プログラミング教育における課題提出システム “OPAS(Open Programming Assignment System)”の構築

杉浦　学[†]　　大岩　元[‡]

概 要

本稿では，慶應義塾大学 環境情報学部 大岩元研究室によって本年度から開始された「プログラミング教育を対象とした支援環境の構築に関する研究」の一環である，プログラミング教育における課題提出システム“OPAS(Open Programming Assignment System)”の構築に関する報告を行う．“OPAS”は，我々が開発するプログラミング教育カリキュラムの特徴である，1). プログラムの目的が明確で，可読性が高いソースコードを記述することを重視する 2). 他の受講者と議論やレビューを繰り返しながら，プログラムの設計やソースコードの可読性について学習する 3). 問題の分析から，設計・実装・評価というプログラミングのプロセス全体を行う能力を育てることを目標とする　をより効果的に実践するためのシステムである．受講者が“OPAS”を利用することで,（ソースコードや設計文書といった）演習課題の成果物を簡単に提出することができ，それらを受講者全員でお互いに閲覧することができる．これにより，講座中に他人のソースコードや設計文書を読む機会を多く提供することができ，これらの成果物に対して受講者同士でレビューを行うといった形態のカリキュラムが容易に実践可能となる．また，“OPAS”は講座の運営側にとっても課題の提出管理・採点環境として活用できるシステムである．

1　はじめに

本稿では,「プログラミング教育を対象とした支援環境の構築に関する研究」の一環である，プログラミング教育課題提出システム “OPAS(Open Programming Assignment System)” の構築に関する報告を行う．

「プログラミング教育を対象とした支援環境の構築に関する研究」は，慶應義塾大学 環境情報学部 大岩元研究室（以下，大岩研究室と略す）によって本年度から開始された．“OPAS” はこの「プログラミング教育を対象とした支援環境」の一部を構成するシステムである．本稿では「プログラミング教育を対象とした支援環境の構築に関する研究」の全体像を明らかにした上で，本年度の研究成果と位置付けられる “OPAS” の構築について報告する．

“OPAS” は，大岩研究室で開発されているプログラミング教育カリキュラムの特色を踏まえた教育活動をより効率的に実践するために構築された．大岩研究室で開発されたプログラミング教育に関する論文のリストを 6 章に，付録に論文の本体を掲載した．

[†]慶應義塾大学 政策・メディア研究科
[‡]慶應義塾大学 環境情報学部

2 背景

2.1 プログラミング教育カリキュラム開発と講座運営

大岩研究室では，学生や社会人を受講対象としたプログラミング教育のカリキュラム開発と，そのカリキュラムを使用した授業や講座の運営[1]を行ってきた．

これらのカリキュラムは，対象者をプログラミングの初学者に設定し，構造化プログラミングからオブジェクト指向プログラミングの初歩 [1] をその学習内容としている．

2.2 カリキュラムの特徴と課題

カリキュラムの特徴としては，以下のような事柄が挙げられる．

- プログラムの目的が明確で，可読性が高いソースコードを作成することを重視する
- 受講者間で議論とレビューを繰り返しながら，プログラムの設計やソースコードの可読性について学習する
- 問題の分析から，設計・実装・評価というプログラミングのプロセス全体を行う能力を育てることを目標とする

こうした特徴を有するカリキュラムを実施する際に，以下のような問題が発生することが明らかになってきた．

- 教育効果が指導者の能力に左右される度合いが大きい
- 講座の運営に必要なコストが大きい
- 受講者の使用に適した教育用のツールがない

次からこれらの問題の詳細について述べる．

2.2.1 指導者の能力と教育効果の関係

プログラム設計やソースコードの記述には，良い設計や可読性の高いソースコードを作成するためのガイドラインは存在するものの，唯一の正解は存在しないという事柄を受講者が認識することが重要になる．正解がないという事実を受講者に認識してもらうためには，指導者が一方的に知識を伝達するような講義形式の教育だけでなく，受講者が自らの力で試行錯誤して演習を行い，その成果物を利用して議論を行うことが有効である．

受講者が演習や議論を行う場面で指導者が果たすべき役割は，受講者の演習に対する取り組みや議論が効果的になるような指導を個別に行い，受講者の成果物に対して教育的な視点からレビューを行うことである．効果的な議論のドライブやレビューを行うためには，表面的な知識だけではない，指導者のプログラミングに関する深い理解が要求される．よって，理解と能力が不足した指導者にもとでカリキュラムを実施しても，議論や演習に対するレビューの質の問題から，意図した学習効果が得られないことが多い．

[1] オブジェクトプログラミング（2001-2003 慶應義塾大学），株式会社 Exa 新入社員研修（2001-2003），ITSS 試験講座（2003）

また，指導者は学習目標や問題の出題理由といった，カリキュラムを開発した者の意図を十分に理解する必要がある．こうした理解がなければ，どのように受講者に働きかければ良いかという指針を得ることができない．こうした問題は指導者に限ったものではなく，演習時間に受講者のサポートを行うアシスタントにも共通するものである．

2.2.2 講座運営コストの増大

演習と議論を中心にしたカリキュラムのプログラミング教育では，受講者が作成した設計文書やソースコードといった大量の成果物が蓄積する．それらを効率的に管理し，指導者が的確なレビューを行う必要があるため，講座の運営にかかるコストは講義形式のみの講座に比較して大きくなる．

成果物の管理作業や受講者の質問に回答する作業は，一人の講師だけで行うのは不可能であるため，講座運営に必要な作業を行うアシスタントが講座運営に必要な作業を行うことが期待される．しかし実際は，大学で行われる授業では特に顕著だが，十分なアシスタントの人数を確保できないことが多いため，講座の運営に携わる個人の負担が増大してしまう傾向が強い．

2.2.3 教育用に開発されたツールの必要性

現在受講者は，プログラミングのプロフェッショナルが使うことを前提にしたツールを使用して，プログラムの設計（HCP チャート [2][3] や UML の記述）を行い，ソースコードを作成している．こうしたツールは教育目的で使用することを想定していないため，熟練した開発者にとって便利な機能が搭載されている．とくに初学者にとって，ツールに搭載された便利な機能は，プログラミングを学習する上で必要な基本的な事柄を隠蔽してしまうので，プログラミングに対する理解を阻害する原因になる．同時に，多くの機能を持ったツールは操作が複雑で，その使用方法に気を取られてしまい，受講者が考えることに集中できないという問題も生じる．

また，我々が独自に開発した教育目的の図法[2]に基づいた図面を作成するエディタは存在しないので，受講者は手書きで大量の図を作成する必要があり，効率が悪い．また運営側にとっても，紙媒体の図面を大量に管理する必要が生じるため，講座の運営に必要な手間が増えるという事柄にもつながる．

[2]付録の論文 “プログラミング入門教育におけるモデルによるプログラムの状態表現” を参照

3　研究目的

3.1　支援環境の構築

2.2節で述べた問題を解決し，プログラミング教育の実施を支援する環境を構築することが，1 章で述べた「プログラミング教育を対象とした支援環境の構築に関する研究」の目的である．

ここでいう支援環境とは，講座運営を支援する Web アプリケーションや各種エディタ等のツールの総称である．この環境が行う支援内容と対象者を図 1に示す．

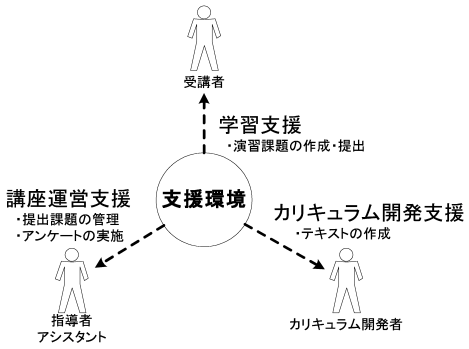


図 1: 支援内容と対象者

次から，現段階で具体的な支援環境として開発を予定（完了）しているツールの概要を述べる．

3.1.1　各種エディタ

(1) 参照モデルエディタ（開発予定）

参照モデル[3]を利用した図面を作成するためのエディタ．

プログラムの状態により，モデルが変化する様子を連続的に表現できる機能が必要．他の開発済みエディタと同様に，テキスト作成支援ツールから利用できるファイル形式での保存をサポートする．

(2) 教育用プログラムエディタ（開発予定）

教育用のプログラム（ソース）エディタ．

コンパイルログの保存機能や，課題提出システム（“OPAS”）と連携して課題提出をサポートする機能が必要．

(3) HCP チャートエディタ（開発済）

HCP チャートの作成をテキストベースで行うことができるエディタ[4]．

我々の開発したチャートに対するレビュー手法[5]に基づいた評価に対応しており，チャートに対する評価を容易に数値化することができる．テキスト作成支援ツールから利用できるファイル形式の保存をサポートしている．

[3]付録の論文 “プログラミング入門教育におけるモデルによるプログラムの状態表現” を参照
[4]http://www.crew.sfc.keio.ac.jp/projects/2004hcpviewer
[5]付録の論文 “プログラム設計教育における HCP チャートのレビュー手法” を参照

(4) **教育用 UML エディタ（開発予定）**

他の大岩研究室のプロジェクトで開発された UML エディタをベースに，受講者のレベルに適合した柔軟なカスタマイズが可能な UML エディタ.

3.1.2 講座運営サポートシステム

(1) **課題提出システム“ OPAS ”（開発済）**

本稿で述べる課題提出システム

(2) **講座評価用アンケートシステム（開発予定）**

受講者による，講座に対する評価を容易にフィードバックするための Web アプリケーション.
評価データを効率的に分析できる形式で保存できる機能が必要.

(3) **テキスト作成支援ツール（開発済）**

SmartDoc[4] と XSLT[5] を利用した XML による Latex/HTML ドキュメント作成をサポートするツール.

多人数によるドキュメント作成作業を容易に行うことができ，作成したドキュメントの再利用性を高く保つことができる.

3.2 “OPAS” の位置付け

3.1節で述べた「プログラミング教育を対象とした支援環境」の一部として，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で出題される課題を快適に提出し，それらの解答を受講者が互いに閲覧できるシステムである.

つまり，“OPAS” によってサポートする内容は図 1のうち,「スタッフによる提出された課題の管理作業」と,「受講者による課題の解答の提出」である.

3.3 “OPAS” の機能とカリキュラムの関連

課題提出システム “OPAS” が備える機能と，大岩研究室で開発したプログラミング教育カリキュラムの特色との関連を以下に述べる.

- 複数の解答ファイルを含むディレクトリを，階層構造を保持したまま提出可能である
  - 受講者は自分の作成したプログラムの環境をディレクトリごと提出することが可能（複数ファイルの提出が簡単）
  - 指導者は受講者の実行環境を再現することが容易で，問題を抱えている学生を的確にレビューすることが可能
- 閲覧時には解答の自動フォーマットを行う

  - 受講者の答案を所定の形式で自動的に文書化することで，毎回の課題の閲覧・採点作業の効率化に寄与する

- 受講者の解答を公開することができる
  - 受講者が他人のプログラムを読む機会を提供し，自分のソースコードの可読性を高めるきっかけを作る
  - 受講者がお互いの成果物をレビューしながら学習するというアクティビティーも行いやすくなる

このように，“OPAS” は課題の管理や採点に関する運営側の労力を軽減するだけでなく，2.2節で述べた，カリキュラムの特色を効果的に実践できるような機能を備えている．

4　研究内容

4.1　開発の概要

4.1.1　“OPAS” プロジェクト

“OPAS” は，2003 年度秋学期の大岩研究会履修者を開発メンバーとした “OPAS” プロジェクトによって開発された.

