社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－IOT－1（11）
2008 / 5 / 8

プログラミング実習におけるコード評価のための eラーニングバックエンドシステムの開発

布目　淳†　福澤　理行†　平田　博章†

† 京都工芸繊維大学 大学院工芸科学研究科 情報工学部門
〒 606–8585 京都市左京区松ヶ崎御所街道町
E-mail: †{nunome,fukuzawa,hrt}@kit.ac.jp

あらまし eラーニングシステムを用いたプログラミング実習において，コード評価を公平かつ効率良く行うバックエンドシステムを開発した．本システムは，提出されたプログラムの文法チェック機能や，類似したプログラムの検出機能，一覧性の高いレポート表示機能を備える．また，低負荷で動作するよう設計されているため，同時間帯に複数のクラスが実習を行う場合でも，レスポンスがほとんど低下しないという特長をもつ．
キーワード eラーニング，プログラミング実習，コード評価，コピー検出，Blackboard, Moodle

Development of an E-learning Back-end System for Code Assessment in Programming Practice

Atsushi NUNOME†, Masayuki FUKUZAWA†, and Hiroaki HIRATA†

† Department of Information Science, Kyoto Institute of Technology
Matsugasaki, Sakyo-ku, Kyoto 606–8585 Japan
E-mail: †{nunome,fukuzawa,hrt}@kit.ac.jp

Abstract An e-learning back-end system has been developed for fair and effective assessment of source codes and reports in programming practice. The system mainly provides three functions: (1) syntax check for rejecting incomplete codes, (2) plagiarism detection by comparing small fingerprints generated from codes, and (3) assessment aids by providing reformatted codes and list views of each answer in reports. In order to suppress the system workload, it is designed to handle answer fragments into which a report is split at the submission. The present system is used simultaneously by about 140 students and works in ignorable time response.
Key words e-Learning, Programming Practice, Code Assessment, Plagiarism Detection, Blackboard, Moodle

1. はじめに

近年，大学などの教育機関ではネットワークを利用した学習支援環境（eラーニング環境）が様々な講義や演習科目で利用されている．これらのeラーニング環境を提供する学習管理システム（Learning Management System; LMS）は，学生（学習者）の使いやすさを重視して設計されており，代表的なLMSでは既に十分な機能が提供されていると言える．また，管理面においても，既存のLMSを基盤にして，各教育機関が必要とする機能を付加する取組み[1]も行われており，学習者と管理者の双方に利点をもたらすシステムとして成長しつつある．

しかし，従来のLMSを教師（採点者）の立場から見た場合，いくつかの問題点が存在する．まず，従来のLMSが様々なコース（科目）に適用できるよう，汎用性を重視した結果，学生からのレポートをそのコースに最適な形で閲覧することができないという問題がある．

例えば，プログラミング実習のように，学生が作成したプログラムを採点する場合では，単純にソースプログラムや考察が表示されるだけでは一覧性が低く，採点の効率が悪い．また，他の科目と異なり，提出されたソースプログラムが言語の文法に従っているか，正常に動作するか，などを個別に確認する必要がある．プログラミング実習に特有のこうした採点業務は，従来のLMSでは特に考慮されていないため，教師の大きな負担となっていた．

一方，学生の中には他人のソースプログラムをコピーして提出する者も少なからず存在する．こうした学生には，教育上，懲罰的な意味合いの減点措置を講じる必要がある．このようなレポートを黙認してしまうと，公平な成績評価ができなくなる．教師はプログラムの内容を評価しつつ，これらのコピーレポートにも注意を払わなければいけないと言える．

これらの採点業務を公平かつ効率良く行うために，従来のLMSとも連携可能なバックエンドシステムを開発した．本システムは，通常のLMSをフロントエンドとして用いることで，学生の使いやすさはそのままに，プログラミング実習の評価に必要な機能をバックエンドから補完することを目的とする．

2. システムの概要と要件

2.1 システムの概要

本システムの原型は，問題提示をWeb上で行い，レポート提出を電子メールで受け付ける(注1)ものであった．その後，安定性と移植性に欠けるファイルロックを使用せずに，レポート提出までがWebベースで行えるよう，内部構造を根本的に作り替えた．

