じる。

教え込むのに時間がかかる作業を彼にあまり任せなかった傾向もあり、開発を進めるなかで、未経験の部分も彼には多くの残っている。

また、本プロジェクトは今期、ユーザテストまではたどり着かず、システムの運用についてはプロジェクトメンバーは経験できず、学ぶことができなかった。

これらのことは非常に残念なことである。

私は、研究会で開発するアプリケーションの規模は、「学ぶ」という点では本プロジェクトより小さなものの方が適していると思う。

### 6.1.4 プロジェクトの運営体制

#### 6.1.4.1 プロジェクトマネージャの存在と甘え

私は、同研究会で前学期に「みるみる」というシステム開発プロジェクトのメンバーであった。そこで、松澤氏はロジェクトのメンター的存在 (兼システムオーナー) となり、プロジェクトを成功へ導いてくれた (詳しくは GreW「みるみるができるまで」阿部雅美, 川勝名奈恵, 岸健司 著 を参照)。今回はこれと似た存在として、プロジェクトマネージャ (PM) が各プロジェクトに配属された。

「みるみる」の松澤氏と本プロジェクトの PM のプロジェクトメンバーから見た違いの１つは、プロジェクトの外側の人間であるか内側の人間であるかである。

両者を比較したとき、PM とメンバーは長い時間を共有でき、より緊密に、より協力して物事ができた。

しかし、一方で、私は、自ら進んで考えずに、PM に対して、過大な期待をもったり、判断を頼ろうとしてしている自分の姿に何度気がついた。何でも PM に依存しようとしていたのである。

PM はマネージメントが仕事である。実際のプロジェクトはメンバー自身が自立して前進させるべきだろう。

「みるみる」プロジェクトに比べ、本プロジェクトは、企画や仕様が揺れ動き、なかなか決まらなかった。「みるみる」ではシステムのオーナーが定まっていた。オーナーの意見が優先度が高く、物事を決定しやすかった。今回は、オーナーの存在があいまいであった。これが、企画や仕様の揺れ動きにつながったと思う。開発プロジェクトにおいては、オーナー的な存在をしっかりと決めるとスムーズに進められるように感じた。

### 6.1.5 記録に残すこと

私は、プロジェクトでの作成資料を Web サイトで管理することを提案した。Web サイトはだれもがページを書き換え可能な「Wiki」を資料管理に便利なように改変したシステムを用意した。

Web での資料管理は、私が、「みるみる」プロジェクトにおいて取り入れ、その資料閲覧の容易性を実感していた。

今回は Wiki の利用で負担の大きい資料のアップロード係を設けずに、各作成者が自分の資料を説明とともにアップロードすることで、管理の手間を省こうという意図を持っていた。

当初、この方法はうまく機能し、好評であった。しかし、作成資料のすべてが Web サイトで共有されているかどうかチェックする人が存在していなかったため、プロジェクト後半になるにつれ、参考資料がアップロードされないなどの事態が生じた。資料共有はプロジェクトを円滑に進める上で不可欠である。そして、資料共有化のチェック機構は重要

である。

記録を残すことは、将来の役に立つ。今回各メンバーは作業ログをつけたが、自分の作業計画を考える上での参考資料になった。また、「みるみる」プロジェクトのドキュメントが今期の研究会の様々なプロジェクトで参考資料として使われている姿を何度も目にした。以前のプロジェクトの資料がその後も使われていたのである。この経験から、記録を残すことの有用性を実感した。

## 6.2 明石 敬

今期の研究会での開発を通して、思ったこと学んだこと感じたことをこの節では述べる。

### 6.2.1 OPAS

OPAS は来期のオブジェクトプログラミングで使われるという前提で動き出したプロジェクトだ。はじめの仕様作成の段階では、採点がしやすくなる、という目標でプロジェクトが動いていたのだが、数々のご指摘をいただき、プログラミング初心者が多くの他人のソースコードに触れる機会を得ることによりソースコードの可読性の重要性を実感できるようになる、という目的に変わった。