以下に “OPAS” プロジェクトに参加したメンバーを示す.

- プロジェクトマネージャー
  - 青山　希　政策・メディア研究科
- 開発メンバー
  - 岸　健司　環境情報学部
  - 明石　敬　環境情報学部
  - 小林　敦　環境情報学部

4.1.2　開発スケジュール

開発期間は，2003 年度秋学期（2003 年 10 月-2004 年 2 月）を利用して行われ，完成したシステムは 2004 年度春学期に開講された「オブジェクトプログラミング（担当 : 松澤　芳明)」で利用された.

以下に実践された開発スケジュールの概要を示す.

2003 年 10 月 …… 企画・分析
2003 年 11 月 …… 設計
2003 年 12 月-2004 年 1 月 …… 実装
2004 年 2 月 …… テスト
2004 年 4 月- …… 導入

4.2　仕様

4.2節では “OPAS” がどのような機能を備えたシステムかを解説し，その利用方法について述べる．

4.2.1　ユースケース

“OPAS” を利用するのは「受講者」と「(講師やアシスタント等の）スタッフ」である．
それぞれが “OPAS” で実現されるユースケースを以下に述べ，ユースケース図を図 2に示す．

- スタッフのユースケース
  - 「課題を管理する」
    課題の出題と各種の設定（締め切り・解答の閲覧可否の設定）を行うことができる．
  - 「解答を採点し，感想に対するフィードバックを付与する」
    採点自体はシステムでサポートしていないが，統一された形式で課題を閲覧し，簡単な操作で受講者が記述した感想に対してフィードバッグを記述することができる．
  - 「受講者の解答を閲覧する」
    受講者が提出した解答を閲覧することができる．
  - 「授業を管理する」
    課題の提出を行うための授業を作成・削除することができる．一つの授業につき一つの URL が付与されるので，この URL を受講者に伝えてシステムの利用を開始してもらう．
  - 「ユーザを管理する」
    ユーザの削除やパスワードの変更といった運用上必要な機能を提供する．
- 受講者のユースケース
  - 「課題の解答を提出する」
    ディレクトリごと課題の解答ファイルを提出することができる．
  - 「課題の提出を確認する」
    課題が提出済みかどうかを閲覧することができる．
  - 「感想に対するコメント（フィードバック）を閲覧する」
    感想に対するスタッフによるフィードバックを閲覧することができる．
  - 「他の受講者の解答を閲覧する」
    （解答閲覧可の設定がスタッフによって行われていた場合）他の受講者が提出した解答を閲覧することができる．

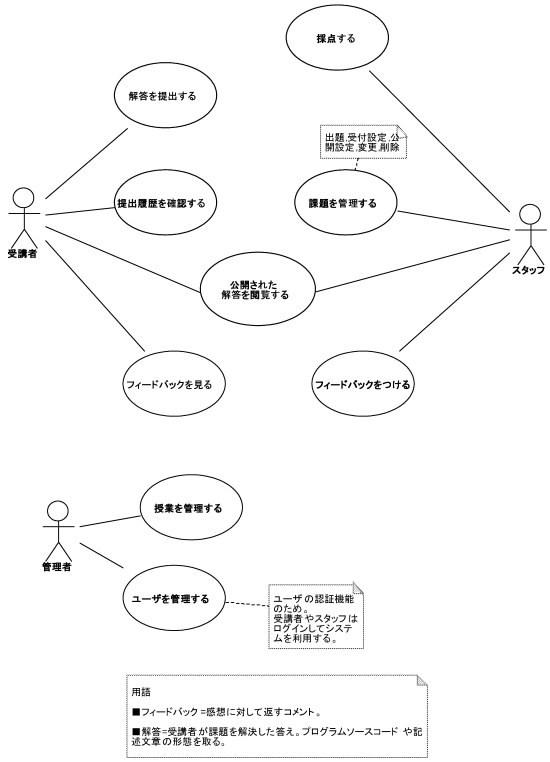


図 2: “OPAS” のユースケース図

4.2.2 ワークフローと画面の解説

スタッフと受講者がどのようにシステムを利用して課題の提出を進めるかを図 3に示す.

スタッフは課題を告知して解答の受付を開始する．受講者は告知された課題に対する解答ファイルを提出し，それに対してスタッフが採点を行うという流れが，課題提出の基本的なワークフローになる.

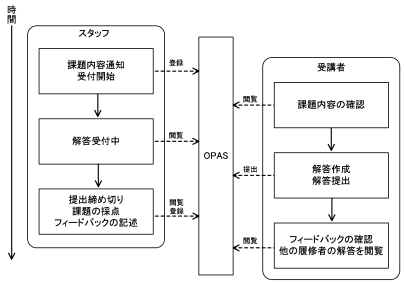


図 3: “OPAS” を利用した課題提出の流れ

図 4: “OPAS” トップ画面

図 4が “OPAS” に Web ブラウザでアクセスした時のトップ画面である．受講者とスタッフはこの画面からシステムにログインし，様々な操作を行うことができる．新規のユーザ登録もこの画面から行う．

(1) 受講者に関係する画面

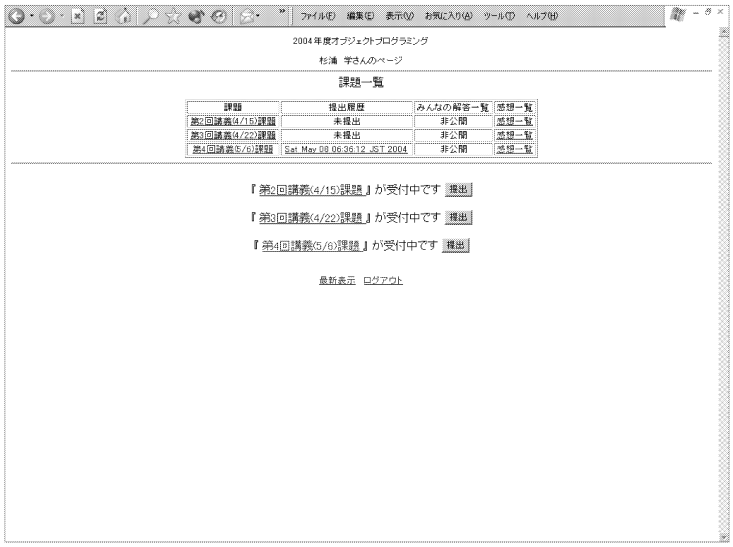


図 5: 受講者トップ画面

図 5が受講者が “OPAS” にログインした時に表示される画面である．ここから課題内容の閲覧と提出，提出状況の確認を行うことができる．

図 6: 課題内容確認画面

図 6が出題されている課題の内容を確認するための画面である．受講者はこの画面を見て，出題されている課題の情報を確認することができる．

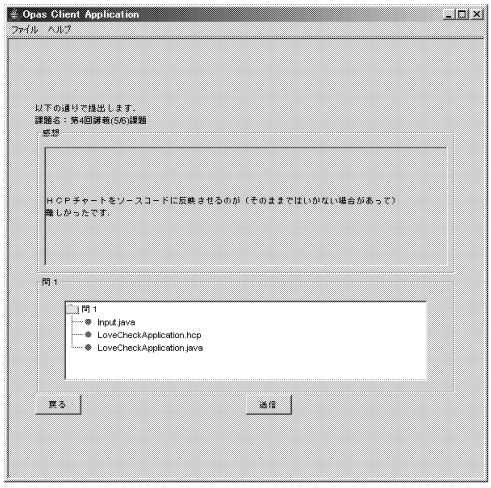


図 7: 提出用アプリケーション

図 7が図 5や図 6の提出ボタンをクリックすると起動する提出用アプリケーションである．受講者は提出したい解答ファイルのあるディレクトリを選択し，感想を記述することができる．提出されるディレクトリの階層構造が解答ファイル群とともにアプリケーションに表示されるような工夫をすることで，提出間違いを防止する．解答の提出が完了すると，図 5に示した受講者のトップ画面から提出内容を確認することができる．

(2) スタッフに関係する画面

図 8: 課題編集画面

図 8が課題に関する各種設定を行う画面である．出題形式（記述問題・プログラムの作成問題）などが選択でき，受講者が他の受講者の解答を閲覧できるかどうか等の各種設定を行うことができる．

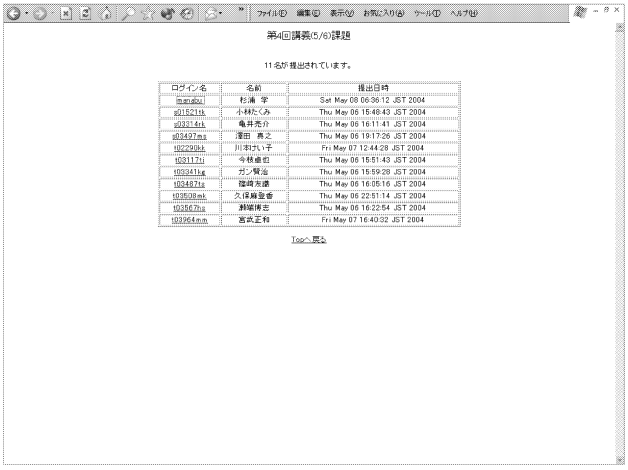


図 9: 課題一覧画面

図 9は提出された課題の一覧を表示する画面である．スタッフは常にこの画面を使って最新の課題提出状況を確認することができる．解答閲覧可の設定がスタッフによって行われている時には，受講者が他の受講者の解答を閲覧する際にも利用される．

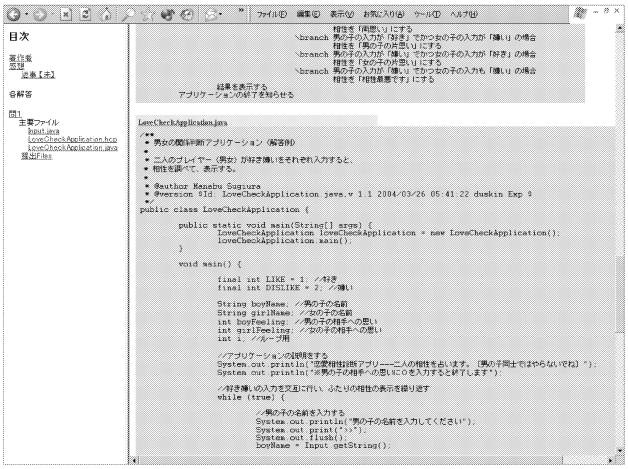


図 10: 課題閲覧画面

図 10は課題の詳細を表示する画面である．スタッフはこの画面を使って課題を採点する．プログラムは決められたフォーマットにしたがって表示され，画面左のナビゲーションリンクを使って，効率的に課題を閲覧できる工夫がされている．

