N-010

プログラミング演習の進捗モニタリングシステムの評価

Evaluation of System to Monitor Students Progress for Programming Courses

内藤 広志* 斉藤 隆† 水谷 泰治‡
Hiroshi Naitoh Takashi Saito Yasuharu Mizutani

1 はじめに

プログラミングの演習では，一般に，教員が教室を巡回して学生の質問に答えることで，学生の進捗や誤りの情報を収集している．そのため，積極的に質問をする学生の情報に片寄ったり，問題の発見が遅れるなど効果的な演習を実施するのは難しい．我々は，学生が作成中のプログラムをソフトウェア工学のテスティング技術を用いて検査し，学生の進捗状況を提示する進捗モニタリングシステム Hercules を開発し，2003 年度より実際に演習で使用してきた．

本報告では，学生の誤りの中から指導の観点から重要なものを見つけるために，Hercules の検査内容の記述法や検査アルゴリズムを改良した結果を述べ，また，JAVA プログラミング演習 [1] の検査結果のログを分析し，Hercules を用いた演習実施法の有効性を考察した．

2 Hercules の概要

Hercules は，図 1 のように課題の評価内容を記述する評価スイート，評価スイートを実行して学生の課題プログラムを評価する評価エンジン，評価結果データから進捗表示ページを生成する出力ジェネレータ，評価エンジンと出力ジェネレータを呼び出して全体の制御をおこなうモニタープログラムから構成される．

Hercules を用いた演習では，学生は Linux マシン上で javac コンパイラや emacs エディタなどの基本的なツールを使ってプログラムを作成する．完成したプログラムは提出メールや HTML フォームを使って提出するのではなく，指定されたディレクトリに保存する．その結果，Hercules は学生が作成中のプログラムを検査して進捗情報を生成できる．なお，学生のプログラムに変更がない場合は評価処理を実行せず，評価結果データ（キャッシュ）を使用して進捗表示ページを生成する．学生のディレクトリは NFS サーバーに格納されているので，複数の Linux マシンを使って評価処理を並行に処理している．

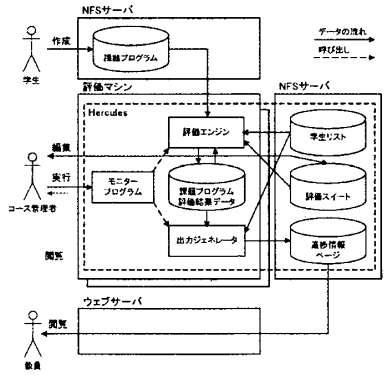

図 1 Hercules のシステム構成

3 検査内容の記述法と検査順

Hercules では，プログラムを実行せずにソースコードを解析する静的検査と，プログラムを実行してプログラムの機能を検査する動的検査の両方を実行する．静的検査は，例題プログラムから変更があるか，コンパイルエラーの原因となるコードがあるかを検査する事前テスト，課題の仕様に合ったプログラム構造であるかを検査する必須テスト，冗長なコードがないかを検査する補足テストに分けて記述する．一方，動的検査は，複数のテストケースに分けて記述し，プログラムの出力が課題の仕様とおりか，メソッドの呼び出し関係や回数が正しいかを検査する．特に，Java アプレットを自動検査するために，ウィンドウに描画する代わりに描画メソッドの名前と引数を標準出力へ出力する擬似 awt パッケージを開発した．実行テストの前に，擬似 awt パッケージを使用するように学生のプログラムを書き換え，Java アプレットを Java アプリケーションとしてエミュレーションして検査する．

これらのテストに含まれる検査項目には，誤りの種類と致命度に対応して，より致命度が高いものから，写（例題プログラムから変更がない），■（課題の学習目標を満たしていない），×（コンパイルエラー，プログラムの構造に誤りがある），▲（実行エラー），□（エラーテストケースの誤り）△（冗長なコード，出力データが正確でない），の 6 個のエラー区分を指定する．プログラムに含まれる誤りをできるだけ検出するため，Hercules の検査アルゴリズムを修正し，評価スイートに含まれるすべての検査項目を実行した後，検出した誤りから最も致命度の高いもの抽出して学生のプログラムの評価としている．

4 進捗情報の提示法

Hercules は，学生のプログラムを検査した結果から全クラス及びクラス別の進捗表示ページを出力する．クラス別の進捗表示ページは HTML フレーム機能を使って記述され，左フレームに各課題の評価日時，ファイル更新学生数，評価値の度数を，右上フレームに課題の誤りのリストを，右下フレームに各学生の座席番号，学生番号，各課題の評価値とファイルの変更日時を表示する．左フレームの課題名をクリックすると，右上フレームにその課題で検出した誤りを人数の多い順に並べて表示する．各行には，誤りの度数（誤りをした学生の数），6 桁のエラー番号，評価値，エラーメッセージを表示する．また，右上