本システムは，学習者向けと教師向けの2つのユーザインタフェースをもつ．ここに提出されたレポートを保存・管理し，必要な処理を行った上で教師の採点業務に提供する形態をとる．

本システムとフロントエンドシステムの間はHTTP [2] プロトコルによって情報交換を行う(注2)ため，それぞれ物理的に異なるマシン上で実行することができる．このように，フロントエンドシステムからの独立性が高いため，フロントエンド処理とバックエンド処理の負荷分散が容易になるという利点がある．また，教師側に必要なソフトウェアが通常のWebブラウザだけでよいという利点も有する．

2.2 システムの要件

我々はプログラミング実習科目の担当経験から，採点時に必要となる作業としては以下のようなものがあると考えた．

プログラムの検証　まず，提出されたソースプログラムが言語の文法規則に沿ったものであるかを確認する必要がある．学生が提出用フォームに書き込む際にコピーミスをする可能性があるだけでなく，理解度の低い学生の中には，未完成の（文法上，プログラムとして成立していない）「プログラム」を故意に提出する者もいる．このチェックを採点時に手作業で行うことは負担が大きい．

コピーレポートへの対応　学生の中には自ら課題に取組まず，他人のソースプログラムをほとんどそのまま提出する者がいる．特に，コンピュータ上でレポート作成が完結するプログラミング実習の場合，他人のソースプログラムをコピーすることは容易であり，実際，難度の高い課題ではコピーの割合が増える傾向がある．公平な成績評価を行うためにも，こうした行為を行う学生に対しては個別に厳重注意をし，採点時には大幅に減点する（または単位を与えない）必要がある．しかし，受講生が多いと，「誰が誰のコピーをしたか」をチェックすることは教師の大きな負担となる．

レポート閲覧の不自由さ　最近では，開講時における学生のプログラミング能力差が大きいため，レベル別の選択問題を提示することが多い．また，前項のコピーレポート対策のために，学籍番号をもとにした問題割当て(注3)を行うことがある．しかし，複雑な割当てを行うほど，同じ種類の問題を連続して閲覧しようとしたときに，操作が煩雑になる問題がある．また，複数の教員やTA（Teaching Assistant）で採点を分担する場合，各自が担当するレポートだけを表示できないと，操作面で効率が悪い．

こうしたプログラミング実習科目に特有の採点業務を支援するために，本システムでは以下の要件を設定し，3つの支援機能を提供することとした．

ソースプログラムの文法チェック機能　学生がレポートを提出した時点で，文法エラーやコンパイラからの警告をチェックする．これによって，文法規則を満たしていないソースプログラムや，構文上の曖昧さを含んだソースプログラムが提出できないようにする．

類似プログラムの検出機能　類似したプログラムを自動的に検出し，コピー元とコピー先の関係が一覧できる機能を提供する．

多様なレポート表示機能　受講生の中から特定の問題を選択したレポートだけ，あるいは特定の設問だけ（全員分のソースプログラムまたは動作結果や考察など）を連続して表示する機能を提供する．

3. 各機能の実装方法

ここでは前節で示した各機能の実装方法を詳述する．

3.1 文法チェック機能

本学の情報工学課程では，プログラミング実習に用いるCコンパイラとしてgcc [3] を採用している．文法をチェックする際には，学生が開発に用いるgccと同じバージョンのgccを利用することで，バージョン間の非互換性が影響しないようにしている．学生に対しては，必ず-Wallオプション(注4)を付けてコンパイルするよう指導し，システム側でも同オプションを付けてエラーまたは警告が出ないことを受理条件としている．エラーまたは警告が出た場合，受理しなかった旨のメッセージと共に，エラーメッセージ，および行番号付きのプログラムを提示することで，問題のあった部分を明示し，学生の再提出を促している．

この機能は言語の文法を満たしていることを確認することが目的であるため，作成した実行ファイルは実行せずに削除している．

なお，本機能の実装はC言語に特化したものではないため，コンパイル後に何らかのオブジェクトコードを作成するコンパイラに対しても適用することができる．実際に，Javaの演習課題で使用した実績がある．Javaのプログラムを受理する際には，gccに含まれるJavaコンパイラ（gcj）を用い，提出されたソースプログラムから正常にバイトコードを作成できるかを判定した．