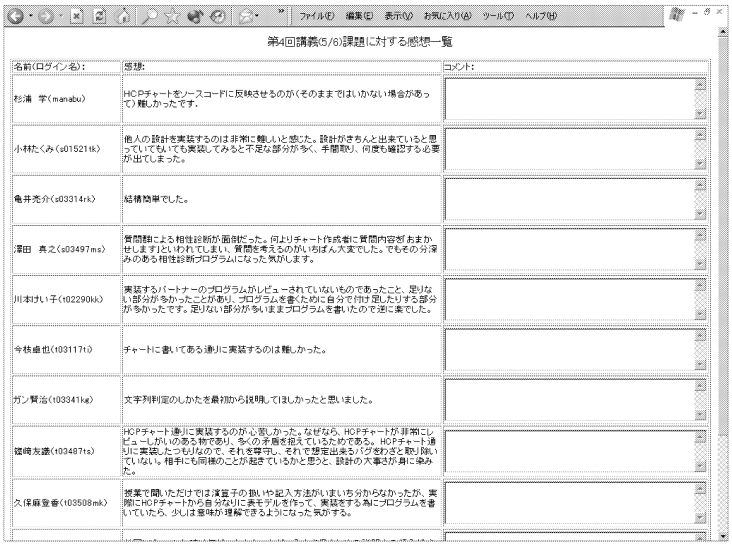


図 11: フィードバック作成画面

図 9は受講者の感想に対して，スタッフがフィードバックを記述する画面である．感想のみを一覧表示することができるので，効率良くフィードバックを記述することができる．感想は講座を進める上で重要な情報源となるので，感想を一覧表示する画面も用意されている．

4.3 システム設計と使用アーキテクチャ

4.3.1 システムの構成

“OPAS” は課題を提出するためのアプリケーションと，課題受付や閲覧を行うためのサーバサイドで稼動するプログラムで構成される．

サーバサイドで稼動するプログラムは Java Servlet を使用して実装した．課題の提出以外の機能は Web ブラウザから利用することができる．

提出アプリケーションは受講生の使用するコンピュータのファイル（解答ファイル）にアクセスすることが簡単であるため，Java のスタンドアロンアプリケーションで実装した．提出アプリケーションは再配布の手間がかからないように，Java Web Start を利用して実行するように実装してある．

システム構成の概要を図 12に示す.

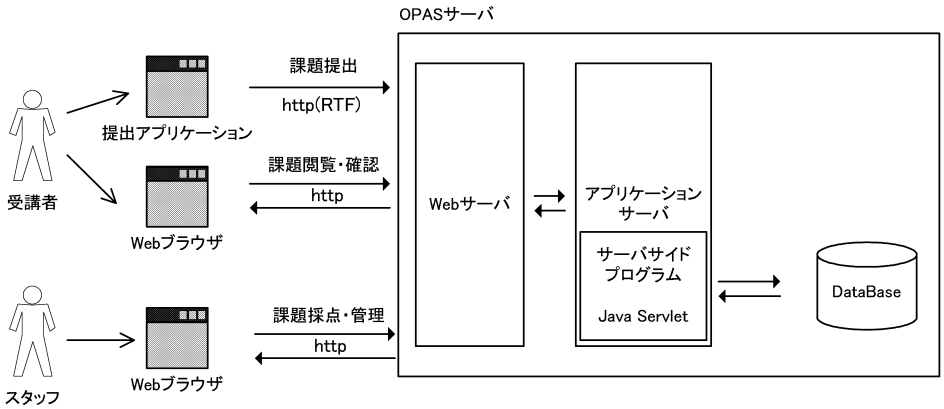


図 12: システムの構成

表 1: “OPAS” の構成要素

OPAS サーバプログラム	OPAS の Web アプリケーション部を担う Java Servlet で実装
OPAS 提出アプリケーション	受講者が提出物を提出するためのアプリケーション WebStart でブラウザから起動する
OPAS 提出アプリケーションサーブレット	OPAS サーバプログラムの一部 提出アプリに必要な情報を送信し，提出物を受信する

4.3.2 設計と実装

システムで扱うモデル部分は UML を用いたオブジェクト指向設計を行って実装した．モデル部分の設計のクラス図を図 13に示す.

Servlet は，Struts[6] に似た独自の簡単なフレームワークを作成して実装を行った.

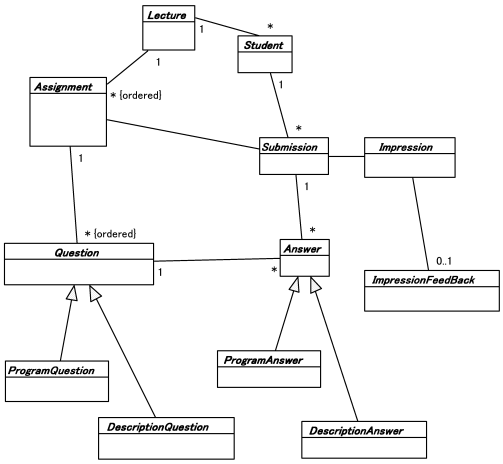


図 13: モデルの設計（クラス図）

提出された課題の情報などのデータはデータベースに保存する必要があるが，オブジェクトをデータベースのリレーションにマッピングする際に Torque[7] を利用した．

4.4 導入例と評価方針

4.4.1 授業での利用

“OPAS” は 2004 年度春学期から開講されている「オブジェクトプログラミング（担当：松澤芳明)」で利用されている．この授業では，“OPAS” を利用している登録者数は 66 名である．

4.4.2 評価

“OPAS” に関する評価は，学期末に「オブジェクトプログラミング」の履修者に対するアンケートを行い，意見を収集する予定である．また，講座運営コストの軽減効果に関する評価も学期末に行う予定である．

これまでの利用で明らかになった問題点は，5.1節にて述べる．

5　課題と今後の展望

5.1　問題点と解決策

“OPAS” を実際の授業で使用した結果明らかになった問題点と解決案を以下に述べる．

- 「提出アプリケーションが正常に稼動しないことがあった」

  Java Web Start を正常に起動できない場合が多く，現在は Java Applet 版の提出アプリケーションを作成している．

- 「提出前日などで，サーバに負荷がかかった場合の動作が不安定になることがあった」

  利用者数が多い大規模な授業での使用に対応するために，データベースに関連する設計・実装の見直しを検討している．

- 「提出アプリケーションや閲覧画面等のユーザインターフェースが原因となって，ユーザが直感的に操作できないことがあった」

  ユーザインターフェースの設計に費やす時間が多く取れなかったため，ユーザの意見を踏まえた改善を行う予定である．

5.2　機能拡張と支援環境との連帯

現在の利用者に “OPAS” の評価を行ってもらい，その結果を踏まえた改善を行うことが優先される．今後の展開として，3.1節で述べた他の支援ツールとの連帯を視野にいれた拡張を行うことを予定している．具体的には，提出されたソースの実行機能や開発を予定しているプログラムエディタと連帯した提出機能などである．

“OPAS” がより安定して稼動し，より便利な機能を搭載していくことで，将来的には大岩研究室が実施するプログラミング教育だけにとどまらず，大学をはじめとした教育機関等で幅広く利用されることも目標にしている．これは，“OPAS” というシステムを通じて，大岩研究室が開発したプログラミング教育のカリキュラムがより多くの人達に受け入れられるきっかけになると考えているからである．

6　発表論文リスト

本稿で述べた “OPAS” に関連し，大岩研究室で行われているプログラミング教育に関する論文のうち，2003 年 4 月 1 日から 2004 年 3 月 31 日までに発表されたものを以下に示す.

“「目的の表現」に注目したオブジェクト指向プログラミング教育とその評価”
松澤芳昭, 青山希, 杉浦学, 川村昌弘，大岩元
情報処理学会研究会報告 (CE-72-15), pp.77-84, 2003.

“プログラミング入門教育におけるモデルによるプログラムの状態表現”
青山希, 松澤芳昭, 杉浦学, 川村昌弘，大岩元
情報処理学会研究会報告 (CE-72-15), pp.109-114, 2003.

“プログラム設計教育における HCP チャートのレビュー手法”
松澤芳昭, 杉浦学, 大岩元
情報処理学会第 66 回全国大会（1C-2）, pp.4-343 - 4-344, 2004

参考文献

[1] “オブジェクト指向技術者養成のためのカリキュラム”
松澤芳昭, 中鉢欣秀, 岡田健, 大岩元 情報処理学会研究会報告 (CE-64-1), pp.1-8, 2001.

[2] “階層化プログラム設計図法 - HCP チャート -”
長野宏宣, 浅見秀雄, 忠海均 企画センター, 1992.

[3] “プログラム設計図法”
花田收悦 企画センター, 1983.

[4] “SmartDoc Version 1.0”
http://www.asahi-net.or.jp/˜dp8t-asm/java/tools/SmartDoc/index_ja.html

[5] “XSL Transformations (XSLT) Version 1.0”
http://www.w3.org/TR/xslt

[6] “The Apache Struts Web Application Framework”
http://jakarta.apache.org/struts

[7] “Torque”
http://db.apache.org/torque

付録 “プログラミング教育に関する投稿論文”

「目的の表現」に注目した オブジェクト指向プログラミング教育とその評価

松澤芳昭[†] 青山希[†] 杉浦学[†]
川村昌弘[†] 大岩元[††]

本稿では，オブジェクト指向プログラミング教育の一手法として,「目的の表現」に注目したアプローチを紹介する．我々の提案する教育手法では，構造化プログラミングの段階でプログラムの目的に注目し，それらをどのように捉え，表現するかを教育する．オブジェクト指向プログラミングの教育では，目的記述中の動詞と名詞を手がかりに，クラスやインターフェイスを設計するように教育する．カリキュラムと評価方法を考案し，若手技術者に対して実験した．結果，提案手法は受講者がオブジェクト指向を受け入れやすく，導入する意義も伝わりやすいという成果を得た．

An Instruction Method for Object Oriented Programming focused on "Expressing Purpose Structure" and Its Evaluation

Yoshiaki Matsuzawa,[†] Nozomu Aoyama,[†] Manabu Sugiura,[†]
Masahiro Kawamura[†] and Hajime Ohiwa[††]

This paper proposes "Expressing Purpose Structure" approach as an instruction method for object-oriented programming. Our method focuses on clarifing and describing the purpose structure of the program which is constructed from sub-purposes at the early stage of programming instruction. Class structure and its interfaces are easily developed by examining objects and verbs in the developed purpose structure. The curricula based on this method have been tried to young engineers of Japanese IT industry and appreciated by them. This method is easy for learners to understand the merit of object-oriented programming.

