研究ノート

プログラミングレポート採点支援ツールと課題設計による評価方法の改善

Improvement in Marking of Programming Exercises using a Marking Support Tool and Subject Design

松浦佐江子
芝浦工業大学システム工学部

Abstract: Since 2003, we have strived to determine the most appropriate method of improving students 'programming skills. One reason for this research is because evaluating programs requires sufficient time and careful consideration of correctness, fairness, and appropriateness, and another reason is because programming should always be evaluated not only by compiling and testing but also by the correctness of the algorithm and the validity of the model. Even if students submit their programs electronically, this does not contribute to improving the correctness of the evaluation in terms of marking their programs because of various factors. Moreover, a lack of concrete assessment items and criteria for the programs means that the marked results will depend on the instructor 's assessment. To reduce the extent to which assessment results depend on the instructor, we propose the introduction of subject design in our programming exercise lessons. We have developed a marking support tool so that we can evaluate programs on one screen where we can readily compile and test the program, browse the code, record the student 's marks, and analyze assessment results. This research sought to improve assessment of programming exercises using the marking support tool we developed and support design.
Keywords: marking support tool, programming exercise, subject design.

1．はじめに

われわれは学生のプログラミング技術向上のため，プログラミング教育方法ならびに評価方法の改善を行っている[1],[2],[3],[4]．本稿では，プログラミングレポート評価方法の改善について報告する．レポート評価には正確・公平・妥当の観点から多くの時間と注意を要する．さらに，プログラミング教育においてはプログラムをコンパイル・テスト実行し，アルゴリズムやモデルを評価するといった多様な観点からの評価が必要であり，評価の手間や評価のための操作の煩雑さにより，電子的なレポート提出のみでは，正確・公平・妥当な評価を行うことは難しい．また，評価基準が明確でない場合には評価が人に依存するため，複数人で評価を分担すると公平な評価が困難になる．そこで，評価項目および評価方法を予め決定するために課題設計を実施する必要がある．われわれはコンパイル・テスト実行・ソースコードの閲覧・評価の記録・評価の分析を一つの画面で学生毎の整合性を保証しながら実施できる採点支援ツールを開発した．課題設計と本ツールの利用により，評価における均一性の確保・付け間違いの防止・効率化を図り，成績評価の正確性の向上を目指す．

Saeko Matsuura
Shibaura Institute of Technology
E-mail:matsuura@se.shibaura-it.ac.jp

2．プログラミングレポート採点における問題点

プログラミングレポート採点における問題は以下のように大別される

① 採点の観点は，プログラムのコンパイル可否・テスト実行結果・アルゴリズムの評価・モデルの評価・仕様の満足度・

（受付：2006年7月1日，受理：2006年10月7日）

プログラミング書法の満足度といったように多様であり，単純にレポートを閲覧するだけでは分からないことが多い．

② 一つの課題は複数の問題で構成される．１課題に対するレポート全体は，各問題番号⇒学籍番号⇒解答プログラムといったファイルの階層構造をもつ．コンパイルやテストの実行は個別に行うことができるが，このような階層構造を辿った閲覧・コンパイル・テスト実行・評価の記録の作業を全学生に対して行うことは教員にとって大きな負荷であり、また課題毎にレポート数が多数（100人×２～３設問程度）でもあるために各学生の解答プログラムとその評価の対応を間違えることがある．