3.2 類似プログラムの検出機能

3.2.1 類似プログラムの傾向

一字一句，変更箇所のないような，完全なコピーレポートの場合は，diff(1)コマンドのような単純な比較によっても，簡単に検出することができる．しかし学生側もこうした比較を警戒し，ここまで完全なコピーを提出する学生はそれほど多くない．

我々のこれまでの担当経験から，コピーしたソースプログラムには次のような傾向があることがわかった．

(1) 改行位置やインデント幅の変更　ソースプログラムの改行位置やインデントの幅を変更することで，別のプログラムに見

(注1)：受信した電子メールを解析し，提出状況を単一のファイルに追記した．この時にファイルロックを必要とした．

(注2)：実際には，通信の盗聴や改竄などを避けるために，暗号化することが望ましい．

(注3)：例えば，学籍番号の下3桁に対してモジュロ4を求めさせ，その値によって4題の中から割当てるなど．

(注4)：すべての警告表示を有効にするオプション．

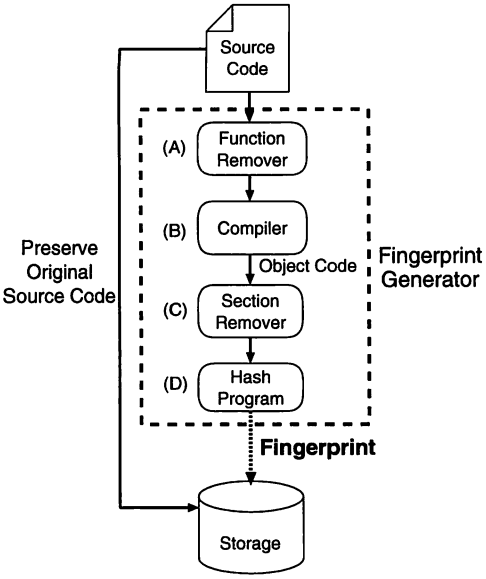

図1　フィンガプリントの生成手順

えるようにする．他にも，ブロックの開始・終了を示す括弧（{, }）の位置を変える手法も見られる．

(2) 表示文字列の変更　入力プロンプトの文字列や，結果を表示する際の文字列を変更する．例えば，英文メッセージから和文メッセージへの変更などが行われる．

(3) コメント文の変更　プログラムの動作に関係のないコメント文を変更する．コピー元となるプログラムの詳細を理解できていないことが多いため，コメントを新規に追加することは少なく，コメントの要約や文言の些少な変更程度である．

(4) 変数名・関数名の変更　プログラム中で用いられる変数や関数の名前を変更する．コピー元で用いられた名前よりも，「より適切な」名前に書き換えるためには，プログラム内容を十分に理解する必要がある．このため，コメント文の変更と同様に，より短い名前に書き換えたり，英語で付けられた名前を日本語のローマ字表記に変更する程度で済ませる場合が多い．

いずれの手法も，プログラミング言語に関する知識はほとんど必要なく，テキストエディタの操作ができれば十分可能という程度である．しかし，高々この程度の変更であっても，前述の diff(1) コマンドでは異なるソースプログラムとして認識してしまう．

3.2.2　特徴の抽出方法

前項で示した4つの傾向を考慮しつつ，既に提出されたソースコードとの間でテキスト比較を行うと，受講者数に比例した処理時間がかかる．

そこで，本システムでは，提出されたソースプログラムに対して，その特徴を示す小さな要約データ（以下，フィンガプリントとする）を作成し，これが等しいものをコピーと見なすようにした．フィンガプリントをハッシュ法で管理することで，受講者数に依存しない処理時間で判定が可能になる．

このフィンガプリントには，前項で示した傾向が影響しないようにする必要がある．このための処理手順を図 1 に示す．

まず，提出されたソースプログラムから特定の関数を取り除く処理（図 1(A)）を行う．具体的には，C 言語であれば printf 関数や fprintf 関数の呼出し部分を取り除く．これらの関数は，前項 (2) のように，プログラム本来の動作と無関係に，画面表示を修飾するために使われることが多いため，フィンガプリントの対象から除外する．