1. はじめに

近年，サーバサイド Java の普及などを背景に，オブジェクト指向技術を使いこなしてソフトウエアを設計，開発できる技術者のニーズが高まっている．今回我々が試行したオブジェクト指向プログラミングの実験講座においても，半月あまりで定員の 2 倍以上の応募があった．

最近では，オブジェクト指向の概念の知識や，フレームワークの利用法などを解説する書籍が多く出回っている．それらの書籍を読んで独学で学習することにより，オブジェクト指向のメリットを享受して，便利なフレームワークを利用したプログラムならば書けるようになる．しかし，知識だけでオブジェクト指向を利用できるのはそこまでである．オブジェクト指向プログラミング能力はガニェのいう言語情報ではなく，知的技能，認知的方略に分類される[5)2)3)]．そうした領域の能力を養わなければ，オブジェクト指向のメリットを活かしたプログラムを自ら設計するような，オブジェクト指向の運用能力を身に着けることはできない．

今回行った実験講座の受講者募集に当たり,「構造化プログラミングなどの経験は有するが，オブジェクト指向のメリットを活かしたプログラムを書くことができない人」という対象を設定した．集まった受講者は，実際に業務をしている若手技術者である．オブジェクト指向の知識が皆無の者は少数で，サーバサイド Java に関わる業務をしている者も少なからずいた．しかし，その大半は前述のように業務で

† 慶應義塾大学 政策・メディア研究科
Graduate School of Media and Governance, Keio University
†† 慶應義塾大学 環境情報学部
Faculty of Environmental Information, Keio University

必要な要素知識を覚えて使用しているといった状況であった．彼らの多くは，関連書籍などを利用して独学で学習しているが，なかなかうまくオブジェクト指向を利用できないという問題意識を持っていた．

オブジェクト指向設計では，システムを理解しやすいように，プログラムをクラスという単位に分解する能力が求められる．しかし，プログラムを分割して構造を整理することは，オブジェクト指向でなくても容易な作業ではない．実務者にはプログラムの理解のしやすさよりも，短期間で機能を実現することが要求されるので，たとえ小さなプログラムであってもプログラムの分割作業についてじっくりと考える時間がない．そのような実務者は，クラス設計の前提技能としてのプログラムの分割能力が低いと考えられるので，独学でクラス設計を学ぶのは難しい．

技術者が，プログラムがどのような処理を行うか，といったソフトウエア実現の「手段」に偏って教育されるのも問題である．オブジェクト指向如何に問わず，設計作業では事象を系統的に捉え，プログラムが果たす「目的」をトップダウンに詳細化していく能力が求められる．手段に偏った教育ではこの視点を獲得することができない．

このことは，試験講座の事前試験として，HCP チャート[7)4)] を用いてプログラムの目的とその構造を表現するという課題 (4.1 節を参照) を行ってもらったところ，ほとんどの受講者がこれを達成できなかったことでも裏づけされる．

それゆえ，我々は，構造化プログラミングの段階から，他人に理解しやすいように，目的を明確に表現することを意識して，可読性が高く，構造も整理されたプログラムを書くことができるようにする教育をすることが重要だと考える．そして，その延長線上にある，さらに強力に目的を表現できる手法としてオブジェクト指向を導入する．

このアプローチで教育するためのカリキュラムを作成し，実験したところ，プログラムの構造を整理して目的を明確に表現するとプログラムの可読性が向上する，という視点を獲得してからオブジェクト指向へ進むので，受講者がオブジェクト指向を受け入れやすく，導入する意義も伝わりやすいという成果を得た．

本稿では，オブジェクト指向教育の一手法として，「目的の表現」に注目したアプローチを紹介する．まず，2 節で，プログラムにおいて目的をどのように表現するか，その方法と，オブジェクトのプログラムへどのように発展するかを具体例を用いて述べる．3 節では，本稿で提案するアプローチの教育方法についてを述べ，4 節では実験授業のカリキュラムと実施体制を示す．そして 5 節で，受講者の成果物とアンケートを用いて考察を行う．

2. プログラム上で目的を表現する方法

本節では，プログラムにおいて目的をどのように明確に表現するかということ，そしてそれをどのようにオブジェクト指向のプログラムに発展させるかを述べる．

2.1 ブロックによる表現

我々は，プログラミング教育の初歩の段階で目的を表現する方法として，プログラムを「ブロック」というまとまりに分解することと，分解されたブロックに適切なコメントを記述するという方法を用いる．

図 1 に示されたプログラムを使って具体的に説明する．このプログラムは，タートルグラフィックスを利用して，亀に家を書かせるプログラムの Java 版である．実行すると，屋根と本体からなる家が描画される．このプログラムの目的は，家を書くことである．

家は，屋根と本体から構成される．これを表現するためにブロックを作成する．そのために，プログラム中の「屋根を書く」という部分と,「本体を書く」という部分の間に空行を 1 つ挿入する．こうして作られたブロックは，いわゆる一般的なプログラミング言語のブロックとは異なり，処理の構造ではなく，論理的な構造を表現する．

すべてのブロックには，そのブロックがどのような目的を持つのかを記述した,「ブロックコメント」を自然言語で記述する．

ここで，適切なブロックコメントをつけることは，実はプログラム初学者のみならず，プログラム上級者でも難しい作業である．一般的なガイドラインとして，コメントには目的を書くべきであるというものがある[1)]．しかし，ここでブロック中の処理を抽象化する視点で目的を考えた場合，図 2 のように，三角形を書く，四角形を書くという目的を記述することもできる．その場合，ブロックの構造は同様になるが，プログラムの様子はまったく異なってみえる．

「屋根を書く」,「三角形を書く」というコメント

```
/**
 * 家を書くプログラム
 */
public class House extends Turtle{

  void start(){
    //屋根を書く
    rt(30);
    fd(50);
    rt(120);
    fd(50);
    rt(120);
    fd(50);

    //本体を書く
    lt(90);
    fd(50);
    lt(90);
    fd(50);
    lt(90);
    fd(50);
    lt(90);
    fd(50);
  }

}
```

図 1　ブロックによる表現

```
/**
 * 家を書くプログラム
 */
public class House extends Turtle{

  void start(){
    //三角形を書く
    rt(30);
    fd(50);
    rt(120);
    fd(50);
    rt(120);
    fd(50);

    //四角形を書く
    lt(90);
    fd(50);
    lt(90);
    fd(50);
    lt(90);
    fd(50);
    lt(90);
    fd(50);
  }

}
```

図 2　ブロックコメントを処理から考えた場合

を記述することは，どちらもプログラムの可読性の寄与のために必要な情報である．目的の粒度が調整されて，これらがうまく表現される必要がある．目的の粒度を整理して，階層構造としてまとめると図 3 のようになる．空行によるブロックの表現では，入れ子になったブロックを表現するのが難しいので，これをうまく表現するために，手続きを利用する．

2.2　手続きによる表現

目的を明確に表現するために，構造化プログラミング言語では手続きを利用できる．

家を書くプログラムの例では，三角形や，四角形を書くプログラムを抽象化して，多角形を書くといった手続きを作ることができる (図 4)．これにより，前節でのブロックの入れ子の問題が解決でき，可読性の向上のために必要な情報を失わないまま，目的がより簡潔に表現される．

さらに，図 5 に示すように，屋根を書く，本体を書くといった粒度の目的に関して手続きを作ることも大切である．家を書くという視点から考えた場合，屋根の形を変更したり，本体の形を変更したりすることも考えられるからである．

```
- 家を書く
    |
    | - 屋根を書く
    |     |
    |     | - 三角形を書く
    |
    | - 本体を書く
          |
          | - 四角形を書く
```

図 3　家を書くプログラムの構造

しかし，この規模のプログラムで，屋根を書く，本体を書くという手続きを作るのは，いささか冗長である．これを実際に手続きにするかどうかは，プログラムの大きさによって決める必要がある．手続きにしない場合も，これをブロックとしてプログラムを分離されているので，十分目的は明確に表現されている．そして，このことは，ブロックとメソッドはプログラムを分解する単位として同義になることを示している．異なる点は，プログラミング言語の機能を利用する（つまり手続きを作る）場合，ブロックコメントがそのまま手続きの名前になるので，

```
/**
 * 家を書くプログラム
 */
public class House extends Turtle{

  void start(){
    //屋根を書く
    drawPolygon(3);

    //本体を書く
    drawPolygon(4);
  }

  void drawPolygon(int){
    …
  }

}
```

図 4　手続きによる処理の抽象化

コメントが不要になることである．

2.3　オブジェクト指向への発展

オブジェクト指向プログラミング言語では，オブジェクトという単位に分解することで，例えば図 6 に示すように表現することができる．

ここで，分解される単位や名前はこれまでの表現方法の場合と同様に考えることができる．ここで抽出された屋根 (Roof) や本体 (Body)，多角形 (Polygon) といったクラスや，書く (draw) といったインターフェイスは，初歩の段階で行ったブロックとコメントによる方法においても表現されている．

つまり，その構造を初歩の段階でうまく捉え，表現されていれば，それらの記述に含まれる動詞と名詞に注目することで，比較的容易にオブジェクト指向のプログラムに発展するのである．

最初の段階でのブロックコメントが，オブジェクト指向でのクラス設計に関連しているので，目的を明確に表現するブロックコメントを定めておくことが非常に重要である．特に，屋根を書く，三角形を書くという 2 つの視点，つまりブロック中の処理と，プログラムの大きな目的からの視点で目的を捉え，表現されていることが重要である．しかし，プログラムの動作に集中してしまいがちな初学者や，今回対象とするような，実務を何年か経験している実務者は，処理から考えた目的，つまり手段に近い目的になってしまいがちである．

ここにオブジェクト指向へ発展するかどうかの大

```
/**
 * 家を書くプログラム
 */
public class House extends Turtle{

  void start(){
    //屋根を書く
    drawRoof();

    //本体を書く
    drawBody();
  }

  void drawRoof(){
    drawPolygon(3);
  }

  void drawBody(){
    drawPolygon(4);
  }

  void drawPolygon(int){
    …
  }

}
```

図 5　手続きによる目的の抽象化

```
/**
 * 家を書くプログラム
 */
public class House extends Turtle{

  Roof roof = new Roof(new Polygon(3));
  Body body = new Body(new Polygon(4));

  void start(){
    roof.draw();
    body.draw();
  }

}
```

図 6　オブジェクト指向による表現

きな壁があると思われる．オブジェクト指向では，大きな目的を捉え，表現する能力がより重要になるからである．例えば，オブジェクト指向の基本的概念であるクラスのカプセル化では，内部の手段を隠蔽し，より大きな目的に近い公開インターフェイスを作成することで，プログラムの保守性を高めることができる．従って，大きな目的を捉えることがで

きなければ，一つのクラスの公開インターフェイスでさえも，うまく設計することは難しい．

オブジェクト指向システムの分析や設計は，これを大きな粒度で捉えたものと考えることができる．オブジェクト指向では，大きな目的からの視点で，概念モデルを構築することが良いクラス設計をする上で重要とされている．オブジェクト指向によるフレームワークの設計では，それらの大きな目的ををさらに抽象化していく能力が必要である．

3. 目的の表現アプローチの教育法

本節では，前節までの議論を踏まえて，どのように目的の表現能力を教育するかを述べる．

3.1 HCP チャートの利用

目的の表現アプローチの初めの段階では，プログラミング言語などの手段にとらわれないように，自然言語を用いて表現を教育するべきである．

我々はこの段階での表現方法として，HCP チャート[7)4)] を用いることを提案する．HCP チャートは目的を階層構造として表現するのにすぐれた図法である．そのため，上位の目的からトップダウンに要素を分解し，下位の目的へと分解する段階的詳細化を行いやすい．さらに，階層構造を表現したまま，繰り返しや分岐などの制御構造も表現できる．

HCP チャートでは，データ構造も表現できる．しかし，この段階ではデータは無視して処理のみに注目させたほうが良い．プログラムにおける目的の記述は処理に対して行われるため，処理のみに注目したほうが，より目的の表現に焦点を絞ることができるためである．

