Vol.2014-CLE-14 No.6
2014/10/25

Mercurial と Eclipse をベースとしたプログラミング演習 課題レポート管理システムの構築

佐藤究[†1]　山下圏[†1]　小嶋和徳[†1]　小笠原直人[†1]　布川博士[†1]

近年,大学の情報・工学系学部においては,プログラミング演習が必須となっており,演習課題では プログラムソースの提出が一般的である.しかし,課題内容が高度になると,一つの課題に対してディレクトリ構造に依存した複数のソース,データ,ドキュメントのファイルを提出する必要がある. また,採点者にとってはこれらの提出物を実行し採点,評価を行う必要があり,それを実行可能状態にするだけで大きな手間となる.さらに本学部においては課題が合格するまでの再提出を学生に課しているため,このバージョン管理も大きな手間である.
本稿では,これらを解決するための, 分散型バージョン管理システムである Mercurial と統合開発環境である eclipse をベースとしたプログ ラミング演習課題レポート管理支援システムの構築と,これらのベースとなるシステムの機能を有効 に利用した課題管理について述べる.

Constriction of Programing Exercise Report Management System based on Mercurial and Eclipse

Kiwamu Sato[†1]　Yamashita Meguru[†1]　Kazunori Kojima[†1]　Naohito Ogasawara[†1]
Hiroshi Nunokawa[†1]

In recent years, in information science and engineering faculties of the university, programming exercises is indispensable. Submissions of program source are generally required in the exercises. In advanced exercise, it is necessary to submit many source, data, and document files that depend on the directory structure.
Grader performs the evaluation by executing these submitted programs. This is a great effort so. It is imposed on students re-submissions until the pass in our faculty. This version management is also a big hassle.
In this paper, we describe programming exercises report management support system based on the Eclipse that is an integrated development environment and Mercurial that is a distributed version control system, and management of submission used by this system.

1. はじめに

中，上級のプログラミング演習においては，いわゆる古典的なプログラミング演習のスタイルが用いられることが多い．古典的なスタイルとは，

（1）学生が課題に基づきプログラムを記述
（2）解答を教員に提出
（3）教員が提出物を採点，評価し，必要であれば再提出を要求
（4）学生が解答の訂正をし，再提出をする

というものである．

このような，演習スタイルにおいて問題となるのは，教員の負荷である．負荷要因として，作題，採点，再提出を含むレポート管理[1][2]があげられる．作題，採点に関しては．自動化による支援等の様々な研究が行われている．

しかし，中級以上の学生では，穴埋め等の簡単な課題は不適切であり，また，形式的ルール等による採点が可能な厳密なコードを書くことも困難である．そのため，各課題について教員が手作業によって採点を行う必要がでてくる．また，課題が高度になると，一つの課題においても，複数のソース，データ，ドキュメントのファイルが提出しされるため，これらの内容を眺めるだけでは採点をすることは困難であり，提出物を実行可能の状態にしたうえで，実行を行いながら採点，評価を行う必要がある．

さらに，その評価の結果やコメントは学生に伝えたうえで，速やかに再提出を要求する必要がある．再提出においては，前回からの変更点を把握し，実行，採点，評価を行う必要がある．

そこで本研究では，作題，採点の自動化ではなく，課題提出，採点，評価，再提出の一連のサイクルで構成される課題レポート管理の円滑化による支援を行うことにより，教員の負荷の軽減をはかるためのシステムの構築と実践について述べる．

本研究では，以下の５点を目標とした．

（1）複数のソース，データ，ドキュメントのファイルの容易な提出，再提出と管理
（2）提出物を速やかに実行可能の状態にする
（3）外部ツール等による採点支援
（4）速やかな再提出，再採点のサイクルの支援
（5）再提出時の変更点の容易な把握

†1 岩手県立大学ソフトウェア情報学部
Faculty of Software and Information Science, Iwate Prefectural University

また，本研究はシステム構築，運用の容易性および低コスト化，他システムとの連携性を考慮したオープン性を実現するために,既存のシステムを最大限に活用した構築,をもう一つの目標としている．

2. プログラミング演習とレポート管理

2.1 本学部におけるプログラミング演習

本システムは,岩手県立大学ソフトウェア情報学部（以下，本学部と呼ぶ）における Java 言語によるプログラミング演習のスタイルを対象として，設計されたものであるため，最初に本学のプログラミング演習について説明する．

