RK-003

# オンラインジャッジシステムと連携可能な Moodle プラグインの実装と比較

## Comparing of programming learning environment using online judge system with moodle

古谷 勇樹† 林 真史† 山本 隆弘† 長尾 和彦†
Yuuki Furutani Masafumi Hayashi Takahiro Yamamoto Kazuhiko Nagao

## 1. はじめに

近年、IT の急速な普及により、IT 技術者の不足が深刻化している[1]。この対策として、日本政府は IT 人材を育成する取り組み[2]を行ってきたが、新 3K 職場など IT 企業に対するブラックなイメージの定着、学生の理系離れ、少子高齢化などの理由から、IT 技術者の人材不足は慢性的なものとなっている。

その一方、IT 技術は社会基盤を支えるものとなっており、IT 技術者の不足は国力低下につながるといえる。このため、世界各国では IT 技術者を育成するため、様々な取り組みが行われている[3]。日本においても、2012 年から中学校でのプログラミング教育の必修化が行われ、他国と並び国力低下を防ぐ取り組みを行っている。

プログラミング教育は理解することが難しく、授業の単位時間のみでの習得は困難である。そのため、プログラミングスキルを習得するには、授業時間外での自学自習が必要不可欠である。多くの大学、専門学校ではプログラム課題を実施し、生徒の自学自習を促している。

しかし、プログラミング課題による学習には 2 つ問題点がある。1 つ目は教員の負担が大きいことである。プログラム課題の採点にはプログラムを実行し、結果を確認する必要がある。そのため、本校を含む多くの学校では教員が手動で課題の採点、成績登録を行っている。

2 つ目は生徒の学習効率が悪いことである。採点は教員が手動で行うため、生徒は教員の採点を待つ必要があるため、学習効率が低下してしまう。

これらの問題の解決策としてプログラムの自動採点を行う Online Judge System(OJS)と web 上で学習管理を行う LMS サーバである moodle[4]を連携した学習環境を利用する。OJS を利用することで、教師の採点作業の効率化、生徒への即時性のあるフィードバックを実現する。また、moodle と連携することで、OJS での採点結果の自動登録、生徒の学習活動の管理を行う。

moodle と連携可能な OJS を調査した結果、3 種類の OJS を見つけることができた。本論文では、これらの実装・試験運用・比較を行い、教員の採点作業の効率化、生徒の学習効率の向上を確認する。

## 2. 本校の学習環境

本校の情報工学科では IT 社会で活躍するための技術者を育成するために、プログラミング学習を行っている。PC 教室で学習を行い、Eclipse を利用し Java をベースに学習している。Java はオブジェクト指向のプログラミング言語であり、開発範囲が広いことから、エンジニアからの支持が高い言語である[5]。

プログラミング学習は 2〜5 年次に行われ、その中でも、2 年次のプログラミング基礎（4 単位）と 3 年次のプログラミング応用（3 単位）でプログラミング教育を重点的に行っている。それぞれの授業内容は以下の通りである。

- プログラミング基礎
  - 演算や制御文、配列などの基礎的なプログラムの作成
  - アプレットを利用した GUI の学習
- プログラミング応用
  - オブジェクト指向のプログラム作成
  - 代表的なアルゴリズムを用いたプログラムの作成
  - XP 手法[6]を用いたプログラム開発（ペアプログラミング、Junit ベースのテストファースト、リファクタリング）

### 2.1 プログラム課題の実施

プログラミング基礎、プログラミング応用では生徒の自学自習を促すためにプログラム課題を実施している。それぞれの内容を以下に示す。

- プログラミング基礎
  - 毎週、課題を実施している
  - 課題の公開はプリント配布で行われ、提出は FTP を利用して行っている
  - 生徒全員分の課題をスクリプトを利用して一斉実行
  - 出力結果を目視で確認し合否判定を行い、結果は Excel に入力し、成績登録を行う
  - 合否結果は翌日、プリント形式で生徒に公開する
- プログラミング応用
  - 毎週、moodle 上にテストファーストを用いた課題を公開している
  - 提出は行わず、授業中に教員が課題の確認し、成績登録を行う
  - moodle 上で課題の内容に関する小テストを実施し、生徒の理解度を把握する
  - 小テストの結果は成績として moodle に登録・公開される

