プログラミング演習におけるテスト支援方法の提案

2010SE055 伊倉慶太　2010SE088 加藤綾真　2010SE092 加藤優磨
指導教員：蜂巣吉成

1　はじめに

現在，多くの情報系の大学でプログラミング言語の学習が行われている．学習方法としては，テキストを用いる講義形式のものと，PC を用いる実習形式のものによる学習形態が多く取られている．実習形式の学習では学習者は演習課題から仕様を読み取り，コーディングを行い，テストによってプログラムが仕様を満たしていることを確認し，提出して課題を終了している．プログラミング学習において，テストによって入力に対する出力を確認することはプログラムが仕様を満たしているか確認するために必要不可欠であり，重要である．

現在のプログラミング演習では，課題の成果としてソースプログラムや実行結果を提出しているが，学習者が実行したテストを必ずしも教員が把握できるとは言えず，教員から学習者へのテストに関するフィードバックを行うことが困難である．

学習者は同値分割や境界値分析などのテスト設計技法を知っていたとしても，実際にテスト設計技法に基づいたテスト設計ができているかはわからない．学習者が自身のテストが十分に仕様を満たしているか判別がつかないことや，教員から学習者へのテストに関するフィードバックが困難であること，学習者がテストの重要性を理解していないといった問題点が挙げられる．

本研究では学習者が課題の仕様を満たすテストを設計をできるようになることを目的とする．そのためには，学習者が行ったテストを評価し，どのようなテストが不足しているのかを学習者に通知する必要があると考える．本研究では main 関数を含み，関数が数個程度の初学習者向けの小規模なプログラムを対象とするので，プログラム全体の入出力の確認を行うテストを想定する．これらの目的を達成するために，Web ベースの統合開発環境 (Web Integrated Development Environ-ment:WebIDE) を構築し，学習者が行ったテストが十分であるか評価し，不十分だった場合は十分なテストを設計できるよう支援する環境を提案する．WebIDE を用いることによる利点は次の通りである．

- ローカルの PC の端末で実行した場合は，実行時の入出力を教員が把握するのは難しいが，WebIDE では Web サーバ上で実行も行うので，サーバで入出力の結果が取得できる．
- 後述する新しい演習プロセスにおいて，テストケースを提出し評価する必要があるが，WebIDE を用いる事によって，テストドライバを自動生成し複数のテストケースを一括で実行するということも可能となり，学習者がテストを行う上での負担を減らすことができる．

また，学習者へのテスト設計支援として，テスト駆動開発の考え方を取り入れた学習プロセスを提案する．

2　関連研究

「テスト駆動開発に基づくプログラミング学習支援システム」[1] では，評価をする際の指導者の負担を減らす点に着目し，Triple Checked Testing という学習者が自ら評価できる評価手法を提案している．Triple Checked Testing は，指導者のコードと学習者のテストケースを組み合わせてテストケースの誤りを検出するテストケースチェック，その後学習者のコードとテストケースを組み合わせて実装の誤りとテスト漏れを検出するセルフチェック，最後に学習者のコードと指導者のテストケースを組み合わせて実装漏れを検出する実装チェックの 3 つで構成されている．指導者がコードとテストケースを用意すればどのような問題にも対応できるという利点があるが，誤りや漏れが検出された際，具体的にどこが誤りなのか，どこが漏れなのかが分かりにくい．本研究では教員がテストケース毎にコメントを設定することにより，具体的な誤りや漏れを学習者に通知できるようにする．

「Java プログラミング初学者に対するテスト方法学習支援ツール」[2] では，単体テストプロセスの学習支援としてテストケースを公開し，テストプログラムをどのように完成させるかを理解させる方法を提案している．本研究ではテストケースは公開せず，どのようなテストケースが必要なのかを考えさせることでテスト設計の支援とする．

テストの重要性を理解させる方法として，「テスト駆動開発手法による Java プログラミング教育支援システムの提案」[3] ではテスト駆動型開発手法に着目し，テストコードを公開することで，学習者はそこからも課題の仕様を読み取る事が可能であり，テストの重要性を理解することができるようになっている．この時のテストコードは，半自動生成され教員の負担も少なく済むが，本研究では学習者が十分なテストを設計できるようになることを主な目的としているので，テストの設計は学習者が行うものとする．

3　学習者へのテスト設計支援

3.1　演習のプロセス

現在大学で行われている演習プロセスは，学習者は課題を確認し，プログラムを作成し，コーディングとテストを繰り返し行い，学習者の裁量で提出し，課題の終了としていることが多い．この演習プロセスではテストは学習者の作成したプログラムが学習者の想定するテストケースで正しく動作するかどうかを確認するのみとなってしまっていて，自身の行ったテストが課題の仕様を満たしているか確認する方法がない点が問題である．