本学部では，1 年次から各講座に学生が配属（1 講座 10 人程度）され，プログラミング演習は学部共通のテキストを用いた古典的なプログラミング演習のスタイルによって各講座ごとに実施されていた．このように少人数教育として設定されていたため，シンタックスやロジックだけではなく，コメントやプログラミングスタイルに対しても指導が可能な密度の濃い演習であった.また少人数であるため，指導教員数は多いが，各教員の負荷はある程度抑えられていた．

しかし，カリキュラムの変更により，昨年度から約 90 人＊２クラス（演習室において同時刻に開講，各ラス教員１名，TA５名）で実施される演習となった．この演習は，以前の演習の形式を踏襲しており，古典的なプログラミング演習のスタイルで実施され，プログラミング課題が毎回２，３題課せられるとともに，課題は合格するまで何度でも再提出することが要求されている．

以上の状況のもとで，本システムが対象としたのは，２年次前期に実施される Java 言語のプログラミング課題である．なお，1 年次後期に同様のスタイルで，C 言語の演習が行われるため，学生はプログラミングについて最低限の素養を身につけている．

2.2 本学部における Java 言語によるプログラミング演習

本学部における Java 言語によるプログラミング演習は,オブジェクト指向を Java 言語を通して身に付けることを目的としている．授業テキストには市販のものを使用しており，演習の前半で教員による説明，解説が行われ，後半で実際に学生が手を動かしてプログラムを書く形式である.演習内容は，オブジェクト指向の基礎から始まり，それらの Java 上での実現方法，さらに応用としてファイル入出力，servlet，Swing 等までを行う．最終課題としては GUI を用いた大きめの課題提出がある．学生の提出物は，基本的にはプログラムであるが，課題によってはテキスト形式のレポートもある．

本演習では，学生のプログラミング環境としてコマンドライン環境ではなく，オープンソースの統合開発環境である Eclipse を採用することとした.統合開発環境による教育には様々な議論があるが，以下の３つの理由から Eclipse を採用することとした．

（１）コマンドラインによる開発は 1 年次前期に行っており基本原理については理解済みである

（２）本学部の卒業生はソフトウェア業界に就職することが多いため，統合開発環境についても，ある程度の習熟が要求される

（３）本システムのフロントエンドとして利用させるため

2.3 教員の演習支援とレポート管理

以上のように本演習では，約 90 人から毎週 2，3 題のプログラム課題が提出され，合格するまで再提出が繰り返されるため教員の負荷が非常に高いものと予想された．

負荷要因の一つは再提出を含むレポート管理である．前述のように大量の課題が提出，再提出されることによるバージョン管理と再提出の要求の業務が大量に発生する．そこで，我々はこのレポート管理を対象とした支援システムを構築することにした．

もう一つの負荷要因としては採点があげられる．採点の自動化[3-8]を行うことも検討されたが，中級のプログラミング演習のため，穴埋等による簡単な課題は不適切であり，また JUnit 等を用いた形式的テストを行えるほどきちんとしたソースを期待できる学生のレベルに到達していない．そのため，今回は採点の自動化ではたく，上記システムを様々なツールと連携可能形で構成することにより，それらのツールにより間接的に採点の支援をする環境の構築を目指すこととした．

このようなレポート管理システムに要求される要件は以下のようになる．

（１）学生にとっての容易なレポート提出

Java 言語では，クラスごとにソースファイルを作成することが一般的であり，また入力用のデータファイルなどが必要な課題もあるため，複数ファイルまとめて提出する必要がある．しかし，zip 等でまとめてメールや web で提出することは,手間がかかり間違いも起こりやすい.また，学生も再提出によって生じるバージョン管理を行える必要がある．さらに，演習室以外で課題を進めるため，ソースファイルや実行設定を用意に移行できることが必要である．

（２）提出物を速やかに実行可能の状態にする

採点者は，これらの内容を眺めるだけでは採点をすることができないため，各学生ごとにこれらの提出物を実行可能の状態にしたうえで，実行を行いながら採点，評価を行える必要がある．