OPAS の開発に自分は不安を抱いていた。来期使われるというプレッシャーがあること、そして、前期開発した「IRC」が実用にあたって無念の「作り直し」という状態になってしまったことがある。前期は自分の力不足を痛感した。今期の開発では休み中に「作り直し」にならないようにするにはどうすればよいか、また、自分が作ったアプリケーションが実用に値するものになるのだろうか、その前にちゃんと完成するのだろうか、とネガティブ思考になってしまっていた。

結果、OAPS はユースケースを満たし、大まかな仕様の通りの実装が完了した。ものが出来上がったという点では非常に満足している。研究会の最後のプレゼンで、「やっぱり無理でした」という情けないことにはならなかった。これはＰＭとプロジェクトメンバーのおかげであると常々思う。

### 6.2.2 スタンドアローン

今回の開発も主に Web アプリケーションのグループ開発だったのだが、自分の担当した部分は提出アプリ部分であった。提出アプリは WebStart で起動するわけだが、実際の中身は Java のただのスタンドアローンアプリケーションと変わりなかった。その提出アプリの実装とともに提出アプリ関連のサーブレットも実装した。提出アプリ関連の実装は一人でやることになり、あまりプロジェクトメンバーとの共同作業がなかった気がする。久々の個人開発だったが、マイペースに自分の速度でゆったりと実装ができる予定だった。しかし、実際はゆったりすぎて他のメンバーとの進捗具合と合わない場面が多く、迷惑をかけてしまったことだろう。

### 6.2.3 新しい技術に触れる

WebStart は web 上にある Java アプリケーションをブラウザから起動するという新しい仕組みである。アプレットとは大分違う、という印象だが、結局のところセキュリティ制限がかかるのは同じであり、大差はない。そして起動される Java アプリケーションは、

普通のクライアントマシンで動くスタンドアローン Java アプリケーションである。新しい技術を習得できる、と思っていたがどうやらさほど大した変わりはないことを知ったときは、少々落胆した。しかし、このように従来の開発とほぼ変わらずに新しい技術を導入できる、という点で WebStart はすばらしいものである。その開発のしやすさから今後 WebStart がますます普及していくだろうと感じている。

また、今回、最近方々でとりあげられている xml を通信部分に用いた。ウェブ上にも様々な xml 技術の紹介、xml の解析の howto ページが存在していたが、howto ページはまだまだひどいものであった。解析の方法がわかるのだが、一体その仕組みがどうなっているのか、などにはほぼ触れずにいる。改めて web ページを１００%信じるものではないことを痛感した。xml 解析に悩んでいるときに参考にしたのが、研究室にある xml 関連の書籍であった。やはり、本は読むべきである。きちんとした本は xml の構造からきちんと書かれている。これらの本を参考にすることで開発を進めることができた。研究会のメンバーも研究室の本を大いに活用したほうがよいと思う。冗談ではなく「ＣＬＭＳ」がはやく導入されて、より本を利用しやすい環境を作って欲しいと思う。

### 6.2.4　PM 制度

今期の開発で一番、今までと違うのは PM(プロジェクトマネージャ) の存在である。今までの研究会では、グループワークの中で自然とリーダーが生まれて、プロジェクトを取り仕切ってきたが、今回ははじめからプロジェクトの進行を管理するＰＭがいた。下で働く開発者にとっては非常にありがたいことである。何か実装で困ったことがあったら相談でき、また、毎週、ここまでやってくればよい、という指標が見えプロジェクトが進んでいくのがわかりやすかった。このようなＰＭ制度は、特に新規履修者にとってすごくありがたいことである。今後の研究会もＰＭ制度を取り入れていって欲しい。

### 6.2.5　最後に

今回のプロジェクトが無事終えられたのは、ＰＭはじめプロジェクトメンバーのおかげであり、深く感謝する。また、毎週の意見、要望を出していただいた、先生、研究会メンバー、CreW メンバーにも深く感謝します。今期、このようないいプロジェクトにめぐりあえて本当によかったです。ありがとうございました。