本研究ではテスト駆動型開発の考え方を取り入れた新しい演習プロセスを提案する．提案する演習プロセスは，学習者は課題を確認した後，仕様を満たしているか確認できるテストケースを作成する．作成したテストケースは 3.2 節で述べるテストケース評価機能によって評価され，十分なテストケースかどうか判断される．テストケースが不足している場合，教員が設定したコメントによって学習者にフィードバックを行い，学習者はコメントから自身のテストケースを修正する．テストケースが完成した後，コーディングを行い，先程作成したテストケースによるテストを行い，プログラムが仕様を満たすものであるか確認し，課題を提出して終了とする．このプロセスで演習を行うことで学習者はコメントによる具体的なフィードバックを受けることができ，学習者が自身のテストケースが十分に仕様を満たしているかわからないという問題を解決することができると考える．

3.2 テストケース評価機能

学習者へのテスト設計支援としてテストケースの評価機能を提案する．テストケースを評価する方法は学習者が行ったすべてのテストケースを取得し，教員がそれを直接確認する方法や，教員の作成したテストケースを取得し，教員の作成したテストケースと学習者の作成したテストケースを比較する方法，ホワイトボックステストのカバレッジを取得する方法など，様々な方法がある．

本研究では学習者に十分なテストケースを作成させる方法として学習者が設計したテストを教員が課題毎に用意する評価基準をどの程度パスするかによってテストケースを評価する方法を提案する．評価基準を書くことができるならどのような課題にも対応できるという利点があるが，評価基準の判定基準は課題によって入力の数やデータ型が一定ではないので，一様に記述できず教員の負担は増える．

また，評価基準に対応したコメントを表示することで学生へのフィードバックを行うことが出来ると考える．コメントは不足するテストケースを設計させることを促せるようなものが望ましい．学習者はコメントを読み，自分の設計したテストの不足している箇所を把握し，修正することで学習者へのテスト設計支援とする．

3.3 テスト一括実行機能

本研究では複数のテストを一括で実行することのできるテスト一括実行機能を実装する．この機能によって，学習者が実行の度にテストケースを何度も入力する手間を省き，テストを面倒に感じることを軽減することができると考える．

4 テストケースの評価基準の作成

4.1 対象とする課題の分類

テストケースを評価するための評価基準を作成する上で，テストケースの入力形式が課題毎に異なり評価基準が同一ではないことが問題となる．テストケースは課題のコーディングを行う前に設計するので，テストケースの入力に着目し，分類方法は入力の数と入力のデータ型の 2 つによる分類を組み合わせたものを考察した．

表 1 課題の分類結果

入力数	数値数	文字型	文字列型	複合型
一定	54	2	17	5
一定でない	14	0	2	4

本研究では教育機関で用いられる事を想定しているので南山大学で行われているプログラミング演習課題を例とし，2010 年度プログラミング基礎実習，及び 2011 年度プログラミング応用実習を対象に課題のテストケースの書き方による分類を行った．

これらの分類が同じ課題は，細かい点では異なる評価基準になるが，評価基準が似ている課題となる．

4.1.1 入力の数による分類

入力の数による分類を行うと，次のようになる．

入力がない　入力がなく，出力のみのもの．該当する課題が少ないので，本研究では扱わない．

入力の数が一定　入力の数が一定のもの．

入力の数が一定でない　2 種類の入力方法が確認できた．最初の入力で残りの入力の数を決定するものと，負の数や，Ctrl+D などある条件を満たす入力がされるまで入力を続けるものがある．

4.1.2 データ型による分類

数値型　整数型，実数型の数を入力するもの．

文字型　文字を入力するもの．

文字列型　文字列を入力するもの．

複合型　整数型の入力をした後に char 型の入力をするなど，異なるデータ型の入力を行うもの．

対象となる全 98 題を分類すると表 1 のようになった．入力の数が一定である課題が大半を占めていた．また，多くの課題が数値型の入力をさせるものであり，文字型の入力を扱う課題は少なかった．

それぞれの分類について該当する演習課題を例に学習者が作成すべきテストケースと，教員が作成する評価基準の具体例を提案する．

4.2 評価基準の記述方法

本研究の利点として，評価基準を設定すれば様々な課題に対応できるという点があるが，これは教員の負担を増やしていると言える．教員の負担をできるだけ減らし，かつ分かりやすい記述法を提案する．評価基準の書き方は次の通りである．

- 評価基準番号 [tab] 判定基準 [tab] コメント

このように，tab で区切って記述する方法を提案する．これをテキストファイルで保存し，PHP に変換したものを

使用する．詳しくは 4.3 章で述べる．

4.2.1 判定基準

学習者のテストケースが仕様を満たすかものであるかを判定するために判定基準を記述する．入力された値について，変数を教員が num，math などの変数を用意しテストケースの入力をそれぞれ当てはめて判定基準を記述する方法が考えられる．教員は自身の決めた変数によって判定基準を書くので変数の意味を理解しているので記述しやすい．しかし，課題毎にどのような入力がされるか考えながら変数を設定することは教員の負担が増加する．提案方法では単純に記述するために，入力数が一定である場合は入力の順番に$1，$2 と入力の前に変数を設定し，入力数が一定でない場合は，入力の最初を$1，最後を$n とし変数を設定する．