（３）外部ツール等による採点の支援

本システムでは，システム自体で評価支援を行うことを目的としていない．しかし，外部ツールと連携可能とすることにより，効果的な採点自体の支援を行うことを可能する．

（４）速やかな再提出，再採点のサイクルの支援

本演習では，採点，再提出のサイクルが頻繁に発生す

るため，教員から学生への評価結果を速やかな伝達．学生の速やかな再提出，さらに，速やかな再採点のサイクルを可能とする必要がある．

（５）再提出時の変更点の容易な把握

再提出が複数回生じるため，前回提出分とどのような違いがあるのかを容易に把握することにより，必要な部分だけを再採点することが可能となり採点負荷を下げることができる．

（６）既存のシステムを最大限に活用した構築

システム構築，運用の容易性および低コスト化，他システムとの連携性を考慮したオープン性を実現するために，既存のシステムを最大限に活用した構築を行う．これにより，授業スタイルの変化等にも容易に対応ができると考えられる．

3. プログラミング演習課題レポート管理システム

本システムは，前節で述べたように，既存のシステムや機能を最大限に利用し構築することを目標としており，web 上のサービスである採点管理システム，提出物を管理するための分散型バージョン管理システムである Mercurial[9]，および学生と教員のフロントエンドである統合開発環境 Eclipse[10]からなる．本システムは，Apple iMac（CPU：Core i7 3.5GHz，メモリ：32G）上で運用されている．

3.1 採点管理システム

採点管理システムは，主に以下の３つの機能を持つ Web サービスである．

（１）学生に対して課題の合格状況と不合格の場合のコメントを提示する機能

（２）教員に対して，学生および各課題の合格，不合格の一覧と，結果およびコメントを入力する機能

（３）学生，および教員に対して課題への URL リンクを提示する機能

採点管理システムは，python ベースのオープンソースの Web フレームワークである Django を使用し，python と HTML で記述されている．課題の合格状況等の情報は sqlite3 によって DB 化されている．

学生は，本システムを利用することにより，提出済みの課題の合否の一覧，不合格の場合のコメントを見ることができる．また，提出済み課題への web ベースのリポジトリブラウザ（次節で述べる）へのリンクをたどることができる．このリンクは課題提出時のリポジトリアドレスを兼ねているため，毎授業において学生は課題の合否をチェックすることになる．

教員は，本システムを利用することにより，ある学生の全ての課題情報（合否，提出日時等），または，ある課題の全ての学生の課題情報の一覧の閲覧，および合否登録とコメント登録をすることができる．

本システムにおいては，同様の機能をもつ学内のシステムが存在したが，クローズシステムであり仕様が公開されていないため，学内システムの利用を断念し自作することとした．将来的には，同種の機能を持つ Moodle[11]等との連携を視野に入れている．

3.2 提出物管理システム

本システムでは提出物管理に，オープンソースの分散型バージョン管理システムである Mercurial を採用した．Mercurial は基本的に python で記述されており，python で記述された採点管理システムと親和性が高く，また，および簡単な設定で web ベースのリポジトリブラウザを提供できるため，これを採用した．

分散型バージョン管理システムとは，一般にプログラミング開発等で利用されるもので，開発で必要な複数のファイルをまとめたもの（以降，プロジェクトと呼ぶ）をバージョンごとに登録，管理，再配布することを可能とするシステムである．本システムでは，この機能を利用することにより提出物管理を行う，登録されたプロジェクトは，ファイルシステム上の各学生に対応したディレクトリの下に，各課題ごとのディレクトリとして保存される．この課題ディレクトリはプロジェクトの再提出も含む全てのバージョンで共通に利用される．

具体的には，学生は，一つの課題に必要なファイルをプロジェクトとして作成し，フロントエンドである Eclipse から容易に一括して登録することができる．この登録が課題提出となる．再提出時においても追加，編集したファイルを同様に Mercurial に登録することによりバージョンを管理した課題提出が可能である．また，演習室以外で課題を行う場合も再配布の機能を利用してプロジェクトを取得し課題を続行することが可能になる．

教員は，この提出物管理システムを利用して，学生の提出物を取得することができる．取得の方法としては，（１）Mercurial 付属の web ベースのリポジトリブラウザを利用し web ブラウザ上でプロジェクト内のファイルを表示，（２）Mercurial の再配布機能によるプロジェクト取得，（３）課題ディレクトリが保存されたファイルシステムのアクセス，の 3 種類が可能である．テキスト形式の課題レポートのように，ファイル内容を読むだけで採点できる課題の場合は，（１）のリポジトリブラウザを利用する．リポジトリブラウザでは登録済みの全バージョンの全ファイルを閲覧することができ，バージョンごとの差分のみを表示させることも可能である．プログラム形式の課題レポートの場合，3.5 で述べるように（２），（３）によってプロジェクトを取得可能である．