③ 評価基準を予め定義しないと，採点途中で評価の基準が変化し，基準の調整のためのやり直しが多くなる．その結果，評価の不公平が生じる．

これらの問題により，限られた時間の中で，個人の手作業のみでは正確かつ公平で妥当な評価を行うことは難しくなっており，一部の評価の自動化や評価の分担が必要である．

３．評価方法の改善

２節で述べた問題点に対して，以下の方針で解決を図り，採点における正確・公平・妥当性を実現する．

① 限られた時間と人力において効率よく評価を行うために，煩雑な確認、調整を軽減して、採点誤りを防止する．

② 課題設計により，課題の学習内容・達成目標・評価内容・判定方法を整理し，評価項目を明確に定義することで，採点者への依存性の低減を図る．

③ 判定方法には自動化可能なものと人間が閲覧により発見できるものがある．これらを合わせて全体を評価する．

④ 評価結果の分析を可能とし，問題点の発見を容易にする．

具体的には，課題設計の実施，①③④を支援する採点支援ツールの開発を行う．次節でこれらについて説明する．

４．課題設計

プログラミング技術の達成目標には，正しい構文で定義できる，アルゴリズムが定義できる，課題条件を満たす，機能要求を満たす，非機能要求（効率，信頼性，使用性，保守性等）を満たすといった段階があり，これらの目標を順次取り入れて課題設計を行う．課題設計では課題毎に表１の項目について具体的な内容を定義する．ここで，課題を実施する演習は講義と関連しているものとする．

表１　課題設計

項　目	内　　容
ポリシー	課題作成の考え方
テーマ	講義科目に対応したキーワード
学習内容	講義科目の構成要素を用いた当該課題で学習する内容
達成目標	学生が定義できるようになることを期待する項目
評価内容	達成目標項目を評価する基準項目とその評価値
判定方法	自動または手動による方法
テーマ間関係	他のテーマとの関係

評価内容には，コンパイルの可否，インデント・命名・定数の扱い，冗長性，コメント等のプログラム書法における適切さといったプログラミング一般の評価項目（規定項目）と学習内容に従った項目がある．

Javaによるオブジェクト指向プログラミングの演習における課題設計の例を表２に示す．本演習はC言語の講義と演習（各２コマ）を終えた２年次後期に行われる演習であり，オブジェクト指向に関する講義と同時進行で実施されるものである．

表２　課題設計の例

第２回課題
ポリシー：サンプルプログラムを理解し，要求を取り入れてプログラムを拡張する． テストケースの一例を示し，テストケースを拡張して実行の確認を行う． テーマ：クラス定義 学習内容： １）クラス・フィールド・メソッド・コンストラクタの定義 ２）フィールドへのアクセス・メソッドのオブジェクトへの適用 ３）インスタンスフィールド（メソッド）とクラスフィールド（メソッド）の使用 達成目標： １）既存のクラス定義にフィールド・メソッド・コンストラクタを要求通りに正しく追加できる． ２）フィールドへのアクセス・メソッドのオブジェクトへの適用が正しくできる． ３）インスタンスフィールド（メソッド）とクラスフィールド（メソッド）の使い分けができる． 評価内容：（規定項目以外） 指定要素（クラス・メソッド・フィールド・コンストラクタ）が正しく定義されているか・指定要素の型の種別は何か・指定要素の修飾子の種類は何か（public|private：なし）・指定メソッドのパラメータの数と種別は何か・フィールドおよびメソッドへのアクセス形式は何か・（直接：get・setメソッドによるアクセス）・フィールドおよびメソッドへのアクセスは正しいか・指定メソッドのエラーケースに関する処理はあるか 判定方法：コード検査（Javaのメタプログラミングを用いた指定要素・型・修飾子・パラメータ情報を取得するプログラムによる自動検査(1)・閲覧 テーマ間関係：なし

５．採点支援ツール

３節で述べた改善を行うために，採点支援ツールを下記のように設計し，Java言語により開発した．

① コンパイル・テスト実行・閲覧・評価の記録を学生毎に一つの画面で扱い（図１），個人と評価の対応を保証し，学生間，問題間の移動が容易に行える環境を提供する．評価の記録は評価項目毎に生成される評価値のチェックボックスをチェックすることで行う．