3.2 相互レビューによる教育

本稿が提案するアプローチでは，表現に主眼を置いているため，学習者が表現した成果物が，伝える相手にどれだけ伝わったかを評価し，学習者にフィードバックする必要がある．

この目的の達成のために，我々は学習者同士で相互レビューを行わせる教育法を提案する．他人の表現に対するレビューを行うことは，学習者に読み手としての評価能力を養うことになり，自己の表現にフィードバックされる．また，実際に表現が不良であると相手に伝わらない経験をすることは，表現能力を養う学習の動機付けにもなる．

この方法の問題点は，人によって表現の受け取り方がさまざまなので，レビューの方向がばらばらになってしまうことである．これを解決するために，我々は表現不良のカテゴリを作成し，レビューの指針として学習者に示した．これを表 1 に示す．

表 1 目的の表現レビューのカテゴリ

構造	構成
	粒度
	過不足
表現	目的記述
	日本語記述

このカテゴリーは，我々が 30 数名の HCP チャートをレビューして，そのパターンをまとめたものである．これらのカテゴリーは，構造とその表現という大きな視点からまとめられている．これらは，2 節で示したように，プログラムによって目的を表現するときの大きな 2 つの視点でもある．

個別のカテゴリーを説明する．まず，構成不良は，構造に問題があり，可読性を損ねているものや，項目の移動や，分解・再構成が必要なものが分類される．

次に，粒度不良は，粒度がそろっていないので，新しいレベルを追加・削除することにより，粒度を調整すると，よりよい構造になるものが分類される．

過不足不良は，下位レベルの記述不足により、目的の実現方法が明確でないもの，もしくは，下位レベルの記述過剰により，上位レベルの目的の見通しを損なっているものが分類される．

目的記述不良は，目的がかかれるべき場所に，手段が記述されているものが分類される．ただし，目的は上位レベルからみると手段なので，正確には，上下レベルの目的-手段の相対関係が適正でないものである．上位レベルに手段が記述されていた場合，構造の変更を伴う場合があるが，その場合は，構造不良として分類する．

最後に日本語記述不良は，記述が不十分，または不適切なので、記述の明確さを損ねているものが分類される．例えば，説明不足，用語定義不足，てにをは不良などがこれにあたる．

このレビューカテゴリに従って，相互にレビューを行うことによって，お互いに共通の視点を持って表現の研鑽を行うことができるようになる．さらに，このカテゴリーを学習者と指導者が共有することで，指導者の指導が学習者に伝わりやすくなる．

4. 実 験 授 業

本稿が提案する教育手法に基づくカリキュラムを

表 2　達成度レベルと評価の指針

達成度レベル	自然言語	構造化手法	オブジェクト指向
達成目標	自然言語を用いて，プログラムの目的を表現できる．	構造化手法を用いて，プログラムの目的を表現できる．	オブジェクト指向をも用いて，プログラムの目的を表現できる．
評価の指針	構造表現 目的記述	ブロック表現 ブロックコメント記述 手続き表現 手続き名の記述 変数表現 変数名の記述	クラス構造表現 クラス名の記述

表 3　カリキュラム

事前試験	アプリケーションの設計，実装	10 時間
講座	HCP チャートによる設計演習	4 時間
	構造化手法による プログラミング演習	10 時間
	オブジェクト指向入門	4 時間
	カプセル化によるデータ抽象	4 時間
	クラス設計の基礎	6 時間
	継承を使ったフレームワーク入門	2 時間
最終試験	アプリケーションの設計，実装	10 時間
	設計レビュー能力判定	3 時間

作成し，実務者を対象に実施した．本節では，その実施体制について述べる．

4.1　カリキュラム

前節までの議論を踏まえ，実験授業のカリキュラムでは，目的の表現の段階として，自然言語，構造化手法，オブジェクト指向という 3 つのレベルを設定し，それぞれの学習目標と評価の指針を設定した(表 2).

この達成レベルのうち，自然言語と構造化手法レベルの教育カリキュラムは，竹田のカリキュラム[6)]を採用した．これは，プログラムの書法や，モジュール構造など，プログラムの表現に主眼を置き，設計文書を書かせ，レビューすることで技術者同士のコミュニケーションを図るように設計されているという理由からである．また，オブジェクト指向レベルの教育カリキュラムは，オブジェクト指向の考え方を議論するように設計された「オブジェクト指向哲学」[5)] を採用した．

これら 2 つの教材をベースに目的の表現能力の育成を強化して，オブジェクト指向教育のカリキュラムへつながるように改良し，試験を除いて 30 時間程度のカリキュラムを作成した．これを表 3 に示す．

このカリキュラムでは，事前課題と最終課題を受講者に課した．これは，受講者の達成度評価を行うためと，対象者を絞るためである．このカリキュラムでは，基礎的なプログラミングは理解していることを前提としている．

事前課題と最終課題はともに同じ課題で,「列車の座席予約システム」の作成を行うものである．10 時間という時間を設定し，できる範囲での仕様を受講者自ら作成し，設計・実装を行うという課題である．

4.2　対　象　者

IT 企業の実務者を対象に募集を行い，受講者を集めた．41 名の募集があり，事前試験を行った結果、合格者は 31 名であった．そのうち全日程に参加した受講者は 23 名である．この 23 名を本稿における評価の対象とする．

受講者の特性を示すものとして，IT 関連業務の経験年数を表 4 に示す.

表 4　受講者の IT 関連業務の経験年数

5 年未満	9
5 年～10 年	6
10 年以上	6
不明	2

また，事前課題としての成果物と，受講初日の様子から受講者のプログラミング能力を分析すると，

- 何らかの言語を習得済みである
- コンパイルエラーなどの対処はできる
- 参照の操作を理解している
- オブジェクト指向は知識として知っている

という傾向があった．

5.　評価・考察

本節では，講座を受けた受講者の成果物と，アンケートから，本稿で提案する教育手法の有用性を考察する．

5.1　成果物による考察

ここでは，ある受講生の成果物を例に取り上げて，達成度を考察する．

初めに，HCP チャートについて，事前課題と事後課題を取り上げ比較する．成果物を図 7，図 8 に示す．(ここで掲載する HCP チャートでは，処理を○で，分岐処理を→で，繰り返しを◎で，階層構造をインデントで表現している．また，詳細部分を割愛している.)

これらを比較すると，構成や目的記述における表現能力が向上していることがわかる．構成において，

```
○座席を予約する
  ○予約者の名前を入力する
  ○人数を入力する
  ○喫煙禁煙の希望を入力する
    → (人数が 1 人の場合)
    ○窓側通路側の希望を入力する
  ○指定された人数にあった、空いている席を検索する
    → (人数の判断をする)
        → (1 人の場合)
      ◎行でまわす（喫煙：1-10、禁煙：11-25）
          →席が確定していない場合
    … 途中割愛

  ○座席予約要素を出力
```

図 7 事前課題の HCP チャート

```
○列車座席予約をする
  ○アプリケーションの開始を知らせる
  ◎コマンドを実行する
    ○コマンド入力を受ける
      ○コマンド入力プロンプトを出力する
      ○コマンド入力を受け取る
    ○コマンドを実行する
      →（座席を予約する場合）
          ○顧客名を受け取る
          ○人数入力を受け取る
          ○窓側／通路側指定を受け取る
          ○喫煙／禁煙指定を受け取る
          ○予約をする
          ○予約結果を表示する
      →（座席状況一覧の場合）
          ○座席状況一覧表示
  ○アプリケーションの終了を知らせる
```

図 8 事後課題の HCP チャート

事前課題の表現では，処理以外は一段のみの平坦な構造なのに対して，事後課題では「コマンド入力を受け取る」や,「コマンドを実行する」などの一段大きな目的を追加することにより，処理にまとまりができるようになった．

目的記述においては,「指定された人数にあった，空いている席を検索する」といった，手段に使い表現が排除され,「座席予約要素を出力」などの明確でない表現が「座席状況一覧表示」という目的の記述に変化している．さらに，列車や顧客などの重要な概念が表現されるようになった．

次に，クラス設計について考察する．この受講者は，事前課題ではプログラムにクラスを用いていなかったので，事後課題における成果物のみ図 9 に

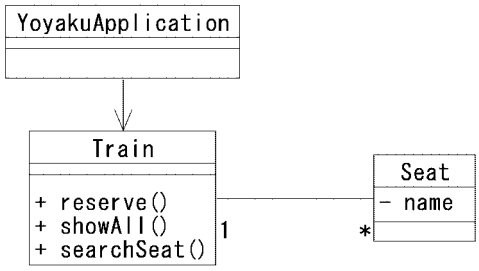


図 9 事後課題におけるクラス設計

示す．

このクラス設計は前述の事後課題の HCP チャートのシステム図 8 と対応している．ソースコードは割愛するが，実際には，Train クラスにおいて，必要最低限のインターフェイスのみ公開とし，データ構造を隠蔽している．このシステムにおいて，座席と列車が重要な概念であることを考えると，妥当なクラス設計だといえよう．

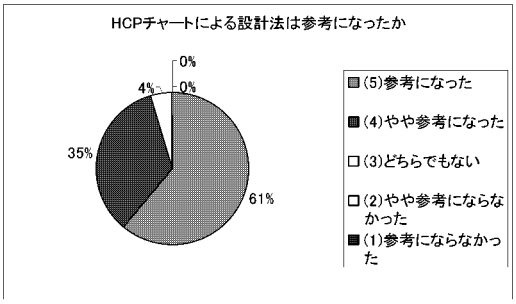
さらに，ここで注目するべきは HCP チャートとの関連である．Train クラスの公開インターフェイスは，HCP チャートの目的記述と対応している．さらに，ここでクラスとして抽出されている列車と座席に関しても，HCP チャートにおける，大きな目的の目的語から導き出されている．このことから，この受講者のクラス設計において，HCP チャートで表現された目的の構造が深く関連しているといえる．

5.2 アンケートによる考察

講座実施後に行ったアンケート結果 (対象者は最後まで受講した 23 名) より，興味深いものを抜粋して図 10 に示す．

「HCP チャートによる設計法は参考になったか」の問いに対して，参加した受講者の多数が HCP チャートによって目的の表現をし，他人に伝えることのできるプログラムを書くということを受け入れたといえる．自由記述にも「目的と手段を常に意識するようになった.」「動くことも重要であるが，他の人が見ても分かるソース作りがとても重要であることが気づけた.」「今までは，コンパイルが通ることを目的としてプログラミングしていたが，ソースを始めて読む人の立場になってプログラミングするようになった.」という記述がみられた．

本稿のテーマである「目的の表現からのアプローチはオブジェクト指向の理解に必要か」との問いに対しても，多数の受講者が肯定的な答えをしてい



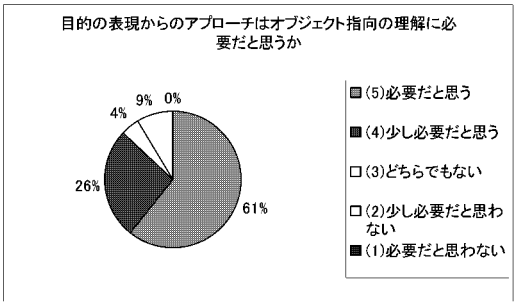


図 10 アンケート結果

る．自由記述にも「あいまいだったオブジェクト指向が，自分の中で，目的という概念が加わったことにより，はっきりしてきた．そして,（目的を主眼に設計した場合）分かりやすさは圧倒的に違うと思った.」「オブジェクト指向プログラミングにおいても，構造化プログラミングの手法は重要であることが分かった.」など，受け入れやすい考え方であることが考察される．

