R-6

C 言語学習者のためのオンラインテストドライバの研究

橋浦研究室　1155461 渡邉　弘樹

1. はじめに

近年，プログラミング教育の必修化など，社会的にプログラミングの重要性が高まっている．その一方で，プログラミングの学習者が自分自身で不具合を見つけることは困難な状況にある．

2. 研究目的

本研究では，ソフトウェアテスト技術を用いたオンラインのテストドライバを開発し，学習者自らが不具合を発見できる環境を提供する．

2. 1. ソフトウェアテスト

ソフトウェアテストとは，プログラムが仕様通りの機能を備えているかを確認するソフトウェア開発の工程の 1 つである．エンジニアが自分でソフトウェアテストを行うためには，被テストプログラムを実行するための入力値（テストデータ）を用意したり，実行を自動化するためのテストコードを作成するなど専門の知識が必要になる．よって，プログラミングの学習者が自分でソフトウェアテストを行うのは容易ではない．

2. 1. 1. 同値類

テストデータを準備するにあたって，重要となるのが同値類である．同値類とは，仕様から判断して，プログラム中で同一の手順により処理されると思われるデータの集合[1]である．例えば，年を入力し，閏年かどうかを判定するプログラムを作成することを考える．このとき，入力値は以下の 4 つの手順（つまり，同値類）に分けられる．

① 400 年で割り切れる値
② ①以外で，100 で割り切れる値
③ ①②以外で，4 で割り切れる値
④ ①②③全てに当てはまらない値

ここで，プログラムの動作確認を行うための入力値として 2000 と 2400 の 2 つの値を用いることを考える．これらは共に①に当てはまる値であるため，これらの値は同値類である．

2. 2. バグが潜んでいるプログラム

十分なソフトウェアテストを行うためには，プログラム中に存在するすべての同値類から 1 つ以上の値を選んでテストを実施することが重要である．

本研究で取り扱う“バグが潜んでいるプログラム”とは，テストデータが前節で述べたプログラム中に存在するすべての同値類を網羅していない状態のことである．当然のことながら，テストデータに存在していない値をプログラムに与えた場合，期待された結果が得られるかどうかはわからない．例えば，前節で述べた例で考えてみると，入力については①の同値類でしか動作確認を行っていないため，これらが正しい結果を示しているからといって，②～④の同値類を与えた場合に正しく判定できるとは限らない．

3. 提案手法

これまでに述べた問題の発生を防ぎ，学習者自らが不具合を発見できる環境を実現するために，学習者がプログラムをブラウザからサーバにアップロードし，その採点結果を受け取ることができれば良いと考えた．加えて採点に利用するテストコードやテストデータ作成には専門知識が必要になるため，問題に合わせてあらかじめ教授者が用意しておき，本研究で開発するツール側で学習者のプログラムと合成を行うこととする．

ところで，サーバ上で学習者のプログラムの評価を行ってしまうと，意図せず無限ループするプログラムなどがあった場合，サーバに過負荷がかかる恐れがある．これを防ぐため，本研究ではプログラムの実行に学習者のブラウザを用いることとした．

これまでの提案をまとめると，本研究のツールに必要となる機能は以下の 4 つとなる．

① プログラムをブラウザから提出できる機能

Copyright © 2019 Nippon Institute of Technology

② プログラムにテストコードやテストデータを自動的に埋め込む機能
③ 自動的にテストを実行する機能
④ ブラウザ上でテストを実行する機能

4. ツールの実現方法

今回，学習者が取得対象としている言語は C 言語である．ウェブブラウザには通常 JavaScript の実行環境が備わっているため，学習者の C のソースコードを emscripten[2]を用いて JavaScript に変換した上で実行することとした．加えて，テストを自動的に実施するためにはテストドライバが必要となるため，Minunit[3]を採用した．

また，ソースコードの提出画面等のウェブアプリケーション部分には PHP を用いて開発を行っている．

5. 実験

ツールの有効性を明らかにするために，日本工業大学情報工学科の学生 12 人を被験者として実験を行った．

本実験における RQ は以下の 2 つである．

RQ1. 学習者はバグが潜んでいるプログラムをどの程度提出するのか？
RQ2. ツールがフィードバックしたエラーを学習者は修正できるのか？

5. 1. 実験方法

実験には「階乗を計算するプログラム」を対象として行った．

また，実験の流れを以下に示す．

① プログラムを作成する
② （ツールを用いずに）動作確認を行い，入力に使用した値を記録する
③ ツールを用いて動作確認を行う
④ ツールからプログラムの誤りをフィードバックされた場合，それがなくなるまで修正を行う．修正の際に動作確認を行った場合，その入力に使用した値を記録する