3.3 フロントエンド

フロントエンドとして，学生，教員とも Mercurial プラグインをインストールした統合開発環境 Eclipse を利用す

る. Eclipse は Java 言語を始めとする様々なプログラミング言語に対応可能なオープンソースの統合開発環境である. プロジェクトによるプログラムの管理機能が充実しており, プロジェクトの作成からコンパイル, 実行, デバッグまでを一貫して GUI を通して利用可能である. リファクタリング, コード支援, デバッグ支援機能も充実している. また, プラグインによる容易な機能拡張が可能である.

学生は, プログラム形式の課題は Eclipse の通常の Java プロジェクトとして作成する. 実行, デバッグ等も Eclipse の機能を利用しておこうことが可能である. Java プロジェクト作成後にプロジェクトを Mercurial のリポジトリとして設定しておくことにより, 課題の提出はコミットと, 提出物管理システムへのプッシュで行うことができ, これらはメニューから簡単に実行することができる. 再提出も, ファイル等を編集後, コミットとプッシュを行うだけで自動的にバージョン管理のもとで提出することができ, また, コメントを付加することもできる. テキスト形式の課題は, 通常のプロジェクトとして作成する. プロジェクトにテキストファイルを作成, 追加し, Java プロジェクトと同様にコミットとプッシュで提出する.

教員は, プログラム形式の課題レポートの提出物を取得し, 速やかに実行可能の状態にし, 実行, 採点するために Eclipse を用いる. 課題ディレクトリが保存されたファイルシステムから, Eclipse の外部プロジェクトの一括インポート機能を用いることにより, ある課題の全学生の提出物を一括して実行可能状態にすることができる. インポートされた全学生のプロジェクトは, メニューからすぐ実行できると同時に, 過去のバージョンのソースファイルやデータファイルも容易に参照が可能であり, 差分の表示も可能である.

3.4 学生の利用手順

（1） 採点管理システムの利用

授業の最初に学生は採点管理システムにログインする. ログインすると図 1 の画面が表示され, 課題の合否, 教員のチェック日時, 教員からのコメントの有無が表示され, 自分の各課題の合否状況を把握することができる. 通信欄が赤くなっている場合, コメントボタンを押すと, 図 2 の様にその課題に対するコメントが表示される.

（2） 課題プロジェクトの作成と提出

毎授業の各課題ごとに課題番号が指定されるので, それに基づき Eclipse を用いてプロジェクトを作成する. 作成時には, 図 3 のコミットダイアログにて提出先 URL, ユーザ番号, パスワードを入力しリポジトリの設定を行っておく. 提出先 URL は図 1 の課題番号のリンク先 URL である.

プログラムが完成したならば, 図 4 のプッシュダイアログにおいて, 必要なファイルの選択をし, 提出時のコメントを入力する. この後, プッシュを実行すると課題が提出される. 再提出は, プログラムを修正し, 同様に図 3 のコミット画面において再提出時のコメントを入力しプッシュを行うだけである.

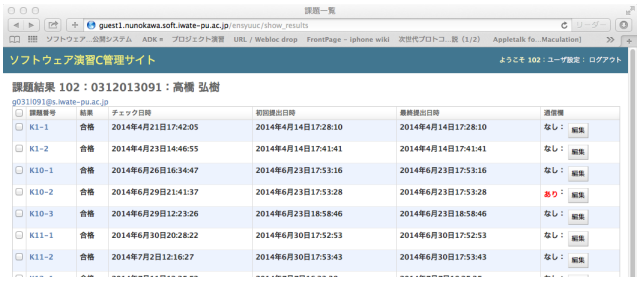

図 1 学生のログイン画面

Figure 1 Top Page for Student.

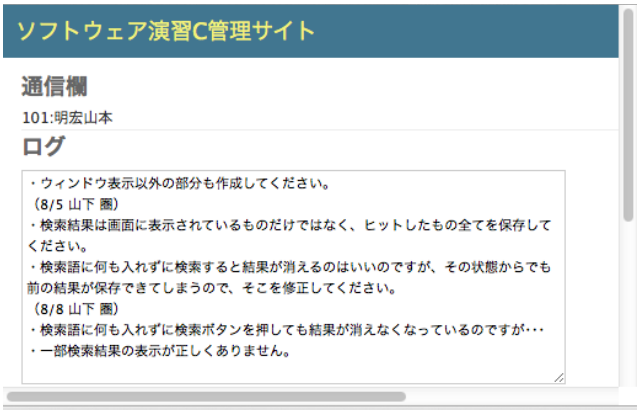

図 2 コメント画面

Figure 2 Comment Page.

