2ZG-03

テストケース生成補助に基づくプログラミング学習支援

福本　大介†　　市川　嘉裕†　　山口　智浩†

†奈良工業高等専門学校

1. はじめに

プログラミング講義の演習では，学生が十分な入力値で実行せずに，論理エラーを含んだままのプログラムを提出し，自力でプログラムの動作テストをしないことがある．学生が初期の学習段階で十分な動作確認や修正作業を経験しないことは，論理エラーに気づいたり対処したりする力が養われないことの原因となる．しかし，教員が個別に学生を補助することは，講義時間の制約や人数の都合上困難である．

この問題に対し，我々は学生のテストをツールによって一律に補助することで解決できると考え，プログラミング環境 ACE[1]を開発した．ACE は入力値を予め設定しておくことでテストの手間を削減するツールである．しかし，学生が適切な入力値を設定することができることが前提であり，実際にはテストケース作成の補助も必要である問題が残された．

2. 提案手法

2.1 テストケース作成支援

本研究は，初学者のプログラミング学習を支援することを目的とし，体系的なテストケース作成を支援する手法を提案する．テストケース作成の支援は，制御パステストで用いられる網羅率という概念を軸として，ユーザと支援ツール間で情報をやりとりすることで実現する．

具体的なテストケース作成支援の流れを図 1 に示す．ユーザは，①ソースコードと，②テストケース（プログラムの入力部と正しいと想定する出力部のセット．複数可）をツールに入力し，ツールは，③ユーザから受け取ったテストケースに基づいてソースコードの網羅率（記述された命令の実行率）を計算し，テストケースの正誤判定と併せてユーザに提示する．網羅率が 100%の場合，④-b テストケース作成完了となる．一方で，網羅率が 100%未満の場合，④-a 網羅率を上げるテストケースを自動生成して入力部のみをユーザに提示する．その後，ユーザの行動に戻って④-b に至るまでサイクルが繰り返される．二週目以降では，ユーザはツールから提示されたテストケースの入力部をもとにテストケースを完成させて追加する，あるいは網羅率をもとにテストケースを追加で作成したり，ソースコードを修正したりする．網羅率の計算は，実行時間を考慮し，分岐網羅率（全ての分岐に対して真と偽を実行した割合により求まる）を用いる．網羅率を上げるテストケースの生成にはシンボリック実行によるテスト入力値の自動生成手法[2]を用いる．

シンボリック実行によるテスト入力値の自動生成はプログラムの論理が複雑な場合や繰り返しを含む場合に困難である．本稿では，整数型の変数と if による分岐構造のみからなるプログラムに限定して研究を行う．

入力値の自動生成により，初学者がテストケースを作るのが難しい構造や論理が複雑なプログラムに対してテストの難易度を軽減することができる．演習時に自発的にテストができれば，どのようなテストケースが必要か検討することや，実行経路をトレースする力がつくことが期待できる．

2.2 提案ツール

提案手法を基に Java のソースコードのテストケース作成を支援するツールを実装した．主な機能は，テストケースの編集・実行機能と，網羅率の表示，入力値の自動生成である．網羅率の計算はテストケースの実行時に行われ，出力値と実行結果が一致するテストケースのみを考慮する．入力値の自動生成は，条件を満たした際に，自動生成を実行するかどうか確認するダイアログに，「はい」と答えた場合に入力値を１つ追加することとした．条件は，網羅率が 100%未満かつ，各入力値に対する出力値が正しく設定されていることである．

3. 評価実験

3.1 実験の概要

本実験の目的は，提案手法に基づいて初学者がテストケースを作成できることの確認と，提案手法が初学者のテストケース作成に与える短期的な効果を検証することである．Java のプログラミング講義を履修し始めて１年未満の奈良高専情報工学科 2 年生 40 名を被験者とした．また，各機能の効果を確かめるために，被験者を使用できる機能の有無でグループ A, B, C の 3 つ

Support System for Programming Learners Focusing on Program Testing
†{Fukumoto Daisuke, Ichikawa Yoshihiro, Yamaguchi Tomohiro}・National Institute of Technology, Nara College