本研究では対象とする言語が C 言語であり，C 言語の論理式は教員も馴染みが深く記述しやすいと考える．入力のデータ型が整数型や実数型のものは C 言語の論理式と同じように書くことのできる PHP の論理式で記述する．また，文字型のものは C 言語の標準ライブラリ関数で記述する方法が考えられるが，記述内容が複雑になる場合があるので，正規表現のパターンとして記述する．

課題によっては入力以外にも課題特有の変数を用意する必要がある．入力数が一定の課題では sum = $1+$2+$3 のように簡単に記述することができるが，入力数が一定でない課題では特有の変数を宣言することは可能であるが，記述が煩雑になりやすい．最大値を求める関数 max() など，あらかじめ関数を用意することで評価基準が書きやすくなり，教員の負担が少なくなると考えられる．

4.2.2 コメント

学習者が設計したテストケースと教員が記述した判定基準からテストケースが不足している場合，判定基準に対応したコメントを表示することで学習者はコメントを読みテストケースを修正，追加する．これを学習者へのテスト設計支援とする．コメントはどの判定基準のテストケースが不足しているか学習者が考えることを促すようなものが望ましい．

一つの評価基準に対応したコメントを一つ記述する方法が考えられるが，一つの仕様に対応する評価基準が二つの判定基準を含む場合，どちらの判定基準をパスしていなくてコメントが表示されるかわからない．そこで二つの判定基準を別々に記述し，判定基準には番号をつけて一つの評価基準を分割して記述する．この方法によって判定基準に沿ったコメントを表示することができる．

4.2.3 評価基準のカバレッジ

評価基準のカバレッジを次の通りに設定することで，学習者に仕様を満たすために必要なテストケースがどの程度設計できているか数値で示すことができる．

- 評価基準のカバレッジ = 学習者の設計したテストがパスした評価基準種類数 / 教員が設定した評価基準種類数

表 2 仕様を満たすテストケース例

テストケース	英語	国語	数学	理科	社会	成績
1	73	82	78	76	71	評価 A
2	100	90	80	77	88	評価 A
3	73	81	78	76	71	評価 B
4	70	74	72	80	70	評価 B
5	70	100	60	70	90	評価 C
6	90	70	69	89	100	評価 C
7	60	60	60	60	59	不合格

4.3 評価基準の具体例

次の課題について仕様を満たすテストケース例，判定基準，評価基準を記述する．

南山大学 プログラミング基礎実習 課題 4-3

各教科の点数を入力して成績を出力する英語，国語，数学，理科，社会の試験の得点がそれぞれ入力されると， 以下の基準で成績を判定して出力するプログラムを作成しなさい．キーボードからは試験の得点として 0 から 100 までの整数値が必ず入力されると仮定して、エラー処理は特に考えなくてよい。

- 評価 A 各教科が 70 点以上、かつ合計が 380 点以上
- 評価 B 各教科が 70 点以上、かつ合計が 380 点未満
- 評価 C 各教科が 60 点以上
- 不合格上記の条件以外のもの

各教科の点数の組み合わせで評価基準が異なる課題である．入力の境界値は 70，69，60，59，合計の境界値は 380，379，であり，これらの境界値を重視したテストケース設計を行う必要がある．今回は 0 から 100 までの整数値が入力され，エラー処理について考えなくてよいとあるので 100，0，負の値については考えないものとする．

入力される点数はそれぞれ英語$1，国語$2，数学$3，理科$4，社会$5 となる．
点数の入力だけでなく，それらの合計点数も評価の判定基準となるので新しく変数を用意する必要があるので，合計点数を sumvar，最小値を minvar を用意し，次のように行う．

- sumvar = sum($1,$5)
- minvar = min($1,$5)

仕様を満たすために必要な判定基準の一例は次のようになる．

- すべての教科が 70 点以上かつ，合計点数が 380 点
- すべての教科が 70 点以上かつ，いずれかの教科が 70

点かつ，合計点数が 380 点以上

これを評価基準番号，判定基準，コメントで記述すると次のようになる．

1[tab]sumvar == 380 && minvar >= 70[tab] 評価 A の場合の合計点について評価 B との閾値のテストが行われていません

1[tab]sumvar >= 380 && minvar == 70[tab] 評価 A の場合の教科の点数について評価 B との閾値のテストが行われていません