## 6.3 小林　敦

今期の研究会が私にとって，グループによる規模の大きなプロジェクトへの初めての参加となった。そこで感じたこと，学んだことを中心に述べていきたいと思う。

### 6.3.1 使用を前提としたアプリケーションの開発

プログラミング入門に始まった私のプログラミングの経験において，今回の OPAS プロジェクトのように，「来期確実に必要となる」といったような，具体的な使用が前提となったアプリケーションの開発は初めてのことであった。この感覚は，今まで感じたことのないものであり，いいものを作ろうというやる気とともに，逆にいいものを作らなければならないという責任も感じるという奇妙な感覚であった。

実際，次節で述べるように，私は仕様・モデルの分析にこれほど時間を掛けたことはなかったし，ここまで分析に時間を掛けることに対して，最初は疑問も感じていた。しかし，この仕様・モデルの分析に時間を掛けたことは，今考えると当然のことであり，むしろこの部分をおろそかにしていたら，大変なことになっていただろう。

利用者の存在を念頭に置くということは，なにも分析に限定されることではなく，設計においても，実装においても頭の片隅に常につきまとった。自分が利用する側だったら，という想像をしつつ作業をすることは，なかなか神経を使うものだった。しかし，こういった経験が，人を幸せにするアプリケーションを作成するために，必要となるのだろうと感じた。

### 6.3.2 仕様・モデルの分析

上で述べたように，本プロジェクトでは，仕様・モデルの分析には多くの時間を掛けた。特に，ユースケースなどの仕様の分析については 1 ヶ月以上もの時間を掛けた。利用者が最初から限定されていて，しかもメンバーの一部は，このユースケースについて経験したことがあるという状況にもかかわらず，これだけの時間が掛かるということは，本当に利用者が望むシステムを開発することは，大変難しいことなのだということを実感した。まして，まだ利用者がいない，または利用者の状況が掴めない状態では，利用者の望むシステムなど到底開発できないだろう。仕様・モデルの分析においては，こういったクライアントに対する理解を深めることの難しさを感じた。

### 6.3.3 アプリケーションの共同開発

グループでのアプリケーション開発も，私にとってはほとんど経験のなかったことといえる。前期の個人作業をしながら，グループでの開発が大変そうながらも楽しそうであったため，今期のグループの開発は期待しており，実際相応に充実感があった。また，プロ

ジェクトマネージャの存在も大きく，グループの中で開発力のない私にも，上手くタスクを振っていただけたと思う。ただ，やはり他の 2 人のメンバーに頼ってしまっている部分は多く，未熟な分，仕事量でカバーをするべきだったが，実際はそうもいかなかった。共同開発において重要なこととは，うまくタスクを分担することと，グループ内でのコンセンサスを取れることであることを強く感じた。そのためにも今後は，より可読性の高いプログラムを組むことなど，自身で常に注意を払っていけることを改めて気を付けようと考える。

### 6.3.4　初めて触れた JSP

今期は，開発の方針などとは別に，新しい技術的なことも多く学んだ。その中でも特に，実装でも触れた MVC 構造という考え方は新しく，それを実現することのできる JSP は私にとって目新しいものであった。JSP という規格自体は，一般的な Web アプリケーションを利用する際に目にしていたが，実際に取り組むのは初めてであった。この JSP については，プロジェクトマネージャや他メンバーに少し教えられただけで馴染むことができ，今後もこの知識を有効に利用していきたいと思う。

### 6.3.5　最後に

私にとって本プロジェクトは，Web アプリケーション開発において大きな一歩となった。OPAS プロジェクトは来期に運用を始めて，保守管理という新しい仕事も増える。それに携わりながら，今後も今回の経験を活かしていきたいと思う。開発だけでなく，ドキュメントの作成など様々な面でご協力くださったプロジェクトマネージャを始め，大岩先生，大岩研究会の方々に，ここで感謝の意を表したい。本当にありがとうございました。