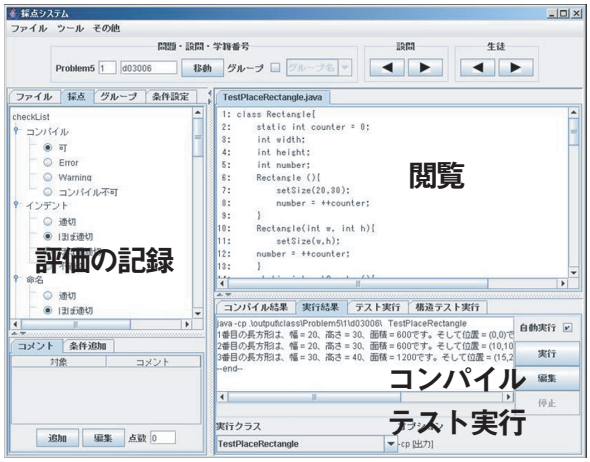

図１　採点支援ツール

② コード検査やテストプログラム等自動化可能な判定方法をツールのプラグイン（拡張部品）として定義できるようにする．

③ プログラミング言語に非依存とする．

④ 評価記録を基に，該当する項目がチェックされている学生をグループ化する機能を提供する．これにより，特定の評価を受けた学生の回答のみを選別して閲覧することができる．

６．演習への適用

(1) 採点方法

Java言語を用いた演習（２回）とC言語を用いた演習（計３回）の採点を複数の教員（３名）およびTA（延べ15名）により，次の

手順で実施した.

① 教員が課題毎に解答例・評価項目と評価値・テストプログラム（一部の課題）・コード検査プログラム（一部の課題）を定義する．評価項目と評価値をCSV形式ファイルで定義し，採点支援ツールで読み込むと評価のチェックボックスが生成される.

② TAおよび教員が評価項目に従い，各レポートの評価値をチェックする．採点支援ツールにレポートデータを読み込むと，各プログラムのコンパイルは自動的に行われ，結果がツールの「評価の記録」欄に記録される.

③ 評価後のデータをCSV形式でファイルに出力し，教員が結果を一覧しながら各項目に配点して成績を決定する.

例えばJava言語を用いた演習では規定項目と表２の「評価内容」の項目を設定し，評価値はyes/noまたは想定される解答候補と「その他」の項目からの択一式の選択とした.

(2) 適用結果

課題設計により，評価項目と評価値の選択肢をあらかじめ決定することで，教員およびTAが役割分担を行って評価を実施することができた．TAは評価項目毎に各プログラムがどの評価値に当てはまるかを判定する．評価値がyes/noである場合は採点者によって結果に相違は生じないが，プログラム書法に関して４段階程度で判定するような場合の定量的な指標は予め示す必要がある．選択肢に当てはまらない場合を想定して「その他」の項目を設けたが，ほとんど使用されなかった．「その他」に分類されたプログラムについてはグループ化により絞り込んで教員が分析を行い，「その他」となっている理由を判定することができた．実際には想定したものと完全には一致しないが意味的には正しい定義が「その他」に分類されていた.

採点支援ツールを使用した感想としては，「GUIは直感的に分かりやすく，使いやすい」「評価が一つの画面で行われるため評価効率がよい」というように評価の正確化には寄与していると思われる．しかし，「評価項目が多い場合は採点に時間がかかる」「ソースコードの閲覧がしやすいようにキーワードの色づけを行ってほしい」「インタラクティブなプログラムの動作確認の際，すべての学生に対し同じ入力値を繰り返し入力することが採点者とって大きな負担になる」といった意見もあった．今回は自動的に判定可能な評価項目も手動で判定したことや，評価項目を複数選択式ではなく択一式にしたために「評価項目が多い場合には採点に時間がかかる」という意見が出たと考えられる.

(3) 考察

当初の目標である成績評価における正確さ・公平さ・妥当性については以下のように考える.

① 正確さ

閲覧・コンパイル・テスト実行・評価の記録が学生毎に管理されるため，付け間違いが生じない.