5. 2. 適合率 P

教授者が用意した同値類（正解例）のうち，学習者が回答したテストデータに含まれている同値類の数を A，学生が動作確認を行ったデータの総数を B とすると，適合率 P は式(1)となる．

$$P=\frac{A}{B} \tag{1}$$

ただし，学生が同じ同値類に属する値を複数利用することも考えられるが，2 つ目以降は不正解と判定することとした．例えば今回の Q の場合，教授者が同値類の数を以下の 4 つに定めたとする．（ただし，値は 32bit，符号有 int 型）．

① 1～12 の値
② 13（オーバーフローする）
③ 0（0 を階乗すると 1）
④ -1（0 未満の値は計算できない）

ここで被験者が動作確認に使用した入力値が{2，5，10，13，0}の 5 つである場合を考える．前述の A 求めるためには，被験者の入力値を教授者が同値類にしたがって分類してみれば良いので，①={2,5,10}，②={13}，③={0}となり 3 と求まる．定義より B は 5 であることから，$P=\frac{3}{5}$と求まる．

5. 3. 再現率 R

教授者が用意した同値類（正解例）の数を C とすると，再現率 R は式(2)となる．

$$R=\frac{A}{C} \tag{2}$$

前節と同様の事例で考えてみると，A は 3，C は 4 であることから $R=\frac{3}{4}$と求まる．

6. 実験結果と考察

被験者の，再現率の平均$\overline{R}$を表 1 に示す．また，バグの残存数はバグが潜んでいたプログラムの数を表している．

ここで RQ1 について考えてみると．表 1 ツール使用前のバグの残存数からツールで判定を行う前のプログラムでは，12 個中 11 個のプログラムにバグが潜んでいる．このことから学習者のテストデータの選定には欠落があり，バグが潜んでいる可能性が高いことがわかる．次に RQ2 について考えてみると．表 1 のツール使用後のバグの残存数が 0 であ

Copyright © 2019 Nippon Institute of Technology

り，適合率の平均$\overline{P}$と再現率の平均$\overline{R}$が共に上昇している．このことから，ツールでフィードバックすることにより，学習者は欠落していたテストデータを用いてバグを修正できるようになったことが分かる．

表１　問題の適合率の平均$\overline{P}$と再現率の平均$\overline{R}$及びバグの残存数

	$\overline{P}$	$\overline{R}$	バグの残存数
ツール使用前	0.48	0.52	11
ツール使用後	0.53	0.77	0

使用前と使用後の再現率の平均$\overline{R}$に対しｔ検定を行った．検定表を表 2 に示す．有意水準αは 0.05 とし，帰無仮説H_0は「ツール使用前と使用後の再現率の平均$\overline{R}$は等しい」とした．表 2 より $p<0.05$ を満たすため，H_0は棄却され，使用後と使用前の再現率の平均$\overline{R}$が向上したことが確認できた．

表２　再現率の平均$\overline{R}$においてのｔ検定表

	ツール使用前の$\overline{R}$	ツール使用後の$\overline{R}$
平均	0.7708333	0.5208333
分散	0.0620265	0.0961174
観測数	12	12
自由度	22	
ｔ値	2.1777316	
p 値	0.0404292	

7. まとめと今後の課題

本研究の目的は，利用者にテスト技術がなくても，バグを見つけ出し，自らバグが潜んでいる可能性が低いプログラムを作成できることであった．実験により，ツールでバグを見つけることで，利用者はバグを修正しバグが潜んでいる可能性が低いプログラムを作成することができるようになることが明らかになった．

今後の課題としては，コード解析ツールを組み合わせることにより，バグが潜んでいた場合どこに潜んでいるのかなど，より詳細なフィードバックを利用者に与えられることが期待できる．

参考文献

[1]. 玉井哲雄,三嶋良武,松田茂広,”ソフトウェアのソフトウェアテスト技法,” 共立出版株式会社,1988.

[2]. Alon Zakai,”emscripten,”<https://github.com/kripken/emscripten>(閲覧日:2018/7/14).

[3]. Jera Design,“JTN002 – MinUniSA minimal unit testing framework for C,”<http://www. jera. com/techinfo/jtns/jtn002.html>(閲覧日：2018/7/14).

Copyright © 2019 Nippon Institute of Technology