「クライアントのさまざまな要求をイメージしやすい手法だと思った.」「プログラムレベルだけではなく，上流工程でも明確な目的の定義が必要だと感じました.」などの意見もあった．これは，この手法が上流工程のオブジェクト指向教育の一歩となっていることを示唆していると思われる．

6. ま と め

本稿では，オブジェクト指向教育の一手法として「目的の表現」に注目したアプローチを紹介した．2 節で，プログラムでの論理的なブロックとそのコメントにより表現することが，オブジェクト指向での表現と関連していることを述べた．3 節では，目的の表現を教育する手法として，初歩の段階として，HCP チャートなどを利用した自然言語での表現から教育することと，受講者による相互レビューの手法とカテゴリについて述べた．4 節で，実験講座の概要を示した後，5 節において，受講者の成果物とアンケートの考察の結果から，提案手法は実務者に受け入れられたことを示した．

目的の表現を客観的に評価する方法の考案と，この手法を利用して教育できる講師の育成を今後のテーマとしたい．

謝辞

本稿でとり上げた実験授業は，経済産業省平成 14 年度高度 IT 人材育成システム開発事業の一環として行われた．尽力くださったクレデンシャル総合研究所の皆様に感謝いたします．

参 考 文 献

1) Brian W.Kernighan, Rob Pike. The Practice Of Programming. Addison Wesley Longman, Inc., 1999.
2) Leslie J.Bringgs, Kent L.Gustafson, Murray H.Tillman. Instructional Design Principles and Appplications. Educational Technology Publications, Inc., 1991.
3) Patricia L.Smith, Tillman J.Ragan. Instructional Design. Wiley & Sons, Inc., 1999.
4) 花田收悦. プログラム設計図法. 企画センター, 1983.
5) 松澤芳昭，岡田健，中鉢欣秀，大岩元．オブジェクト指向技術者養成のためのカリキュラム．情報処理学会研究会報告 (CE-64-1), pp. 1–8, 2002.
6) 竹田尚彦，大岩元．プログラム開発経験に基づくソフトウェア技術者育成カリキュラム．情報処理学会論文誌, Vol. 33, No. 7, pp. 944–954, 1992.
7) 長野宏宣，浅見秀雄，忠海均．階層化プログラム設計図法－ HCP チャート－．企画センター, 1992.

プログラミング入門教育におけるモデルによるプログラムの状態表現

青山　希[†]　松澤芳昭[†]　杉浦　学[†]
川村昌弘[†]　大岩　元[††]

プログラミングの導入教育において，学習者が自分でプログラムを作成することができるようになるためには，制御構造を理解し，プログラム実行時の動作を正しくイメージする必要がある．さらにプログラムの 実行時の状態を把握し，処理が実行された瞬間の状態遷移を正しくイメージできる必要がある．しかし，学習者がプログラムの状態を理解するための，簡潔で，わかりやすい記法はない．

これに対し，我々はプログラム実行時の，オブジェクトの状態を図示する 3 種類の説明モデル「表モデル」,「入れ子モデル」と「参照モデル」を提案し，実際にプログラミング教育の場で試用してみた．本稿ではこれらのモデルについて，その有効性について考察する．

Expression of State of Program with Models In Introduction to Programming Courses

Nozomu Aoyama,[†] Yoshiaki Matsuzawa,[†] Manabu Sugiura,[†]
Masahiro Kawamura[†] and Hajime Ohiwa[††]

In introduction to programming courses, a leaner must understand the control structure of programming and must be able to imagine what is done and how the state of the program is changed at the time of program execution. However, there is no appropriate notation for the learner to understand the state of a program.

For this problem, we propose three kinds of explanation models named "Table Model", "Nest Model", and "Reference Model" to illustrate the state of program at the time of execution. We have tried them for programming instruction, and describe how these models are useful.

1. はじめに

プログラミングの導入教育において，学習者が自分でプログラムを作成することができるようになるためには，まず制御構造を理解し，プログラム実行時の動作を正しくイメージできる必要がある.[6] しかし，それだけでは自分でプログラムを作成するためには不十分である．それに加え，プログラムの実行時の状態を把握し，処理が実行された瞬間の状態遷移を正しくイメージできる必要がある．

プログラミング教育の初期段階において，学習者が作成したプログラムが学習者の思うように動かず，なぜそのような結果がでるのかの検討がつかなくなることがままある．これはプログラム実行時の動作と，その時点での状態を正しくイメージできていないためであると考えられる．動作と状態を正しくイメージでき，プログラムの 1 行 1 行で何が起きているのかを把握できるようになって初めて，学習者はプログラムの実行とともにどのような処理が行われているのかを理解することができ，同時に，プログラムのデバッグを行うことができるようになる．

プログラムの処理の構造を理解するための記法としては，処理の流れを図解するフローチャートや，目的の視点でプログラムの構造を図解する HCP チャート[5] などがある．しかし，プログラムの実行時の状態を正しくイメージするための図解は十分であるとはいえない．

ある瞬間のプログラムの状態を図示するための手法として一般的によく用いられるものに，図 1 のよ

† 慶應義塾大学 政策・メディア研究科
Graduate School of Media and Governance, Keio University
†† 慶應義塾大学 環境情報学部
Faculty of Environmental Information, Keio University

番地	内容
100	101
101	'a'
102	
103	
104	

図 1　メモリイメージの図解サンプル

うなコンピュータのメモリイメージをモデル化したものがある．しかし，コンピュータの仕組みを意識したことのない学習者にとって，最初から実際のメモリの仕組みを想像するのは困難である．同時に，講師の側からしても，学習者にとって想像しにくいメモリイメージを用いた図解で，プログラムのデータの移り変わりを説明するのは困難である．

筆者らは，この問題を克服するために，3 種類の説明モデル「表モデル」「入れ子モデル」「参照モデル」を提案する．これらのモデルは，変数表を拡張した記法を用い，構造化プログラミングからオブジェクト指向プログラミングまで一貫した記法でプログラムの実行時の状態を図解することができる．

慶應義塾大学大岩研究室では，構造化プログラミングからオブジェクト指向プログラミングまでを教えるカリキュラムを作成し，大学でのプログラミング教育やシステムエンジニア育成のための新入社員教育などの場で実践している．今年度版のカリキュラムにおいてこれらのモデルを用いた教育を実践してみた．本稿ではこの実践の成果を報告する．以下，2 章では 3 種類のモデルの記法について具体例を使って述べ，3 章では試用した結果とモデルの有効性についての考察を行い，4 章で今後の課題について述べ，5 章でまとめを行う．

本稿ではまず，3 種類のモデルの記法について述べる．次にそれらを実際に使用した結果を示し，その有効性と今後の課題について述べる．

2. モデルの概要

2.1 表 モ デ ル

表モデルはプログラム実行時の変数と値の関係を表すためのモデルである．学習者は，表モデルを通じて，変数が評価されて値になるという考え方を身につけることができる．

変数と値の関係を表すためのモデルとして一般的

```
void run(){
  int a;                    //(1)
  a = 5;                    //(2)

  int[] b = new int[3];     //(3)
  b[0] = 4;                 //(4)
}
```

図 2　表モデルのサンプルプログラム

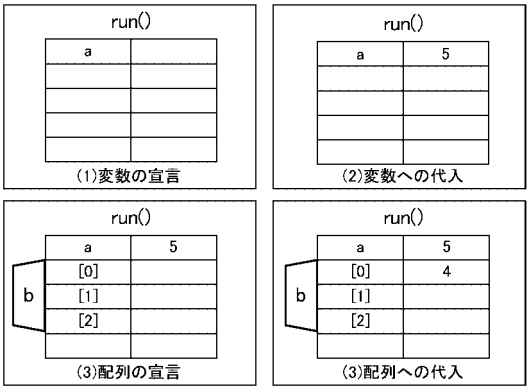


図 3　表モデルのサンプル

なものに，変数表がある，表モデルは変数表を拡張し，記法を定義したものである．表モデルでは，メソッドごとに変数の値が移り変わっていく様子を図に表す．通常の変数表と異なる点は，配列が変数のまとまりであることをわかりやすくするために，変数まとまりに配列が格納されている変数名のタブをつけて表すことである．

図 2 のようなプログラムがあったとき，(1) から (4) の各行が実行されたときのプログラムの状態を表す表モデルは図 3 の各状態に対応する．

2.2 入れ子モデル

オブジェクト指向プログラミングの導入教育において問題となるのが，オブジェクトがさまざまな値を持つ"もの"である，というイメージをつかむのが難しいことと，さらに，アドレス参照の考え方を身につけなければならないことの 2 つである．プログラミング言語として Java 言語を使った場合，学習者は 2 つの新しいことを同時に理解しなければならないため，個々の概念を余計に難しく感じてしまう．

入れ子モデルはオブジェクトの状態を図示するための，表モデルを拡張したモデルである．このモデ

```
void run(){
  Section section = new Section();     //(1)
  section.name = "sales";              //(2)

  Employee employeeA = new Employee();
  employee.name = "Yamada";            //(3)

  section.employees[0] = employeeA;    //(4)
}

class Section {

  String name;
  Employee[] employees;

  public Section(){
    employees = new Employee[3];
  }
}

class Employee {

  String name;
}
```

図 4　オブジェクトの構造をつくるサンプルプログラム

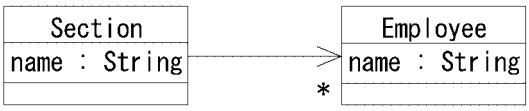


図 5　サンプルプログラムのクラス構造

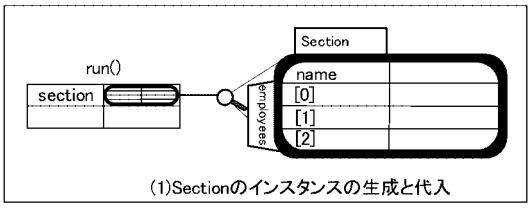


図 6　入れ子モデルサンプル (1)

ルは，参照の理解の問題を避けるため，参照の概念を理解しなくてもオブジェクトのイメージがつかめるよう考えられている．

入れ子モデルではオブジェクトは変数に代入されると，参照が渡されるのではなく，オブジェクトがコピーされて変数の値として格納されると考える．これは初学者にとって，オブジェクトがコピーされて変数の値として格納されるという考え方が，参照に比べて理解しやすいと考えたためである．記法も

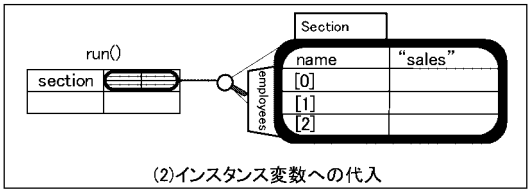


図 7　入れ子モデルサンプル (2)