次に，通常のコンパイラによって，オブジェクトコードを作成する（図 1(B)）．この段階で，変数名や関数名の差異はアドレス情報に正規化されるため，前項 (1),(3),(4) の変更は無効化することができる．

その後，オブジェクトコードから文字列定数のセクションを削除する（図 1(C)）．これも，前項 (2) の影響を除外するためである．この処理には，GNU binutils [4] に含まれる strip コマンドを用い，.rodata セクションを削除した．

最後に，処理済みのオブジェクトコードから MD5 [5] の値を求め（図 1(D)），これをフィンガプリントとする．フィンガプリントを保存した後はオブジェクトコードを用いることはないため，削除する．なお，用いる要約関数については，MD5 に限定する必要はなく，衝突可能性を危惧する場合は，SHA-1 [6] などの生成ビット長の長い関数を選択しても構わない．

以上の方法により，各ソースプログラムの特徴を示すフィンガプリントを求めることができる．

なお，学生が提出した元のソースプログラムは，そのままの状態で保存するため，コーディングスタイルやアルゴリズムの評価に用いることができる．

3.2.3　コピー関係の判定

これらのフィンガプリントが一致するレポートをコピーと見なすが，この情報だけではどれがコピー元であるかを判定できない．これに対し，本システムでは提出日時が最も早いものをコピー元と見なすようにした．さらに，コピー元となるレポートが最も早く提出されるよう，学生に対しては提出順位をほぼリアルタイムで公開することにした．これによって，提出順位を成績に考慮していることを暗黙に示し，コピー元プログラムの提供者が，まず自分のレポート提出を済ませることを期待した．

フィンガプリントと提出日時によって，類似レポートの関係を判定した例を図 2 に示す．図の表中，太字はコピー元と判定したレポート，数字は同じフィンガプリントになるレポート数，英字は同じフィンガプリントの系列，丸印は他に同じフィンガプリントがなかったオリジナルをそれぞれ示す．例えば，課題 08 において，学籍番号 001, 005, 009, 012 は同じ系列のコピーレポートである．そのうち，提出日時が最も早いために，学籍番号 005 のレポートをコピー元と判定していることを示している．また，下線部はリンクになっており，クリックすることで当該レポートを表示できる．

3.2.4　誤判定と問題点

本機能はソースプログラムの特徴で分類するために，コピーしていないソースプログラムがコピーと判定されたり，コピーしたソースプログラムがオリジナルと判定される可能性がある．前者の誤判定は，図 2の課題 01 や課題 02 のように，簡単な（誰が作成しても同じようになる）課題で顕著に見られた．このような課題については，判定自体を無視して対処した．

一方，後者の誤判定は，バッファに用いる配列の要素数を変えるなど，オブジェクトコードとしては異なるが，一見すると同じように見えるというもので稀に見られた．こちらについては，問題文で配列の要素数を指定するなどの工夫が必要である．

レポートコピー状況

2月1日15時00分現在

学籍番号	課題01	課題02	課題03	課題04	課題05	課題06	課題07	課題08	課題09	課題10	課題11	コピー率
001	74a	71a	○	○	○	○	○	11a	○	○	○	27.3%
002	74a	71a	9b	○	○	3b	○	5b	○	2f	○	45.5%
003	74a	○	4d	○	○	○	○	○	○	○	○	18.2%
004	74a	71a	2h	○	○	○	○	○	○	○	×	20.0%
005	74a	71a	○	○	○	○	○	11a	3c	○	○	9.1%
006	74a	71a	4e	○	○	○	○	○	2e	○	○	27.3%
007	74a	2b	○	○	○	○	○	2f	○	○	○	18.2%
008	74a	71a	○	○	○	○	2b	○	○	5a	○	18.2%
009	74a	71a	5c	○	○	○	○	11a	○	○	○	36.4%
010	74a	71a	○	○	○	2c	○	○	○	○	○	27.3%
011	74a	71a	2k	○	○	○	○	○	○	○	○	18.2%
012	74a	71a	5c	○	○	○	○	11a	○	○	○	36.4%