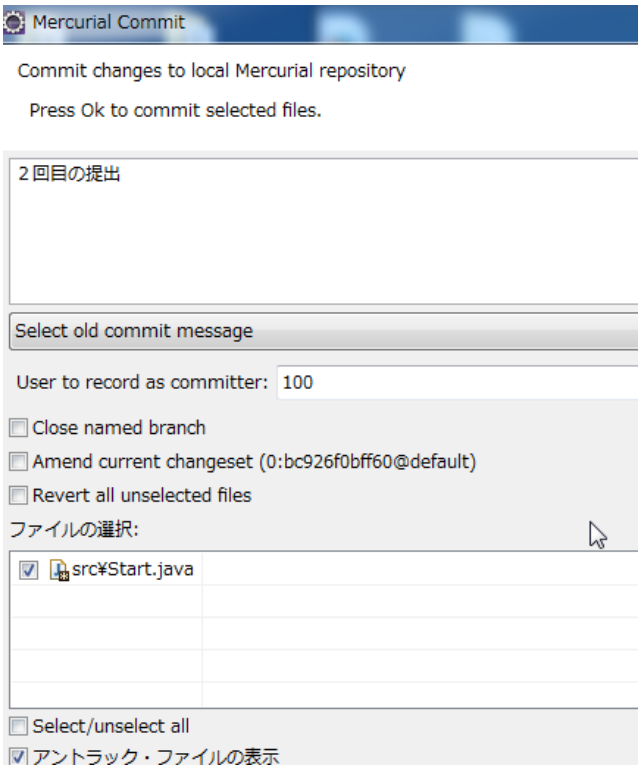

図 3 コミットダイアログ

Figure 3 Commit Dialog Box.

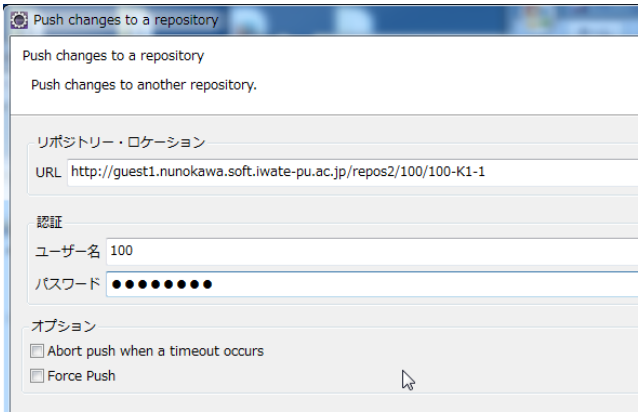

図 4　プッシュダイアログ
Figure 4　Push Dialog Box.

3.5 教員の利用手順

（1）　採点管理システムの利用

教員が採点管理システムにログインすると，図 5 の課題管理画面が表示される．本画面ではユーザごとの合否の参照・入力，およびコメントの入力を行うことができる．また，ユーザ番号が，そのユーザのその課題の提出リポジトリのリンクになっている．

（2）　課題リポジトリの閲覧

上記のユーザ番号部分をクリックすると，Mercurial が提供する，そのユーザのその課題の提出リポジトリブラウザ画面に遷移する．この図 6 はユーザ番号 123 の第 2 回目の授業の 2 個目の課題のリポジトリであり，2 回目の提出が行われている状況を示している．Description の欄が図 3 で学生が入力したコメントになっている．右側の browse をクリックすることにより図 7 の様な提出ファイル一覧が表示され，これをさらにクリックすることによりファイルの内容を閲覧することが可能である．この図 7 の課題は，テキスト形式のレポートのため.project と report.txt しか表示されていないが，プログラム課題等で複数のファイルを提出した場合はここに全てが表示される．

（3）　バージョン差分の表示

図 6 の様に再提出（2 回目の提出）がある場合は，description の 2 回目の提出をクリックすることにより，diff 形式で，一回目の提出との差分を閲覧することができる（図 8）．この画面では，Q4 の小問 b,c, Q5 の小問 d が 2 回目で，それぞれ，「オ」→「デ」，「オ」→「手」，「差分プログラミング」→「継承」に変更されたことが表示されている．これにより，変更部分のみを速やかに把握，採点することが可能になる．

（4）　プログラム課題の一括取得

課題ディレクトリが保存されたファイルシステムを経由して，学生が提出した課題を一括して Eclipse にインポートすることにより自動コンパイルが行われ，全員の課題を速やかに実行可能な状態で取得することができる．

図 5　課題管理画面
Figure 5　Exercise Management Screen.