プログラム演習の自動採点を行う取り組み[7][8]はこれまでにも行われているが、独自のシステム環境を利用するため、LMS サーバとの連携ができない。そこで、大学・高専で最も利用されている LMS サーバ moodle とプログラムの自動採点を行う OJS が連携したシステムを利用す

† 弓削商船高等専門学校 National Institute of Technology, Yuge College

ることが望ましい。この場合、以下の要件を満たす必要がある。

- 問題が DB 化され、再利用が可能であること
- 学習履歴が管理できること
- 採点結果が素早く返ってくること
- 生徒、教師が容易に利用できること
- Junit ベースのテストファーストに対応すること
- 安全なシステムであること

## 3. Online Judge System (OJS)

OJS はプログラムの自動採点を行うシステムである。TopCoder[9]や AIZU ONLINE JUDEGE[10]などのプログラミングコンテストで主に利用されている。ユーザは Web ページを介してプログラムの提出を行い、ジャッジサーバでプログラムのコンパイル・実行・採点を行う。採点は解答となる入出力の関係が書かれたテストケースを利用して行う。採点結果は Web サーバへ送られ、Web ページに表示することで、ユーザは採点結果を確認することができる。

OJS の自動採点の仕組みを図 1、OJS の利用画面を図 2 に示す。

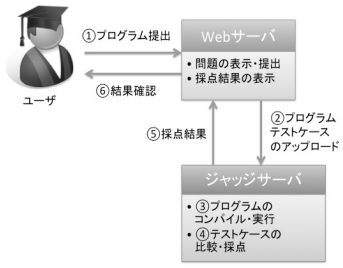


図 1　0JS の自動採点の仕組み


図 2　0JS の利用画面

## 4. LMS サーバ moodle

moodle はオープンソースの LMS サーバである。LMS サーバとは Web ページ上で学習管理を行い、生徒・教師の活動を支援するものである。その中でも moodle は大学・高専で一番多く利用されている LMS サーバである[11]。moodle の主な機能を以下に示す。

- 生徒
  - 学外での学習が可能
  - 個々のペースでの学習が可能
- 教師
  - Web ページでの問題、教材の配布が可能
  - 成績の管理が容易
  - 生徒の学習活動の把握

moodle には多くのプラグインが提供されている。ユーザは学習に合ったプラグインを moodle にインストールすることで、機能の拡張を行い、システムの質を向上することができる。

## 5. OJS と moodle の連携

OJS と moodle の連携を実現するために、OJS に対応した moodle プラグインをインストールする必要がある。これにより得られる機能を以下に示す。

- moodle 上で OJS を用いたプログラム問題の利用（図 3）
- OJS の採点結果の自動登録
- 生徒の学習活動の把握（図 4）

OJS に対応した moodle プラグインの調査を行った結果、以下の 3 つが確認された。

- Online Judge Plugin for Moodle[12] (OJPM)
- upchecker[13]
- CodeRunner[14]

