1Z-3

拡張可能な診断記述項目を用いた多視点からのプログラム診断システム

高野佑一† 海谷治彦† 海尻賢二†
信州大学大学院工学系研究科情報工学専攻†

１．序論

教育現場で実施されるプログラミング演習では、生徒が作成したプログラムに対して何らかの評価が与えられる。質の高い評価を実現するには、プログラムの持つ様々な要素に対して診断を行う必要があり、評価の自動化を目的とした種々の研究[1][2]が既に行われている。

本研究では、Java で記述されたプログラムに対して複数の視点からの診断を実施するシステムを実現した。システム自身も Java で作成されている。演習ごとに診断手続きに必要となるデータを用意してしまえば、その後の診断作業は自動で行われるため教師の負担・評価に費やす時間を大幅に軽減できる。これは学生にとっても迅速な評価結果を得られるという利点になる。また、プログラムの要件は課題によって異なるため、課題内容に応じて診断項目を自在に選択・追加可能な仕組みを用意した。

２．システムの適用対象

複雑な処理が必要となる課題では、目的の処理を行うプログラムに多くのバリエーションが発生してしまう。そのため正解とみなせるパターンが無数に存在し、診断の自動化が困難となる。したがって、システムの適用対象を以下のように定める。

適用対象：プログラミング初級演習

３．診断の種類

システムが実施する診断は、大別すると実行テストとソースコード診断の 2 種類に分けられる。ソースコード診断についてはさらに、課題ルールチェックとコーディングスタイルチェックに分類される。

Extendable and Integrated Program Diagnosing System
†Yuichi Takano, Kenji Kaijiri, Haruhiko Kaiya: Shinshu University

3.1 実行テスト

プログラムの出力結果に基づいた動作テストを行う。診断対象のプログラムは、CUI ベースのアプリケーションとする。

教師は模範解答プログラムを事前に用意する。このプログラムと診断対象プログラムに同じ入力値を与えて、その出力結果同士を比較することでプログラムの動作が正しいことを検証する。

出力結果の比較方法は、課題に応じて適切な方法を自由に選択できる。また、比較用のメソッドを新たに実装することで、比較方法を新規追加することも可能である。これにより、柔軟性を備えた出力結果比較が実現できた。

3.2 ソースコード診断

ソースコードの記述内容に対する診断である。基本的には、プログラム内の特定要素の有無をチェックすることで診断を行う。

3.2.1 課題ルールチェック

教育目的のプログラミング演習では、課題ごとにプログラム内部で満たすべき条件が指定されている場合が多く、そのような条件をプログラムが満たしているか否かのチェックも必要である。

課題ルール例

- *クラス Shop のサブクラスとしてクラス BookStore を作成する*
- *クラス Shop にメソッド public boolean sell(int number)を定義する*

2.1.1 コーディングスタイルチェック

プログラミング初学者ほどコーディングスタイルを重要視しない傾向がある。コーディングスタイルのチェックを行い評価に反映することで、生徒のコーディングスタイルに対する意識を初期段階から高めることが期待できる。適用したコーディング規約は[3]から抜粋した。

４．ソースコード解析

ソースコード診断ではテキスト形式の Java ソースコードを、JavaML を用いて XML 形式のファイルへ変換する。この XML 形式のファイルを対象に診断が実施される。

4.1 XQuery を用いた診断項目記述

XML 問い合わせ言語である XQuery を用いる。それぞれの診断項目に対応した XQuery を記述し、それをシステムに登録する。

前述した課題ルールの一つに対応した XQuery を以下に示す。

```
1.  <rule name=”extend”>
2.  {
3.  for $c in //class[@name=”BookStore”]
4.  where $c/superclass[@class=”Shop”]
5.  return
6.  <matching>
7.      <location>
8.          <java-source-program>{$c/ancestor::java-source-
    program/@name}</java-source-program>
9.          <class>{$c/ancestor::class/@name}</class>
10.         <method>{$c/ancestor::method/@name}</method>
11.     </location>
12. </matching>
13. }
14. </rule>
```

図 1：課題ルール「Shop のサブクラスとして BookStore を作成」に対応した XQuery

図 1 に示した XQuery の 3・4 行目以外の箇所は、出力する XML ファイル用の固定記述である。そのため、実質的には 3・4 行目の記述部分を考慮するだけでよい。

ある課題についてどのような課題ルールを適用するかという情報は、XML 形式のファイルで管理する。そのファイルの一例を以下に示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<exercise-rule>
  <rule name="extend" exist="true" id="001">
    <description>Shopクラスのサブクラスとして BookStoreクラスを作成する</description>
    <xquery-file name="rule1.xq"/>
  </rule>
  <rule name="method" exist="true" id="002">
    <description>Shopクラスにメソッド sell()を定義する</description>
    <xquery-file name="rule2.xq"/>
  </rule>
</exercise-rule>
```

図 2：課題ルール情報を示した XML

図 2 で示した XML では、１つの*<rule>*が１つの課題ルールを表す。***<xquery-file>*** の name 属性の値で、課題ルールに対応した XQuery ファイルを指定している。

このように、JavaML と XQuery を組み合わせることで、ソースコードの記述内容に対する診断項目を比較的容易に記述することが可能である。本システムの構造は、XQuery をデータとして扱ったデータ駆動型となっている。診断項目の選択・追加作業を XQuery の選択・追加で実現している。これにより、システム本体に変更を加えることなく課題ごとの異なる要件に対応することができる。

4.2 プログラミングによる診断項目記述

ある診断項目が XQuery で記述不可能なとき、プログラミングによる診断項目記述も選択可能である。この場合、診断項目に対応するメソッド名をシステムに登録することで、Reflection API によるメソッドの動的呼び出しで診断を行う。

５．考察

教育現場で以前行われたことのある演習に対して、本システムを実際に適用してみた。はじめに実行テスト用のテストデータおよびソースコード診断用の XQuery を用意してしまえば、生徒の人数に関係なく自動的に診断を行い、個々のプログラムに対する評価を得ることができた。したがって、システムを適用しない場合と比較して、診断作業の負担軽減・費やす時間の大幅な短縮が確認できた。

診断用データを課題ごとに用意することで、システム本体に変更を加えずに様々な種類の課題に柔軟に対応できることが、本システムの最大の利点である。

６．今後の展望

XQuery 作成のサポート機能や既存の XQuery の再利用・カスタマイズ機能を設ければ、より効率的な評価を実現できる。

また、本システムを Web ベース化してプログラムの提出や診断結果の閲覧を可能にすれば、生徒側にとっても利便性の高いシステムとなるだろう。

参考文献

[1] 小河原直行、山本富士男、宮崎剛；非定型 Java 演習問題に対する自動採点システムの試作；情報処理学会第 67 回全国大会
[2] 内藤広志；プログラミング課題の採点システムの拡張可能なフレームワーク；情報処理学会第 68 回全国大会
[3] 電通国際サービス: Java ルールブック