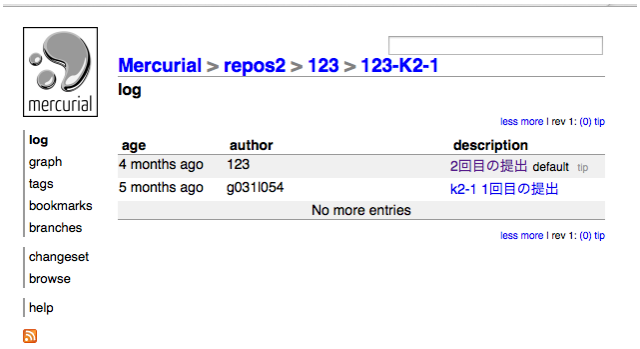

図 6　提出リポジトリ画面
Figure 6　Submitted Repository Screen.

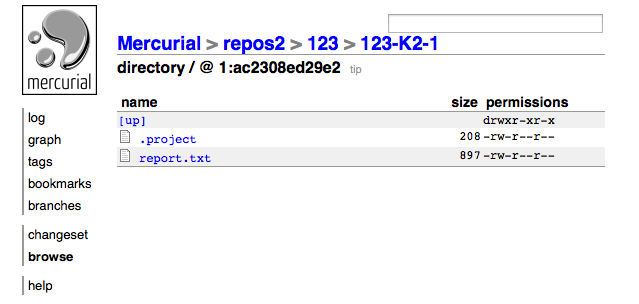

図 7　提出ファイル一覧
Figure 7　Submitted File List.

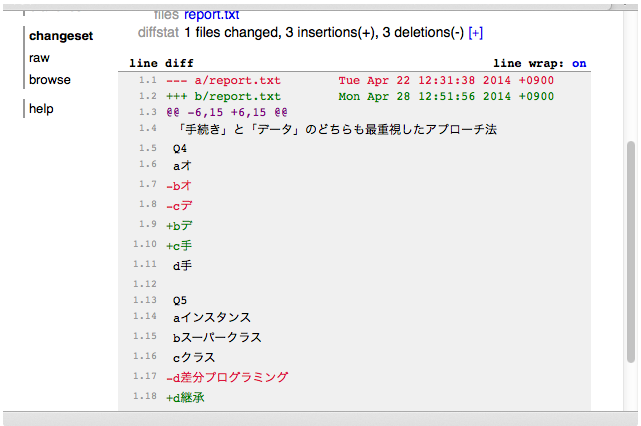

図 8　差分閲覧
Figure 8　Difference Browsing.

（５） 再提出の自動通知

Mercurial には，リポジトリの変更にともなう RSS フィードの機能がある．この機能を利用して，再提出が必要な課題のリポジトリの RSS を登録しておくことにより，再提出がおこなわれるとすぐに通知を受け取ることができ，再採点の見逃しを防ぐことができる．また，RSS には変更のあったリポジトリの URL が含まれるため，再提出者の課題だけを再配布機能を用いて受け取り，即座に実行，採点が可能になる．

4. 運用結果

4.1 構築と運用の容易性について

既存のオープンソースシステムをベースとして設計，構築を行ったことにより，一週間で設計から運用まで行うことが可能であった．

また，実際に 2 年次後期の Java プログラミング演習（180 人（90 人＊２クラス）*15 回*2〜3 課題）において試験的な運用を行ったが，大きな問題は発生しなかった．

4.2 学生にとってのシステム利用

演習の初回に，本システムの利用方法を１時間程度使い説明した．利用法の説明時間としては長めであるが，Eclipse のような統合開発環境や Mercurial のような分散バージョン管理システムは，一般の現場で利用されるものであり，この説明自体にも教育的意義もあると思われる．今回は試験的な運用のため，評価アンケート等は行わなかったが，プログラミングから実行，デバッグ，課題提出までを単一のシステム内で行うことに関して，学生からの不満や戸惑いは特になかった．

