N-011

プログラミング演習の総合支援システムの概要
Total Support System for Programming Course

内藤　広志 †
Naito Hiroshi

1. はじめに

大阪工業大学情報科学部では，C 言語や Java 言語を用いたプログラミング演習科目を多く設け，学生が基本的なプログラミング能力を修得できるように努めている。特に，基本的なプログラムを多く作成することが重要だと考え，毎回 5〜9 個の課題を出題している。履修学生数が 450〜800 名と多いこともあり，課題のプログラムを採点する負担が非常に大きく，そのために採点システムを開発して利用してきた [1]。

しかし，大人数の学生を対象にプログラミング演習を実施するには，採点という後処理だけでなく，演習時間中に進捗状況を把握し，問題点を学生にフィードバックすることが学習効果を高め，効率的に演習を進めるために重要である。演習時間中に学生のプログラムの評価することを，採点処理とは異なる処理と考え，診断処理と名付けた。

本発表では，演習中に教員をサポートするため必要な機能を検討し，プログラミングの学習モデルに基づいた診断処理を実装し，それを Java プログラム演習に利用した結果の評価とその原因の分析をしたので報告する。

2. プログラミング演習の進め方

本研究の対象とした演習は，Java プログラミング入門の演習 (1 回 2 コマで半期 12 回) である。受講者は C 言語の基本を習得した学生で Java 言語の基本，オブジェクト指向，GUI プログラミングをテーマとして Java アプレットと Java アプリケーションを作成した。受講者数は約 750 名で，午前と午後あわせて 8 クラスを開講し，のべ 14 名の教員と 13 名の TA が指導した。

演習のスタイルは，ウェブ資料を利用した自習形式で，次のように演習を進めた。

(1) 教員は前回の課題の解答と問題点を説明する。
(2) 教員は今回の演習テーマの概要を説明する。
(3) 学生はウェブ資料に書かれた教科書 [2] の該当ページを読み，ウェブ資料に書かれた順番に課題のプログラムを作成する。
(4) 教員は課題の進捗状況を見てヒントを出したり，説明を追加する。
(5) 学生の多くが課題のプログラムを作成したところで，模範解答を学生に示す。
(6) 時間内に課題が終わらなかった学生は，自習時間にプログラムを作成する。

このように自習スタイルの演習スタイルをとっているため，教員は演習中に次のことを適時行うためのコンピュータの支援が必要である。

(1) 学生の進捗状況を把握できる。
(2) 学生が犯している誤りを早めに発見できる。
(3) 学生の誤りを発見するのに必要なデータを容易に得ることができる。
(4) その誤りを学生に適切にフィードバックできる。
(5) いつ学生にフィードバックするかを判断するデータを得ることができる。

3. プログラミングの学習モデルと診断手順

採点の場合は，より正確なプログラムの評価が必要であるが，診断の場合は問題点の早期検出が重要である。そのため，まず，入門者がどのようにプログラムを作成するかを示す学習モデルをまず検討した。学習モデルとして，図 1 のようなコンパイル・実行・改造を繰り返してプログラムを完成させる反復型のモデルを想定した。

図 1: プログラミングの学習モデル

この学習モデルに基づき，プログラムの診断の手順を検討した。プログラムの診断は，一般に動的な分析と静的な分析に分かれる [3]。動的な分析は，実際にプログラムを実行し，入力値と出力値を比較して，プログラムの動作が正しいかを調べる方法である。一方，静的な分析は，プログラムを実行せず，ソースコードを解析することでプログラムの問題点を検出する方法である。

本システムでは，図 1 の学習モデルに従い，診断を次の 7 つのフェーズに分けて実行することにした。(3) と (6) は静的な分析で，(4) と (5) が動的な分析である。(5) は個別のメソッドをテストしたり，アプレットを自動テストしたい場合に用いる。

(1) Java ソースファイルの有無のテスト
(2) Java ファイルのコンパイル
(3) 課題の基本項目に関するソースコードテスト
(4) 実行テスト
(5) メソッドなど実行テスト
(6) アルゴリズムの良し悪しなどのソースコードテスト
(7) class ファイルや HTML ファイルの有無のテスト

静的な分析をまとめて実行するのではなく，問題の早期発見を可能とするために (3) と (6) に分けた。また，学生がプログラムを完成していなくとも診断を可能とするため，すべてファイルが存在しなくとも診断を実行できるようにした。

診断の結果は，○（アルゴリズムが正しい），△（アルゴリズムに問題がある），▲（実行エラー，課題を満足していない），×（コンパイルエラー），★（ファイルが存在しない）の 5 レベルで示す。また，上のどのフェーズで誤りを検出したのかも示す。

†大阪工業大学情報科学部情報メディア科

4. 診断・採点システムの概要

本システムは sh と csh 言語を使って実装され，Linux 上で稼動する。診断・採点用のスクリプトも csh 言語で記述される。動的な分析と静的な分析の検査は正規表現を使って行う [1]。