図2　類似レポートの検出例

3.3　レポート表示機能

本システムでは，提出されたレポートを設問単位(注5)で分割し，それぞれを独立したファイルに保存するようにした．こうすることで，レポート単位や設問単位で表示する場合でも，基本的にファイルの連結操作だけで済むため，システムの負荷軽減につながる．

さらに，レベル別選択問題や学籍番号に基づく割当て問題については，個別に用意した割当てテーブルをもとに表示する．前者用の割当てテーブルは学生から教師への申告によって作成するが，後者用の割当てテーブルは機械的に作成することができる．

また，プログラミングの初心者は，インデント幅や括弧の位置が適切でないソースプログラムをそのまま提出することが多い．こうしたソースプログラムは文法上は問題ないものの，教師が読みにくく，コードの構造を理解しにくい．このため，本システムにはコード整形プログラムである Artistic Style [7] で整形して表示するオプションも実装した．これによって，必要に応じて教師が読みやすい形式に整形できるため，従来よりもソースプログラムを理解しやすくなった．

4.　本学での運用実績

本学学部課程のプログラミング実習科目である「ソフトウェア演習 I，II」を対象に，平成15年度から本システムの運用を行っている．運用当初のシステム環境を表1に示す．運用当初はHTMLフォームをフロントエンドとし，LMSに相当する課題管理・提示機能も実装していた．当時としても高速とは言えない環境であったが，授業時間中のロードアベレージは0.3〜0.7付近を推移する程度で，過負荷状態や応答時間が遅くなることはなかった．

その後，本学の情報科学センターが提供する汎用のLMSと組み合わせることで，課題の管理と提示については，それらのLMS側で対応することとした．具体的には，1クラス約70名の受講生を，平成18年度はBlackboard [8] で，平成19年度はMoodle [9] でそれぞれ管理した．

(注5)：作成したプログラムや動作結果，考察など．

表1　導入当初のシステム環境

Processor	UltraSPARC 200MHz
Memory	512MB (ECC 60ns SIMM)
OS	Solaris 8

表2　現在のシステム環境

Processor	Opteron 248 2.2GHz (Dual)
Memory	4GB (PC2700 Registered ECC)
OS	FreeBSD 6

また，平成18年度からは，表2に示すシステムに移行した．ハードウェア性能の向上に加え，前述のようにフロントエンドシステムと分離したため，2クラス並行に実施した場合でも，授業時間中の負荷はほとんど考慮しなくて良いレベル（ロードアベレージは0.1未満を推移）にまで低減した．

なお，本システム内で最も重い処理を行う部分は，3.2.2節で説明した，フィンガプリントの作成処理である．現在のシステム環境では，50行程度のCプログラムのフィンガプリントを，全受講生（71人）分，一括して作成するのに3.8秒を要した．実際には，学生がレポートを提出すると同時に作成されるため，学生が送信ボタンを押してから約0.05秒程度がフィンガプリントの作成に費やされることになる．この程度の遅延時間では，学生から苦情が出ることは全くなかった．

4.1　Blackboardとの連携

Blackboardでは，レポートの設問のうち，通常のテキストはBlackboardの解答フォームで，ソースプログラムは任意のファイルを送信する機能である「ディジタルドロップボックス」機能でそれぞれ受け付けた．このディジタルドロップボックスは，

教師側から見ると，全学生の全課題に対する提出ファイルが全く分類されずに，時系列に沿って列挙されるだけであったため，一覧性が非常に悪いという問題があった．また，1つの課題に対して複数回ソースプログラムを提出した場合，どれが最新のものか（評価対象か）が分かりにくいという問題もあった．

このため，Blackboard のディジタルドロップボックスから，定期的にファイルを一括ダウンロードし，本システムに改めて登録するスクリプトを作成した．これによって，プログラムの採点業務を Blackboard 上ではなく，本システム上で行えるようにした．こうした問題から，演習の後期では，ディジタルドロップボックスでの受付を中止し，本システムへ直接提出するように，運用形態を変更した．

このように，Blackboad には教師がファイルを受け取り，システム上で効率良く一覧する手段が用意されていなかったため，密な連携ができなかった．また，HTTP の Basic 認証 [10] のような一般的な認証方式ではなく，Java Script を用いた独自の認証方式であったため，非対話型ツールによる自動化ができなかったことも問題であった．