問題点としては，一度作成，提出したプロジェクトを誤って削除した後，新たなプロジェクトを作成，再提出しようとすると，システム上別なプロジェクトと見なされ，変更履歴や提出日時がリセットされてしまう問題があった．この修正は今後の課題である．

4.3 教員にとってのシステム利用

システムの全体の機能については特に問題は無かったが，採点中に「前回送ったコメントを見ながら採点できるようにしてほしい」，「リポジトリ画面でも学生名等の情報を表示してほしい」とのページのデザイン上の要望があったので今後改良を加える予定である．

4.4 演習について

Eclipse では，ソースファイルの保存時に自動的にコンパイルが行われ，シンタックスエラー等の静的エラーとその理由をソース行にアイコン表示する．これにより，自分のプログラムが実行不能であることがすぐに分かるため，実行不能なプログラムの提出が行われなくなった．また，似通った変数名，メソッド名等の打間違えに起因する演習時間中の質問がほぼ皆無となり，演習中の TA の負担が大きく軽減された．

4.5 採点について

一括インポート後にボタン一つで実行可能なため，実行結果の採点が非常に容易であった．また，Eclipse のアウトライン表示とエラー／警告機能により，問題のある行，例えば，未使用変数，未使用メソッド，大文字で始まるインスタンス変数, abstract 等のメソッド種別等が色づけされるためソースの採点が容易であった．さらに，Eclipse のプラグインとして提供されている潜在的なバグの発見ツールである，Find Bug[12]，PMD[13]等を使用することにより，到達不能コード，返値の無視，といったロジック上の問題点を容易に採点することが可能であった．

5. おわりに

古典的なプログラミング演習における，課題提出，採点，評価，再提出の一連のサイクルで構成される課題レポート管理の円滑化による支援を行うことにより，教員の負荷の軽減をはかるためのシステムの構築と実践について述べた．また，システム構築，運用の容易性および低コスト化，他システムとの連携性を考慮したオープン性を実現するために，既存のシステムを最大限に活用した構築を実現した．今後，継続した本システムの運用を通して，機能の拡張，利便性の向上をはかっていく予定である．

参考文献

1) java プログラミング演習向け課題レポート提出・管理機能を付加した授業支援システム，熱田智士，松浦佐江子，第 3 回情報科学技術フォーラム一般講演論文集，pp.359-362（2004）
2) 古典的プログラミング演習のスタイルに基づく演習支援システム:角亮，檜垣 泰彦，2009 年電子情報通信学会総合大会講演論文集，pp.214（2009）
3) Java プログラミング学習支援システムの穴埋め問題機能の拡張と授業への適用：中島 秀樹, 高橋 直久, 細川 宜秀，電子情報通信学会論文誌, J88-D-I(2), pp.439-450（2005）
4) 比較的大きなプログラミング課題のための自動採点システム：田上恒大，阿部公輝，情処研報， コンピュータと教育, Vol.2006(16), pp.135-140（2006）
5) Java プログラミング書学者に対するテスト方法学習支援ツール：上河内頌之，松浦 佐江子，信学会技術研究報告.ET-106(364), pp.37-42（2006）
6) 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:内藤広志，斉藤隆，情処研報，コンピュータと教育，Vol.2008(13), pp.33-40（2008）
7) テスト駆動型開発手法による Java プログラミング教育支援システムの提案： 松島由紀子，笠井康裕，舩曳信生，中西透，天野憲樹，信学会技術研究報告. ET-109(82), pp.7-12（2009）
8) プログラミング演習における答案診断のためのプロファイル生成システムの実現：岩間信介，高橋直久，第 6 回情報科学技術フォーラム一般講演論文集，pp.395-396（2007）
9) Mercurial SCM：ttp://mercurial.selenic.com（2014 年 9 月 23 日参照）
10) Eclipse：https://www.eclipse.org/home/index.php（2014 年 9 月 23 日参照）
11) Moodle：https://moodle.org（2014 年 9 月 23 日参照）
12) Find Bugs : http://findbugs.sourceforge.net（2014 年 9 月 23 日参照）
13) PMD：http://pmd.sourceforge.net（2014 年 9 月 23 日参照）