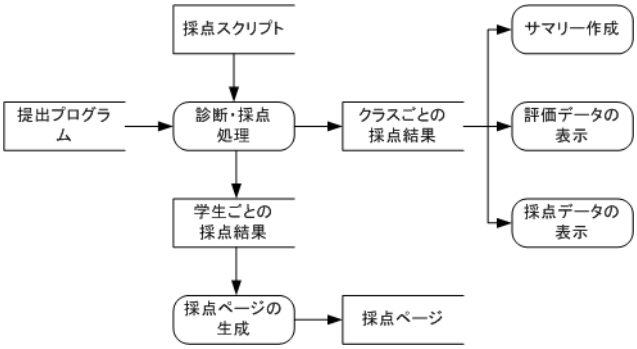

図 2: 本システムの全体処理フロー

図 2 のように，指定したディレクトリに学生が格納したプログラムを採点スクリプトを用いて採点・診断処理を行う。採点の結果として「学生ごとの採点結果」と「クラスごとの採点結果」を生成する。「学生ごとの採点結果」は，学生に提示する採点用 HTML ページを生成するために使用される。「クラスごとの採点結果」は診断時に誤りを分析するために使用される。各データを直接見ることもできるが，次のコマンドを使うことでより簡単に情報を得ることができる。

(1) Display.sh　採点・診断結果から問題点を分析するため,「クラスごとの採点結果」のファイルを個別に表示する。
(2) Show.sh　診断の結果を学生にフィードバックするため,「クラスごとの採点結果」の採点評価データをフィルタリングして表示する。
(3) Summary.sh　学生の進捗状況を調べるため，特定クラスの採点結果のサマリーを出力する。

特に，Summary.sh コマンドでは，○や▲などの評価ごとサマリーだけでなく，進捗状況の把握が容易になるように，どのフェーズでエラーが発生しているかを表示する機能を提供した。

```
workc3     ma   クラス   全   75   名
  INI   14   名     (   18   %)
  CMP    5   名     (    6   %)
  PAT    2   名     (    2   %)
  SHU    1   名     (    1   %)
  END   53   名     (   70   %)
```

5. 評価と課題

半期の演習が終わった後で，担当の教員 (全 8 名) を対象としてアンケートによるシステム評価をおこなったところ，表 1 の結果を得た。学習プロセスを従って診断フェーズを分け，フェーズ毎にエラー発生状況を表示する方法により，進捗状況を把握するという点に関しては良い評価を得られた。しかし，その他の点に関しては問題があり，特に (5) と (6) に関しては評価が低かった。

単純に答えを比較すればよい選択式や穴埋め式の問題とは異なり，プログラムの診断では，多様な解答が可能である。そのため，事前に可能な解答を予測し，誤りを検出する採点スクリプトを作成するのは困難である。特に，必要な処理がないことを検出するのは容易だが，不要な処理を検出するのは難しい。このような理由によって (6) の評価が低くなったと考えられる。

また，本システムでは，コンパイルが正常終了して初めて問題を検出することができる。しかし，学習テーマの理解に不十分なため，学生はコンパイル時に問題をかかえることが多い。そのため，コンパイル時の問題を早期に発見することは本システムでは困難である。また，学生間の進捗の差が激しく，本システムでは中間レベルの学生の進捗状況を監視することになる。そのため，先の課題を取り組んでいる学生のトラブルを見つけるのは遅くなる。

(4) と (5) に関しては，教育上の問題と考えられる。どのタイミングでヒントを出し，正解を学生に示すかは各教員の判断に委ねられている。あまり早く正解を示すと学生のやる気がなくなるし，いつまでも一つの課題に留まっていると演習が進まないという問題がある。単純に何％の学生が終了したという判断では決めることができない。

表 1: システムのアンケート評価

		5	4	3
1	演習中に学生の進捗状況を把握できましたか。	6	2	
2	学生が犯している重要な誤りを早めに発見できましたか。	2	6	
3	誤りの分析に必要なデータを容易に得ることができましたか。	1	5	2
4	その問題を学生に適切にフィードバックできましたか。	2	4	2
5	いつ学生にフィードバックするかを判断するデータを得られましたか。	2	2	4
6	診断の内容は適切だっと思いますか。	1	3	4

6. おわりに

大人数の学生を対象にプログラミング演習を実施するには，演習時間中の進捗把握や採点の支援などコンピュータによるサポートが重要である。教員のアンケートで指摘された問題を解決するために，プログラミングの学習モデルを再検討し，より的確に，より早期に問題を検出できる診断方法を検討する予定である。

なお，本システムの診断機能を半期の後半より学生にも提供し，学生が自分のプログラムを確認できるようにした。これによって，学生の理解が深まると考えられたが，また多くの問題も発生した。この点についても詳細に分析を行い，プログラミング演習の総合支援システムとして完成させてゆきたい。

参考文献

[1] 内藤広志: ”ヒューリステックを用いたプログラミング演習用の効率的な自動採点システム”, 情報処理学会第 66 回全国大会, 2003.

[2] 斉藤 隆, 内藤広志: ”演習でマスターする Java プログラミング”, 共立出版, 2004.

[3] Truong, N., Roe, P., and Bancroft, P., ”Static Analysis of Students’ Java Programs”, the sixth Australian Computing Education Conference, 2004.