6H-2

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

内藤　広志
大阪工業大学 情報科学部 情報メディア学科

1　はじめに

大阪工業大学情報科学部では，C 言語や Java 言語を使った演習において課題のプログラムを採点システムを使って採点しているが [1, 2]，新たに XML 言語を使った演習のために本システムを拡張した。その際に，本システムの設計を見直し，他の言語への対応を容易にできる採点用のフレームワークを考案し実装したので報告する。

2　採点方法の特徴

プログラムの採点では，プログラムを実行し，入力データに対して正しい結果が出力されたかを調べる動的な分析だけでなく，課題の教育目標を理解しているかを評価するためにプログラムコードを検査する静的な分析が必要である。

本システムでは，確度のたかい採点をおこない，誤りの原因を特定できるテストスイート（採点スイート）を作成するために，まず，課題の解答例から事前に採点スイートを作り，演習時間中に学生のプログラムを採点して誤りの多い項目を見つけ，これらを検査するヒューリスティック（検査項目）を追加・編集するというアプローチをとっている。ヒューリスティックはスクリプト言語（csh 言語）によって記述する。ヒューリスティックの記述を容易にするために，正規表現を指定してソースプログラムや標準出力を調べる多くの検査コマンドを提供している [1]。

3　採点スイートの構造

採点スイートは，課題の「解答例」,「静的な分析」,「動的な分析」から構成される。図 1 は，採点スイートの構造を UML のクラス図で示したものである。1 の多重度は省略している。

「解答例」は，課題でどの言語を使用しているかを採点エンジンが判断するために利用する。通常，1 つのコースで 1 つの言語を使用するが，XML の演習では，XML，XSLT，Java など複数の言語を使用するため，課題毎にどの言語を使用しているかを判断する必要がある。「静的な分析」は，ソースプログラムが必ず満たすべき項目を検査する「必須ソーステスト」と，

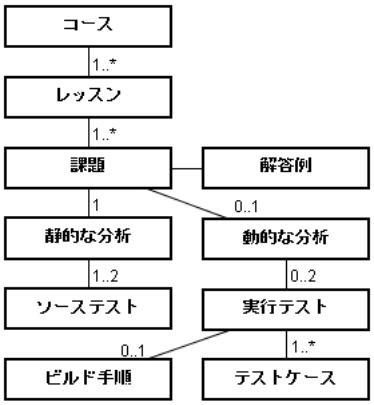

図 1: 採点スイートの構造

冗長なコードなどのアルゴリズムの良し悪しを検査する「オプションソーステスト」から構成される。「動的な分析」は，学生のプログラムそのものを実行する「主実行テスト」と，学生のプログラムの一部分（関数やメソッド）を切り出してテスト対象プログラムを生成し実行する「副実行テスト」から構成される。どちらの実行テストも実行ファイルを生成するための「ビルド手順」と，複数の「テストケース」から構成される。採点スイートの保守が容易になるように，これらのテストは複数のファイルに分けて記述する。

4　採点フレームワークの概要

採点スイートの作成は，2. で述べたように試行錯誤プロセスであるため，採点スイートは容易に変更できる必要があり，記述すべきコード量もできるだけ少ないことが望ましい。また，課題のプログラムの複雑さも多様であり，1 つのソースファイルだけを作成する課題から複数の異なる種類のソースファイルを作成する課題までサポートできなければならない。

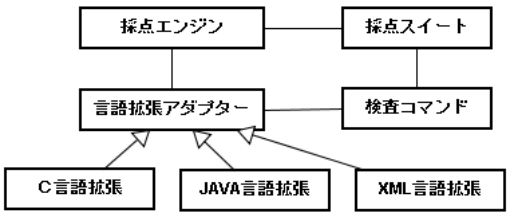

図 2: 採点フレームワークの構成

そのため，本フレームワークでは，図 2 のようにデザインパターンのアダプターを用いて言語固有な処理

A extensible framework for program marking system
Naito Hiroshi, Department of Media Science, Faculty of Information Science and Technology, Osaka Institute of Technology

をカプセル化している。ソース整形やコンパイルなどの言語固有の処理は「言語拡張」として定義する。現在，C言語，Java 言語，XML 言語 (DTD と XSLT 言語を含む) 用の「言語拡張」を実装している。「言語拡張アダプタ」はファイルの拡張子などから言語を識別し，対応する「言語拡張」を呼び出す方式としているので,「言語拡張」を追加するだけで新たな言語をサポートできる。

また，１つのソースファイルからなる単純な課題のためのソースファイルのコピー操作やビルド操作を「言語拡張」に用意することで，課題が複数ファイルから構成される場合だけ「ビルド手順」の記述が必要とすることとし，採点スイートの作成を簡略化している。