評価基準番号が 1 から 4 まであるので，それぞれの評価基準に 25% のカバレッジが設定される．例えば，sumvar == 380 \&\& minvar >=70 のみをパスしなかったテストケースに対しては，カバレッジ:75% 評価 A の場合の合計点について評価 B との閾値のテストが行われていませんと学習者に表示し，テストケースの修正を行う．

5 実装と検証

5.1 実装

学習者のテスト設計を支援をするために，WebIDE を設計，実現した．本研究ではテストの実行時にテストケースを取得し，分析する必要があるので，コンパイル機能，テストケース一括実行機能，アカウント管理機能，ファイル管理機能，テストケース評価機能を実現した．実装言語は JavaScript と PHP を用いた．

5.2 学習者の WebIDE を用いた演習プロセス

WebIDE は教員が用意した評価基準が記入されたテキストファイルから，評価基準のカバレッジのチェックと，評価基準に対応したコメントを通知する PHP プログラムを生成する．学習者は課題に対するテストケース表を記述し，実行ボタンを押すことで，WebIDE によりテストケースが評価され，テストケースの評価基準に対するカバレッジとコメントが表示される．学習者はこの表示されたカバレッジとコメントを用いて，テストケース表を修正する．学習者はテストケース表の修正が完了した後に，ソースコードを記述し，テストケース表の一括実行機能を用いて，ソースコードのテストを行う．この際，Web 上でのコンパイルは，ソースコードをサーバ上で gcc を呼び出してコンパイルし，各ユーザ用のディレクトリに実行ファイルを出力する．実行時の標準入力値やコマンドライン引数については，プログラムの実行中に入力待ちを行うことができないので，予めテストケース表に値を入力しておく必要がある．また，テストケース一括実行機能については，PHP を用いてテストドライバを用意し，サーバ上の実行ファイルを順番に呼び出すことで実現している．また，プログラムの実行時に gcov を用いることで，C0・C1 カバレッジを測定し，学習者に通知する．

5.3 検証

本研究がテストケース設計支援を達成できているかどうか確認するために，WebIDE のテストケース分析機能を既にプログラミングを学習した大学 3，4 年生 5 人に使用してもらった．

4.3 節で記述した課題のテストケースを実際に作成してもらい，実行ボタンを押すごとに記録をとり，学習者に表示するコメントの詳細度ごとにテストケースのカバレッジを集計した．今回は，コメントの表示をコメントを表示なし，表示あり，詳細なコメントを表示するの 3 パターンに分けて検証を行った．コメント表示なしの評価を確認した場合のテストケースのカバレッジは平均 35%，コメント表示ありの評価を確認した場合のテストケースのカバレッジは平均 70%，詳細なコメントの評価を確認した場合のテストケースのカバレッジは平均 100% となった．コメントを表示することによってテストケースのカバレッジの向上が見られ，詳細なコメントを表示することで更にカバレッジの向上が確認でき，最終的に 5 人とも仕様を満たすテストケースを設計することができた．不足するテストに対応したコメントを表示することで，学習者はテストケースの修正を行い仕様を満たすテストケースを設計することができた．これにより本研究がテスト設計支援を達成していることを確認した．

5.4 考察

学習者の理解度に合わせたコメントを表示することで，テストケース設計の支援とし，コメントは詳細度を分けていくつか用意することが望ましいと考える．各学習者の理解度に合わせたコメントを表示することで更に細かく学習者のテスト設計支援が可能であり，用意したコメントを学習者の進捗に応じて自動に表示する機能も検討する．

6 おわりに

本研究では，学習者は自身のテストケースが課題の仕様を十分に満たすものであるかどうかを確認し，不足する場合は学習者へのフィードバックを行うテストケース評価機能を提案した．実際に学習者に利用してもらい，評価を行ったところ，支援方法が有効であることが確認できた．教員がテスト設計技法を基に課題に応じた評価基準を考えなければならないことなど，教員の負担が大きいことが問題である．今後の課題としては，評価基準の自動生成など教員の負担を減らす方法を考察する必要がある．

参考文献

[1] 吉田 英輔，角川 裕次：”テスト駆動開発に基づくプログラミング学習支援システム 初心者開発者のためのセルフトレーニングアーキテクチャ”，電子情報通信学会技術研究報告．ソフトウェアサイエンス，Vol. 105，No. 331，pp. 27-32，2005.

[2] 上河内 頌之，松浦 佐江子：”Java プログラミング初学者に対するテスト方法学習支援ツール”，電子情報通信学会技術研究報告．教育工学，Vol. 106，No. 364，pp. 37-42，2006.

[3] 松島 由紀子，笹井 康裕，舩曳 信生，中西 透，天野 憲樹：”テスト駆動型開発手法による Java プログラミング教育支援システムの提案”，電子情報通信学会技術研究報告．ET，教育工学，Vol. 109，No. 82，pp. 7-12，2009.