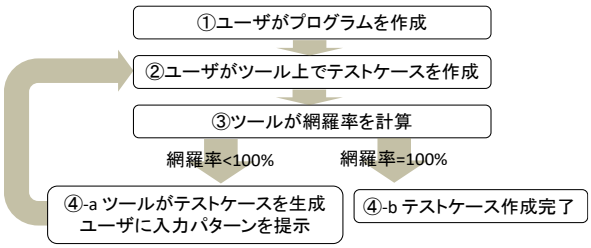


図 1　提案手法のテストケース作成フロー

に分けた．テストケースの編集・実行は全グループ共通であり，網羅率の表示はグループ B，C のみ，自動生成はグループ C のみ使用できる．

3.2　実験の手順

被験者はプログラムの問題と解答ソースコードが与えられ解答ソースコードの網羅率が 100%になるようなテストケースを作成する課題を行う．課題の完了にかかる時間を計測するため，被験者は課題実施の始めと終わりに，ツールに備わった開始・終了ボタンを押す．実際には，ソースコードへの理解度を被験者間で揃えるために，問題と解答ソースコードを配布したあとの 5 分間は，コードを読む時間とし，合図の後の 10 分間を課題に取り組む時間とした．グループ A の被験者は，網羅率 100%を達成できたと思ったタイミングで終了ボタンを押すこととした．課題は全 2 問（P1，P2)を用意した．以上の条件で，各グループにおける網羅率 100%を未達成の被験者数と，課題完了時間を測定した．

3.2　結果と考察

表 1 に各グループの網羅率が 100%未満で課題を終了していた被験者の数とグループ C の自動生成を使用した回数を示す．

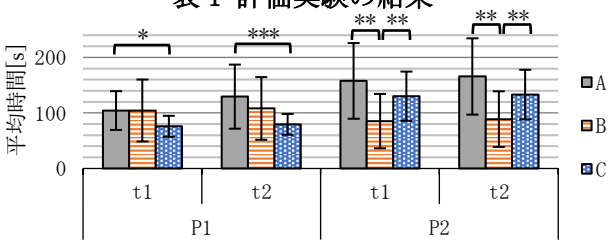
表 1 から，グループ A の 3 名が P2 において網羅率 100%を達成できていない．グループ B には未達成がおらず，自動生成機能による支援効果は確認できなかったが，網羅率の表示が適切な体系的テストを支援できることが示唆された．

また，自動生成機能は利用されていることが分かる．P2 で使用回数が多いのは自動生成を用いた場合に網羅率 100%にするためのテストケースの数が多いためと考えられる．

図 2 に課題ごとの各グループのテストケース作成にかかった平均時間を示す．ここで，開始ボタンを押してから初めて網羅率が 100%になる実行がされた時間を t1 とし，終了ボタンを押すまでの時間を t2 とした．なお，開始・終了ボタン操作のログがない被験者のデータは除外した．また，t1 と t2 の差から明らかに大きいデータについて，グループ B，C では押し忘れと判断して除外したが，グループ A ではその他の要因が考えらえるため除いていない．また，網羅率 100%を未達成のデータも除いている．

		A(n=14)	B(n=14)	C(n=12)
網羅率 100%未達成数[人]	P1	0	0	0
	P2	3	0	0
平均自動生成回数[回]	P1			1.55
	P2			3.27

表 1 評価実験の結果



*…p<0.1，**…p<0.05，***…p<0.01

図 2　各グループの平均時間

まず，グループ A で t1 と t2 の差が大きくなった．この要因に関して，網羅率 100%になった後もテストケースを追加している例が 2 件確認された．この例以外でも網羅率が 100%であることに確信が得られるまでの時間がかかることが考えられる．このことから，網羅率の表示はテスト完了の指標として有用であるといえる．

その他の結果として，平均時間は P1 ではグループ C が短く，P2 ではグループ B が短くなった．網羅率の結果から，比較的簡単な課題であったために，自動生成に必要な操作等に時間がかかったと考える．一方で，標準偏差については P1，P2 共にグループ C が小さい．この原因としては被験者のテストケース作成能力にはばらつきがあり，その一部をツールが補ったことで時間の分散が小さくなったと考える．

4. おわりに

本研究では提案手法に基づいたツールを用いて初学者でもテストケースを生成できることを示した．また，網羅率の表示，自動生成による時間短縮や網羅率の向上はテストの手間の軽減や能力の補助につながると考える．自動生成手法を繰り返しや文字列等にも対応できるよう改善することで，より初学者の支援範囲が広がると考えられる．それにより論理・構造が複雑なプログラムを対象とした検証ができると考える．

参考文献

[1] 福本 大介，市川 嘉裕，山口 智浩：プログラミング初学者のためのリアルタイムテストシステムの試作，第 18 回　情報処理学会関西支部支部大会，E-02，2019.

[2] C. Cader and K. Sen,: Symbolic Execution for Software Testing: Three Decades Later. Commun. ACM, Vol. 56, No. 2, pp. 82-90, 2013.