* 大阪工業大学情報科学部情報メディア学科
† 同 情報システム学科
‡ 同 情報システム学科

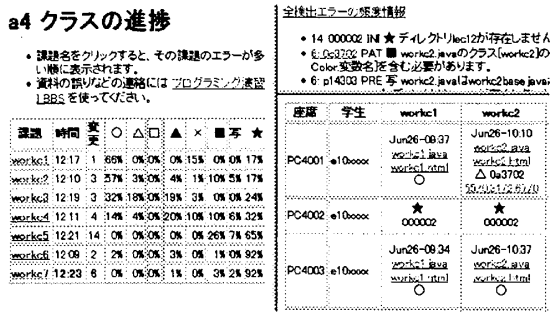

図 2　クラスの進捗表示ページ

フレームに表示している誤りエラー番号をクリックすると，誤りを犯している全学生のソースプログラム，実行結果などを含むエラーの詳細情報を表示する．

5　検査ログの分析

Hercules の検査ログを分析し，教員の指導と学生の反応の関係を調べるために，図 3 のようなグラフを作成した．この折れ線グラフは検査ログを 8 分間刻みに集計したものである．「存在」は課題で作成すべきファイルが存在する学生数，「更新」は 8 分間に変更があった学生の延べ数，「○△」は検査結果が○または△と評価された学生数，「静的検査」は検査結果が■または×と評価された学生数，「動的検査」は検査結果が▲と評価された学生数である．

図 3 は，マウスイベントを処理する課題 [2] の進捗変化である．このクラスでは，この課題に対して次のような指導をおこなった．

- 10:05 学習内容の説明
- 10:15 1 回目のヒント
- 10:25 2 回目のヒント
- 10:40 解答例提示・解説

1 回目のヒントに対して○及び△の学生数が思ったほど増えなかったため，更に 2 回目のヒントを出したと教員は述べている．図 3 では 10 時 16 分から 21 分にかけて「更新」が減り，学生は教員の説明を聞いていることがはっきりとわかる．また，この結果として「○△」の数は増えていない．2 回目のヒントを出した後の 10 時 28 分から「更新」が増え，更に「○△」も増え，半数以上の学生が正解に達したため，10 時 40 分に解答例を公開した．その後，約 10 分間で学生はファイルを修正し，殆どの学生がこの課題を終了したことがわかる．また，「存在」と「○△」の時間差が問題の難易度を示し，学生がこの課題を終了するのに約 1 時間かかる難しい課題であったことが推測できる．

6　処理効率の評価

Hercules の処理効率を評価するため，マウスイベント処理を学習内容とする演習回の各課題の評価スイートを分析した．表 1 に，課題の解答例のプログラムの行数，静的検査用の検査項目の個数，動的検査用の検査項目の個数，動的検査におけるテストケースの個数，1 つの正解プログラムを検査するのに必要な時間（秒）をまとめた．

キャッシュを使うことで変更のあったファイルだけを検査すればよい．なお，キャッシュを使った場合の処理時間は 0.15 秒と非常に短く効果的である．また，6 クラスで約 600 人に対して，12 台のマシンを用いることで 8 分以内に全学生の全課題を検査できている．

表 1　課題の検査時間と検査項目数

課題名	行数	静的検査	動的検査	テストケース	時間（秒）
workc1	50	100	13	2	5.2
workc2	58	88	67	5	8.7
workc3	66	66	43	4	6.9
workc4	37	40	21	6	5.5
workc5	55	49	30	3	8.3
workc6	59	59	42	3	9.0
workc7	91	46	48	2	12.8

7　おわりに

5 章で述べたように，Hercules を用いることで学生の進捗状況に合わせてプログラミング演習の指導が可能である．現在は検査ログから図 3 の進捗グラフを生成しているが，演習中にリアルタイムにこのグラフを生成できれば，指導に対する学生の反応を直ちに知ることができ，よりよい指導をおこなえる可能性が高い．また，従来より厳密な検査をおこない，詳細な検査結果データを生成した結果，検査の処理時間がかかるようになった．このために複数マシンによる分散並列処理をおこなっているが，障害対策や負荷分散の処理は自動化されておらず，人間が監視して処理している．これらの自動化などの改良を検討したい．

参考文献

[1] 齊藤　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).
[2] 内藤広志，斉藤　隆：プログラミング演習のための進捗モニタリングシステム，情報処理学会研究報告 コンピュータと教育研究会報告，No. 13, pp. 33–40 (2008).

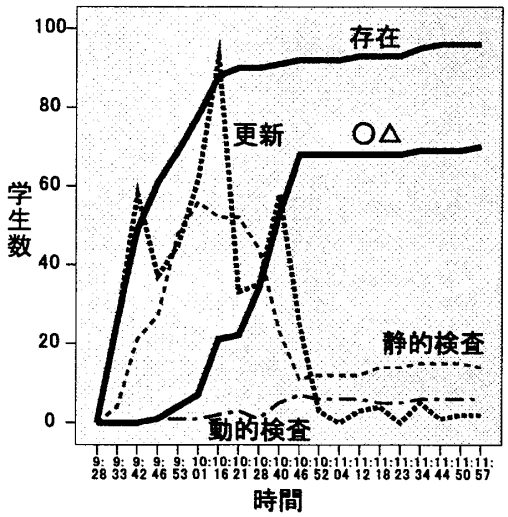

図 3　1 クラスの演習状況の時間変化