4.2 Moodle との連携

Blackboard と異なり拡張が可能な Moodle との連携では，Moodle 側から CGI 経由で認証情報を提供してもらうことで，シングルサインオンが実現できた．Moodle では教材の提供と出席管理，小テストだけを行い，レポートの提出はすべて本システムで受け付けるという運用形態が確立した．これによって，学生は Moodle 上でレポート提出から提出状況の確認までのすべてが完結するかのような使用感が得られるようになった．

4.3 汎用 LMS との連携に必要なインタフェース

Blackboard では，学生は Blackboard での認証に加え，レポート提出の度に本システムでの認証を行う必要があり，使い勝手は悪かった．使い勝手を改善するためには，フロントエンドシステムから認証情報を提供してもらうようなインタフェースが必要と言える．

一方，Moodle ではほぼ問題のない環境になったが，教師側から見ると，小テストの成績が Moodle 上に，それ以外のレポートが本システム上にと，成績情報が分散することになった．成績情報を集中管理するためには，フロントエンドシステムにある成績情報を安全に取り出せるようなインタフェースが備わっているべきである．

また，プログラミング実習というコースの性質上，提出されたソースプログラムのファイルを，そのままの状態でダウンロードできることが必要である．ソースプログラムの前後に，コンパイラにとって不要な情報が付加されると，コピーチェックのための処理や教師によるコンパイル作業がスムーズに行えない．さらに提出されるソースプログラムの数が多くなるため，これらを一括して取得できることが望ましい．

5. おわりに

本稿では，プログラミング実習のための e ラーニング環境をバックエンドから補完するシステムについて述べた．本システムでは，プログラミング実習のコースを構築する上で，従来の汎用 LMS では実装されていない機能，特に学生から提出されたソースプログラムを効率良く評価するための機能を実装した．これにより，教師は基本的な文法チェックやコピープログラムの判別のような作業から解放され，本来のコード評価に専念できるという利点がある．特にコピーレポートを放置することは，公平な成績評価に反し，学生の学習意欲の喪失にもつながるため，このチェックを半自動化できる意義は大きい．また，プログラミング実習に特化したレポート表示機能を提供することで，コースを複数の教師が共担する場合においても，各自が担当する課題の評価を見落としなく行えるという利点もある．

今後は，プログラムの動作結果も自動的に検証する機能や，本システム自体の管理を既存 LMS と同等のユーザインタフェースで行える仕組みを導入したい．

謝辞　本研究の一部は，日本学術振興会科学研究費補助金（課題番号 18700056）の補助による．

文　献

[1] 米満 潔，梅崎卓哉，藤井俊子，江原由裕，穂屋下茂，角 和博，高崎光浩，大谷 誠，大月美佳，皆本晃弥，岡崎泰久，渡辺健次，近藤弘樹：“Moodle と XOOPS を基盤とした大学の要求を考慮した学習管理システムの開発と運用”，情報処理学会論文誌，Vol. 48, No. 4, pp. 1710–1720, 2007.

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee: “Hypertext Transfer Protocol — HTTP/1.1,” RFC 2616, June 1999.

[3] Free Software Foundation, Inc.: “GCC, the GNU Compiler Collection — GNU Project — Free Software Foundation (FSF),” `http://gcc.gnu.org/`, 2008.

[4] Free Software Foundation, Inc.: “GNU Binutils,” `http://www.gnu.org/software/binutils/`, 2007.

[5] R. Rivest: “The MD5 Message-Digest Algorithm,” RFC 1321, April 1992.

[6] National Institute of Standards and Technology: “Secure Hash Standard,” Federal Information Processing Standards (FIPS) Publication 180-1, April 1995.

[7] Jim Pattee: “SourceForge.net: Artistic Style,” `http://sourceforge.net/projects/astyle/`, 2008.

[8] Blackboard Inc.: “Blackboard — Educate. Innovate. Everywhere.,” `http://www.blackboard.com/`, 2008.

[9] Moodle: “Moodle — A Free, Open Source Course Management System for Online Learning,” `http://moodle.org/`, 2008.

[10] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart: “HTTP Authentication: Basic and Digest Access Authentication,” RFC 2617, June 1999.