5　採点フレームワークの処理フロー

本フレームワークでは，図３のように採点エンジン，採点スイート，言語拡張アダプター，言語拡張が連携して採点処理をおこなう。採点エンジンは，指定されたディレクトリに格納された学生のプログラムをコード変換・整形した後で採点エンジンのローカルなディレクトリにコピーし，実行ファイルをビルドし，採点スイート中のテストを順に実行してゆく。

複数のファイルを作成する課題では,「ビルド手順」にファイル名のリストを記述する。採点エンジンは採点スイートに「ビルド手順」があった場合は，そのリストを「言語拡張アダプター」の「ファイルコピー」操作に指定して学生のファイルをコピーする。「ビルド手順」がない場合は「言語拡張アダプター」の「課題コピー」操作を呼び出してファイルをコピーする。「課題コピー」操作は課題が１つのファイルからなる場合のコピー処理をおこなう。

「言語拡張アダプター」の「ファイルコピー」操作では，コピーと同時にコード変換と整形をおこなうが，この処理は言語によって異なる。通常のプログラミング言語では nkf や astyle などのプログラムをパイプで組み合わせることでコード変換と整形ができる。しかし，XML 言語ではファイルの先頭に書かれた XML 宣言の属性で文字コードが指定されるため，XML 文書をパースし正しい構造（整形式文書）でないとコード変換も整形もできない。そのため,「言語拡張」にコピー操作がある場合は，コピー処理のすべてを「言語拡張」に委ねる。ない場合は「言語拡張」の「整形」操作を用いて「言語拡張アダプター」がコピー処理をする。

実行ファイルのビルドも「ビルド手順」があった場合はそれにビルドを委ねる。「ビルド手順」がない場合は,「言語拡張」の「課題ビルド」操作を用いてビルド処理をおこなう。「課題ビルド」操作は課題が１つのファイルからなる場合のビルド処理をおこなう。

実行ファイルがエラーなくビルドできた後,「必須ソーステスト」,「主実行テスト」,「副実行テスト」,「オプションソーステスト」の順にテストをおこなう。これらの実行順は，採点の目的が評価か診断かによって変える [2]。

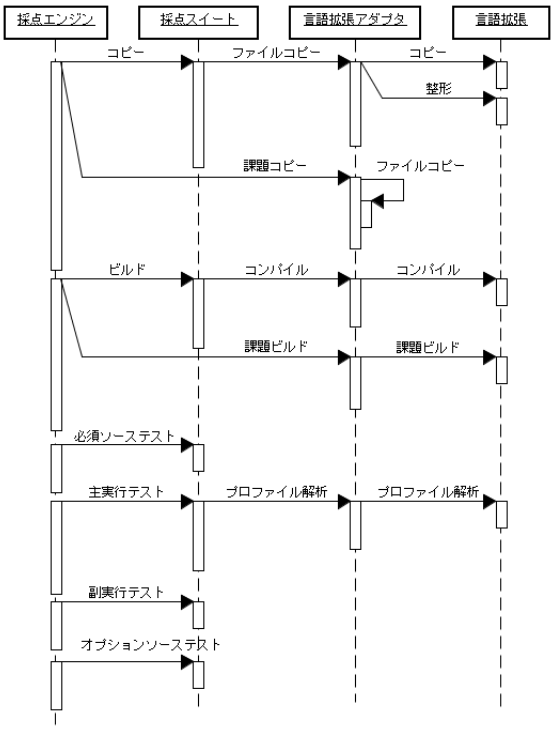

図 3: 採点の処理フロー

6　おわりに

本システムと同じように，拡張性を考慮した採点システムに Oto システム [3] がある。Oto システムは JUnit テスティングフレームワークに基づいて単体テストをおこない，makefile 風に採点内容を記述する。本システムは shell 言語を用いてテストを記述するため，テスト内容を柔軟に記述できる。また，静的な分析もおこなうためソースプログラムの問題点も検出することができる。しかし，ヒューリスティックをスクリプト言語で記述するアプローチのため，採点スイートを作成する作業は多くの時間を要する。この作業を容易にするためのツールを今後は検討する予定である。

参考文献

[1] 内藤広志: ”ヒューリステックを用いたプログラミング演習用の効率的な自動採点システム”, 情報処理学会第 66 回全国大会, 2004.

[2] 内藤広志: ”プログラミング演習の総合支援システムの概要”, FIT 2004, 2004.

[3] Tremblay, G., Guerin, F. and Pons, A. ”A GENERIC AND EXTENSIBLE TOOL FOR MARKING PROGRAMMING ASSIGNMENTS”, Education and Technology 2005, 2005.