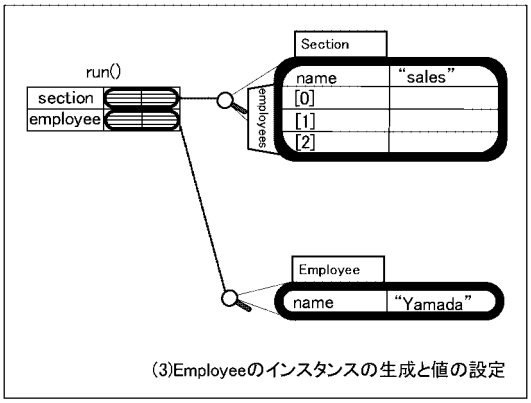


図 8　入れ子モデルサンプル (3)

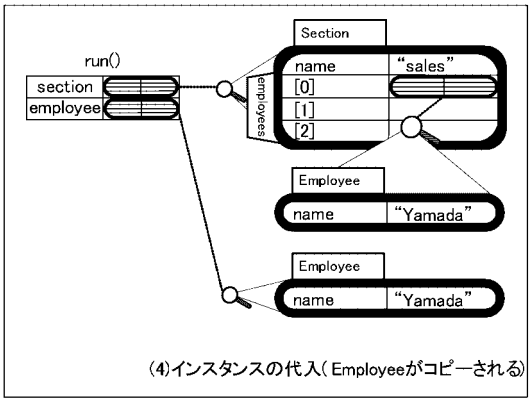


図 9　入れ子モデルサンプル (4)

それに合わせて，変数に対応するオブジェクトが値として中に入っているようなイメージのものになっている．(図 6)

会社組織における部署と社員を管理するサンプルプログラム図 4 が，図 5 のようなクラス構造を持つとき，(1) から (4) の各行が実行されたときのプログラムの状態を表す入れ子モデルは図 6 から図 9 のようになる．

入れ子モデルでは代入式が評価されるとオブジェクトがコピーされると考えるので，同じ参照を持つ

図 10　オブジェクト図のサンプル

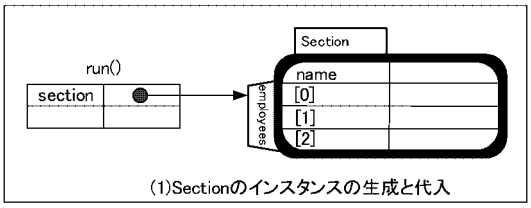


図 11　参照モデルサンプル (1)

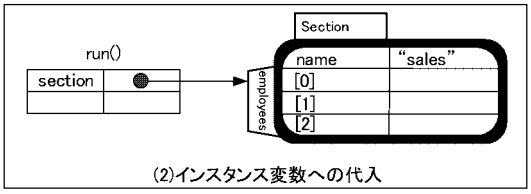


図 12　参照モデルサンプル (2)

2 つの変数があったとき，その関係を説明することができない．そのため，入れ子モデルを用いて説明する段階では参照の理解が必要な例題や課題は教育カリキュラムの中で扱わない．

2.3　参照モデル

参照モデルは，入れ子モデルを通じてオブジェクトのイメージがつかめた学習者に参照の考え方を教えるためのモデルである．参照モデルでは，変数に格納されている他のオブジェクトへの参照を丸と矢印で表す．

オブジェクト間の参照を表すための記法として一般的なものに，UML[3] に定義されているオブジェクト図がある．図 4 の (4) が実行されたときのオブジェクト図は図 10 のようになる．参照モデルがオブジェクト図と異なる点は，これまでの表モデルや入れ子モデルの記法を踏襲することで，変数と，変数に格納されるオブジェクトへの参照の関係を明確にしている点である．

入れ子モデルのときと同じプログラム (図 4)(図 5) を用いて参照モデルでの表現を説明する．(1) から (4) の各行が実行されたときのプログラムの状態を表す参照モデルは図 11 から図 14 のようになる．

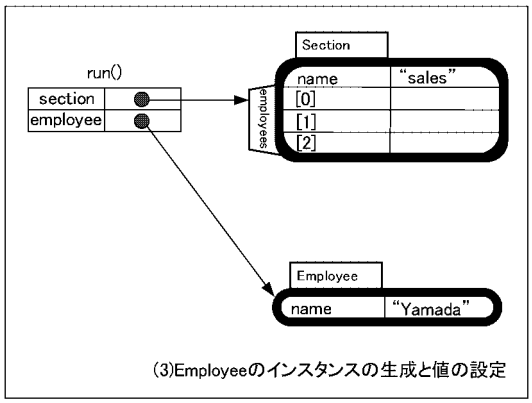


図 13　参照モデルサンプル (3)

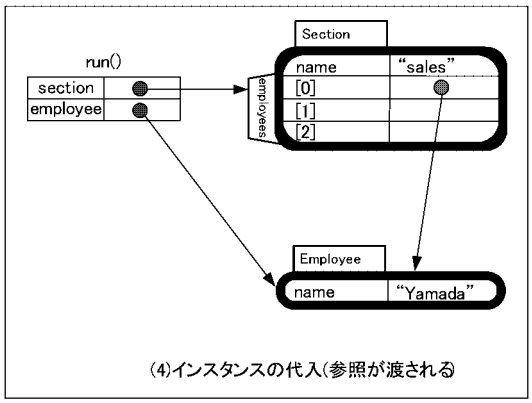


図 14　参照モデルサンプル (4)

表 1　カリキュラム実践環境

新入社員教育	プログラミング初心者	60 名
大学における授業	プログラミング初心者	約 80 名
社会人向けプログラミング講座	実務経験者 (オブジェクト指向は未修得)	31 名

3.　モデルの評価

筆者らは，本稿で提案する 3 種類のモデルを導入したプログラミング教育カリキュラム「人にやさしいプログラミングの哲学」を大学での授業，新入社員教育，社会人向けプログラミング講座の 3 つの場で実践した．実践を行った環境の詳細を表 1 に示す．本章では，その評価を行い，理解を助けるツール，コミュニケーションのためのツール，評価のためのツール，の 3 種類の観点からその有効性を考察する．

3.1　理解を助けるツールとしてのモデルの有効性

昨年度までに実施していたプログラミング教育カリキュラムでは，多くの学習者が概念の理解に苦し

む項目がいくつかあった．具体的には戻り値のある手続き，クラスとインスタンスの関係，オブジェクトのネットワーク構造の構築などである．今年度実践したカリキュラムでは，これらの概念の理解について昨年に比べ学習者の苦労が軽減されたという感触があった．理由としてはカリキュラム自体の改善もあるが，概念の説明にモデルを使用したことも理由のひとつである．

モデルの導入が学習者の理解を助けた理由として，表モデルをベースとする一貫した記法により，式が評価されて値になるという概念が学習者に定着したことが挙げられる．例えば，戻り値のある手続きに関して言えば，直接はモデルと関係がないが，表モデルで変数が評価されて値になったように手続きが評価されて値になる，という説明で昨年に比べ多くの学習者が概念を理解することができた．このことは，オブジェクトの構造を構築するという項目を説明する際にも同様の現象がみられた．

また，オブジェクト指向の理解についても，入れ子モデルと参照モデルの導入により，実体であるインスタンスが明瞭に理解され，それを定義するクラスの役割をはっきりさせることができた．その結果，両者の違いが理解できない学習者は昨年に比べ少なかった．

上に述べたようなことは，学習者を対象とするアンケートやヒアリング，課題提出の際の感想からも見てとることができた．

3.2 コミュニケーションツールとしてのモデルの有効性

通常，学習者がプログラミングの実習を行う中で何らかの問題につきあたっているとき，その原因がどこにあるのかを講師が瞬時に判断するのは難しい．また学習者自身にとっても，問題の原因に気づくのは難しい．例えば，学習者がオブジェクトを追加するアルゴリズムがわからないと言ったとしても，その原因が，アルゴリズムが理解できていないことなのか，クラスとインスタンスの関係がわからないことなのか，手続き，式と評価などより基本的な項目が理解できていないことなのかは瞬時にはわからない．

そのようなときでも，モデルを用い，プログラムの 1 行ごとの状態のイメージを学習者に書かせることで，どの段階で理解が曖昧になっているのかを把握することができる．例えば，入れ子モデルを正しく書くことができているかどうかで，クラスとインスタンスの関係まで理解できているかどうかがわかるといった具合である．

また，原因を特定した後の，学習者への説明のツールとしても，これらのモデルは有効である．これは，3 つのモデルが，1 行ごとの処理によってプログラムの状態がどのように変化するのを明確に示すことができるためである．例えば，new 演算子が評価されてインスタンスが生成される，というのは学習者にとっては抽象的で理解しにくい事柄である．しかし，入れ子モデルや参照モデルを使えば，インスタンスが生成されるということがどのようなことであるのかを，図で明確に示すことができる．

以上のことから，3 つのモデルは，講師が学習者が抱える問題に対して個別に対応する際のコミュニケーションのツールとして有用である．

3.3 評価ツールとしてのモデルの有効性

プログラミング教育の場において常に問題となることのひとつに，プログラミングに関する概念の，理解度の測定が挙げられる．一般的にプログラミングの概念理解を測るための試験では，概念の説明を文章で記述させる，実際に概念を使ったプログラムを書かせる，などの方法がとられている．

しかし，プログラミングに関する概念は，言葉で説明できるだけでは理解として不十分であり，実際にプログラムが実行されていく中でどのような処理が行われ，どのように状態が遷移するのかを理解している必要がある．またプログラムを実際に書かせる試験には問題に対する普遍的な正解がない．そのため学習者の解答が多種多様なものとなり，必ずしも解答が理解を測りたい概念を利用したものだけとは限らないので評価が困難である．

この問題を解決するために，理解を測りたい概念を使ったソースコードを学習者にみせ，概念と関係がある段階のプログラムの状態をモデルとして記述させるという方法が有効であると考えられる．概念を理解していない学習者は，ソースコードからプログラムの実行時の状態をイメージすることができないためである．例えば，クラスとインスタンスの関係を理解しているかを測るために，オブジェクトを生成し，操作するサンプルプログラムの参照モデルを書かせる．このような方法により，概念の理解度を測る試験に正解を設定することができるので，容易に概念を理解しているかを測ることができる．

モデルを書かせることによって，概念の理解を測るという試みは，前述した実践の場で実際に行われた．モデルを書かせると，概念の理解ができていない学習者は一目瞭然であった．それゆえ，3 つのモデルは評価のためのツールとしても有用であると考えられる．

4. 今後の課題

この章では実際にモデルを使ったカリキュラムを実践し，明らかになった問題点と，それに関する考察を行う．

まず，入れ子モデルについて考察する．入れ子モデルはオブジェクトのイメージをつかむという点では，学習者の理解を助けたが，一方で，オブジェクトが代入の際にコピーされるという入れ子モデルの考え方は，実際に Java 言語のプログラムで起こる動作とは異なるため，学習者を混乱させた面もあった．

教育カリキュラムでは，入れ子モデルでオブジェクトがコピーされると教えたすぐ後に，参照モデルが登場し，実はオブジェクトは入れ子になっているのではなく参照を持っているという説明を行う．結果としてまだ入れ子モデルの理解が完全ではない状態で，実は参照を持っているという説明をすることになり，理解に不安を感じている学習者を混乱させる原因となった．また，どの実践の場においても，学習者の中に若干他よりもプログラミング経験の多い学習者がいた．そのような学習者から本当はコピーされないのではないかというような質問がでると，実は入れ子モデルは事実と違うということを説明せざるを得ず，これも学習者を混乱させる原因となった．