② 公平さ

予め決定した評価項目に従って，プログラムを絶対評価することができる．評価項目の選択肢に当てはまらない場合は新たに選択肢を追加し，グループ化機能により，評価の見直しを行うことができるため，曖昧な記憶で評価することがなくなる．また，課題設計を十分に行うことで，評価の見直しを減少させることができる.

③ 妥当性

プログラミング技術の習熟には４節で述べ

たように達成目標に応じた段階があり，それぞれ異なる判定方法がある．参考文献[5][6]の研究ではこのような判定方法の自動化による採点方式を提案しているが，プログラムの記述のバリエーションが多い場合には，漏れなく判定するプログラムを定義することが難しい．評価項目として定義した内容を判定するプログラムを用意することで自動的な判定は行えるが，この場合には予め予期した項目しか判定することができない．そこで，自動的な判定と教員による閲覧とを組み合わせて判断を行う必要がある．また，予期しない項目があった場合には，該当するレポートだけを分類して調べることは有効である．

７．まとめ

プログラミング技術の習得には，多くの課題を評価しながら繰り返し解くことが重要である．その上に妥当な評価を正確かつ公平に行わなければならない．本稿では，教員が本質的に重要な課題設計に時間を割くことができるように，採点時における評価の正確さを向上する方法を提案した．2004年度および2005年度には学習状況の把握のため，提出課題以外の提出も可能とした．2005年度からは本ツールの導入により，教員が課題作成に時間をかけることが可能となり，学生が自己採点できるように解答例をレポート提出締め切り後に速やかに公開することができた．その結果，2004年度の非提出課題の提出率（非提出課題の平均提出数／提出課題の平均提出数）35%に対し，2005年度には非提出課題の提出率は48%に向上した．また2004年度の単位取得率は36%，2005年度は53%であった．２年間の比較であるので不十分ではあるが，解答例の公開により学生は自己採点して学習の機会を増やしていると考えられる．

判定方法を自動化するためには評価項目を個々の学生のプログラムに依存しないようにプログラムの仕様を詳細に規定する必要がある．今回適用した演習では，仕様を詳細に規定してプログラムを作成することが目的に合致しているため問題はないが，プログラムの構成の妥当性を評価する場合には，予めコード検査やテストプログラムをすべての学生に共通に作成することはできない．このような場合には，プログラムの客観的な指標であるソフトウェアメトリクスの測定といったソフトウェア工学手法を用いる必要がある．今後はコード検査・テストプログラムの判定方法をプラグインとして定義するとともに，これらも判定方法として定義することを検討し，これらの判定方法を学生自身が自己のプログラムを評価する手段として提供する予定である．

注

（1）ただし，今回の採点においてはこれらのプログラムを５節で述べる採点支援ツールのプラグインとして実装せず，採点者が手動で判定した．

参考文献

[1] 熱田智士, 松浦佐江子: Javaプログラミング演習向け課題レポート提出・管理機能を付加した授業支援システム. 情報処理学会, FIT2004情報科学技術レターズ, Vol.**3**, pp.359-362, 2004.

[2] S.Matsuura and S.Atsuta: Lesson Support System for Programming Exercise Lesson Improvement. WBE2005, pp.131-136, 2005.

[3] 松浦佐江子:プログラミング演習における評価方法の改善. 論文誌IT活用教育方法研究, 私立大学情報教育協会, 第**8**巻, 第1号, pp.51-55, 2005.

[4] S.Matsuura, S.Atsuta and T.Fujiwara: Programming Exercise Evaluation Method using Code Review for Improvement of Programming Skill. CATE 2005, pp.89-94, 2005.

[5] 内藤広志:プログラミング演習の総合支援システムの概要, 情報処理学会FIT2004, N-011, 2004.

[6] 小西達裕, 鈴木浩之, 伊東幸宏: プログラミング教育における教師支援のためのプログラム評価機構. 電子情報通信学会論文誌D-I, Vol. **J83-D-I**, No.6, pp. 682-692, 2000.