しかし，クラスとインスタンスの関係と参照の概念を同時に学ぶことが学習者にとって難易度が高いのは事実である．そのため，今回実践した場よりプログラミングに関する知識の少ない学習者が多いようなときには，入れ子モデルはやはり必要であると思われる．さらなる実践を重ね，このような入れ子モデルの有効性と弊害について，検証することは今後の課題である．

実際のプログラムの動きと異なることの弊害は，表モデルにおける配列に関しても同様のことが言える．Java 言語における配列は参照型であるため，表モデルで説明できない場合がある．また，表モデルは二次元配列については考慮されていない．これらの点に関しても今後検討が必要である．

5. おわりに

本稿ではプログラム実行時の状態を図示するモデルの提案とその評価について述べた．プログラム実行時の状態を図示する試みとして，現在，Kent Beck らが開発中の Object Spider や，SmallTalk ベースのオブジェクト指向プログラミング教育環境である Squeak[2] などがあり，今後このような試みは広まっていくと思われる．本稿で述べた 3 種類のモデルも今後改良を加え，モデル作成用のツールなどの環境も整えたいと考えている．

謝辞

本研究を進めるにあたり (株)EXA の児玉公信様，井関知文様を始めとする技術部の皆様には大変お世話になりました．この場を借りて感謝いたします．

参 考 文 献

1) Gerald Jay Sussman,Harold Abeson,Julie Sussman. 計算機プログラムの構造と解釈 第二版. ピアソン・エデュケーション, 1999.
2) Mark J. Guzdial,Kimberly M. Rose. Squeak 入門. エスアイビー・アクセス, 2003.
3) Object Management Group. UML 仕様書. ASCII, 2001.
4) 松澤芳昭，岡田健，中鉢欣秀，大岩元．オブジェクト指向技術者養成のためのカリキュラム．情報処理学会研究会報告 (CE-64-1), pp. 1–8, 2002.
5) 竹田尚彦，大岩元. プログラム開発経験に基づくソフトウェア技術者育成カリキュラム．情報処理学会論文誌, Vol. 33, No. 7, pp. 944–954, 1992.
6) 長谷川聡, 山住富也, 小池慎一．プログラミング教育における制御構造のイメージと理解度について．情報処理学会論文誌, pp. 1180–1183, 1998.

プログラム設計教育における HCP チャートのレビュー手法

松澤　芳昭†　　杉浦　学†　　大岩　元‡

†慶應義塾大学　政策メディア研究科　　‡慶應義塾大学　環境情報学部

macchan@crew.sfc.keio.ac.jp

概要

我々は HCP チャートを用い，指導者や学習者同士によるレビューを通したプログラム設計指導を行っている．その際，レビューの方針と内容が定まらないと以下の問題が生じる 1)レビュアーによって指摘項目のばらつきが生まれる 2)レビューした事柄を体系化する方法がないため，レビューの意図を的確に伝えることが難しい 3)初学者が本質的なレビューを行うことが難しい．我々はこれらの問題を解決するため，実際の指導経験に基づいて，設計が「第三者に十分に伝わるか」という視点でレビュー項目を分類した．この分類を用いてレビューを行うことで，指導者と学習者間で価値観を共有してコミュニケーションができるので，レビューの意図が明確に伝えやすくなるという成果を得た．

1. はじめに

我々は，他人(第三者)に分かりやすいプログラムが書けるようになることを目的としたプログラミング教育のカリキュラムを実施している[1][2]．これらのカリキュラムでは，入門段階から学習者に HCP チャート[3][4]を用いてプログラムの設計を行わせる．HCP チャートは，プログラムを分かりやすく設計し，他者に分かりやすく表現することをねらった図法である．

我々のカリキュラムでは，学習者の作成した設計文書に対して，指導者や他の学習者がレビューを行う．しかし，レビューの方針と内容が定まらないと以下の問題が生じる．

1) レビュアーによって，重視するレビュー項目にばらつきが出てしまうので，価値観の共有が難しいこと．設計でレビューすべき事柄は，機能，インターフェイス，アルゴリズム，記法など多岐にわたる．

2) 熟達した指導者は，設計の不備を経験的に察

を体系的に説明するのが難しいこと．その場合，学習者が，受けたレビューを一般化して理解し，他の問題に適用できるような能力を身に付けることが難しい．

3) 学習者にもレビュー能力が必要である．よい設計文書を書けるようになることは，よい自己レビューができるようになることにほかならない．しかし，学習者同士のレビューにおいて，初学者が本質的なレビューを行うことは難しく，何らかの指標が必要になる．これがないと，彼らのレビューはインターフェイスや「てにおは」などの細かい非本質的な項目にとどまってしまう．

A Review Method for HCP Chart
Yoshiaki Matsuzawa†, Manabu Sugiura† and Hajime Ohiwa‡
† Graduate School Media and Governance , Keio University
‡ Faculty of Environmental Information, Keio University

我々は，こうした問題を解決するために，実際の指導経験に基づいて「第三者に十分伝わるほど分かりやすいか」という点に焦点を絞ったレビュー項目の分類法を考案した．

これにより，指導者のレビューの視点が絞られ，体系化された形で学習者に伝えることができる．レビューの結果を学習者が分類に基づいて理解することで，自己レビューする能力を育成することができる．これは，学習者同士のレビューを的確に行いやすくすることにもつながる．

本稿では，そのレビューの方法について述べる．

2. レビュー項目の分類

我々の提案するレビュー項目の分類は「日本語不良」「目的不明確」「構造不良」「粒度不良」の 4 つである．列車の座席予約システムを具体例として，各分類の詳細を説明する．

2.1. 日本語不良

チャートの各項目の記述が日本語として意味不明なもの．例えば，「行でまわす」「希望人数の指定座席の結果を表示する」「1 つの座席ごとの予約状況を列ごとに表示する」といった記述は，理解不可能な日本語であるので，日本語不良である．

2.2. 目的不明確

目的が明確でない項目．HCP チャートは目的の階層構造を記述するため，各項目は，目的が明確に記述されている必要がある．例えば，図 1 の 3 行目「一桁の場合，スペースを入れる」という記述は，上位の「行を表示する」という目的を考慮しても，手段としての目的が分からない．

同じ記述でも目的不明確とならない場合がある．例えば，図 2 は先ほどの指摘に対して，目的が明確になるように修正したチャート(実は，この項目は「行番号を表示する」の手段であった)であるが，この場合は，目的が明確である．

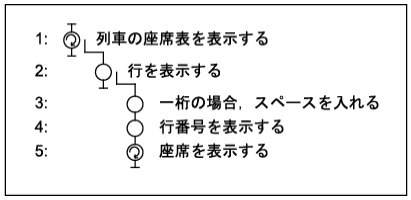


図 1 目的不明確のチャート例

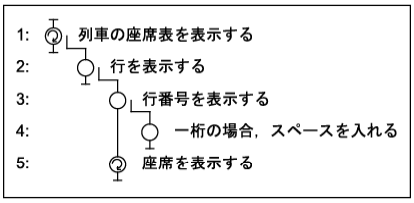


図 2 図 1 のチャートを改善したもの

2.3. 構造不良

ある項目とその下位レベルの項目が目的-手段の関係になっていないもの．構造不良はさらに「矛盾」と「不足」の 2 種類に分類される．

2.3.1. 矛盾

項目が，上位の目的の手段として明らかにそぐわない（矛盾している）もの．この不良は目的-手段の関係が逆になっているケースが多い．例えば，図 3 において，6 行目の「座席を予約する」は，その上位レベルである 4 行目「座席の検索をする」の手段として，明らかに矛盾している．

2.3.2. 不足

上位の項目の目的が，提示された下位の項目の手段群では達成不可能だと考えられるもの．例えば，図 3 の 1 行目「列車の座席を予約する」は，下位レベルの 4 つの項目によって達成されると記述されている．しかし，それらの項目の中に，実際に予約するという手段が記述されていない(4 行目「検索する」の中に入ってしまっているため)．

2.4. 粒度不良

あるレベルにおいて，適正な粒度で項目群がまとめられていない場合．例えば，図 3 の 2 行目と 3 行目は，同レベルの他の項目と比較して，粒度が細かすぎる．これらの項目は，図 4 の 2 行目「予約のための情報を入力する」のように，まとめた項目を作ることで粒度が適切に調整される．

3. レビューの優先度

レビューを行う際には，優先度が高い日本語不良から順に，目的不明確，構造不良，粒度不良と見ていく．これは，優先度が高い分類の不良としてレビューされた項目は，優先度の低い分類の不良かどうかを判定するのが不可能であるためである．例えば，「行でまわす」という項目がある場合，日本語不良と指摘される．そのため，優先度が低い分類の不良，例えば目的不明確かどうかを判断することは不可能である．

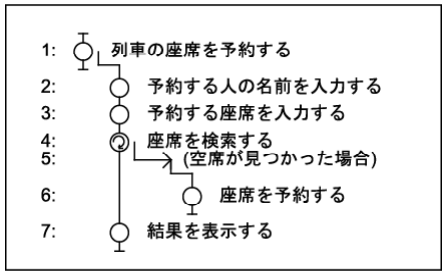


図 3 構造・粒度不良のチャート例

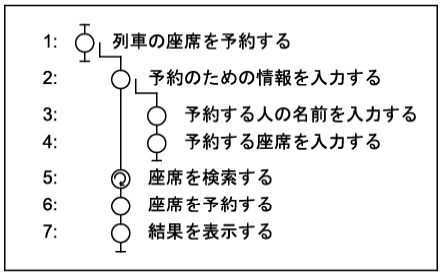


図 4 図 3 のチャートを改善したもの

4. 考察

この方法を用いて，実際に設計教育を行ったところ，まず，レビュアーによる指摘項目のばらつきが激減した．現在，この分類に熟達した指導者ならば，ほぼ同一のレビューを行うことができる．

学習者が相互レビューを行う際に，お互いが分類を意識してコミュニケーションするため，レビュアーの意図が作成者に伝わりやすくなった．これは，指導者と学習者の場合も同様であった．

5. おわりに

本稿では，HCP チャートによる設計のレビューの視点となるレビュー項目の分類法を提案した．この分類を用いることにより，指導者や学習者に共通の視点ができるので，指導者の意図が学習者に伝わりやすくなり，学習効果が得られやすくなることが考察された．

我々は，設計の目的を明確にし，構造や粒度を適切に調整するという指導は，オブジェクト指向指向の入門教育としても重要であると考えている．

最後に，このレビュー方法のレビューをしていただいた，愛知教育大学の竹田尚彦先生に感謝します．

参考文献

[1] 松澤芳昭，青山希，杉浦学，川村昌弘，大岩元．「目的の表現」に注目したオブジェクト指向プログラミング教育とその評価．情報処理学会研究会報告(CE-27-11)，pp.77-84，2003.

[2] 竹田尚彦，大岩元．プログラム開発経験に基づくソフトウェア技術者育成カリキュラム．情報処理学会論文誌，Vol33，No.7，pp.944-954，1992.

[3] 長野宏宣，浅見秀雄，忠海均．階層化プログラム設計図法－HCP チャート－．企画センター，1992.

[4] 花田收悦．プログラム設計図法．企画センター，1983.