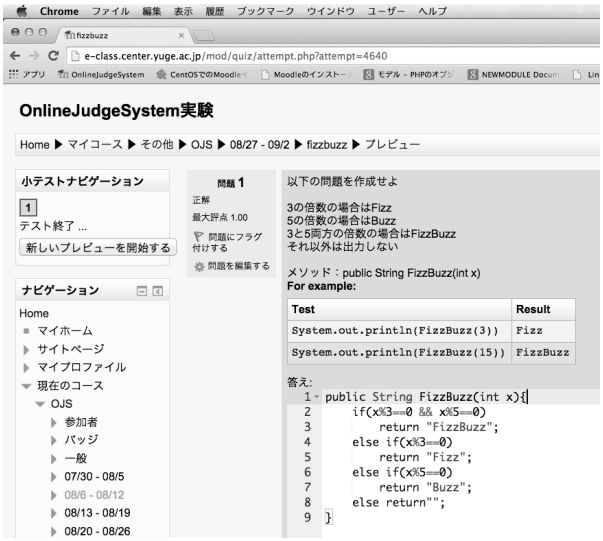


図 3　moodle 上での OJS の利用画面

| | 姓 / 名 | メールアドレス | 状態 | 開始日時 | 受験完了 | 所要時間 | 評点/100.00 |
|---|---|---|---|---|---|---|---|
| | test 1 受験をレビューする | earb@yuge.ac.jp | 終了 | 2014年 10月 3日 05:04 | 2014年 10月 3日 07:30 | 2 時間 25 分 | 100.00 |
| | 商船 太郎 受験をレビューする | eahklgh@yuge.ac.jp | 進行中 | 2014年 10月 3日 06:33 | - | - | - |
| | test 1 受験をレビューする | earb@yuge.ac.jp | 終了 | 2014年 10月 4日 01:09 | 2015年 03月 26日 04:58 | 173 日 3 時間 | 100.00 |
| | test 2 受験をレビューする | epaihoaerih@yuge.ac.jp | 終了 | 2014年 10月 12日 05:38 | 2015年 03月 26日 04:20 | 164 日 22 時間 | 0.00 |

図 4　moodle 評定画面

### 5.1 Online Judge Plugin for Moodle (OJPM)

プログラムファイルの自動採点を行うプラグインである。ジャッジサーバは Ideone[15]というインターネット上に公開されたサーバを利用する。学外のサーバを利用するため、サーバ実装を必要とせず環境構築が容易である。その反面、提出するファイル名の指定など外部サーバによる制約もある。

また、Ideone を利用するためには、Ideone のアカウントを作成する必要がある。アカウント作成は無料で、ユーザ名とメールアドレスを登録するような一般的なものである。作成したアカウントで Ideone にアクセスするこ

とで、自動採点を行う。OJPM のシステム構成図を図 5 に示す。

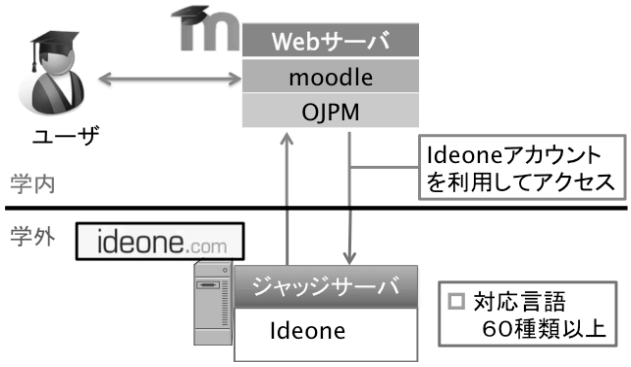


図 5　OJPM のシステム構成図

### 5.2 upchecker

公立はこだて未来大学と株式会社 VERSION2 が共同開発した moodle プラグインである。ジャッジサーバを自作することで、ユーザの要求を満たす自動採点を実現することができる。

本校では、Junit テストファーストを用いた学習を行っていることから、Junit を利用した自動採点を行うジャッジサーバを開発した。問題ファイル、Junit テストケース、Build ファイルの入ったプロジェクトフォルダを提出することで、自動採点を行う。Build ファイルとはプロジェクト内の構成が書かれたファイルであり、フォルダ内にあるファイルの一斉コンパイル、実行を行えるものである。自動採点は Build ファイルを用いて Junit テストケースのコンパイル・実行することで実現する。また、提出する際、プロジェクトフォルダを zip 形式に圧縮する必要がある。

ジャッジサーバの開発はフルスクラッチで行うため、実装には専門的な知識や手間が必要となる。upchecker のシステム構成図を図 6 に示す。

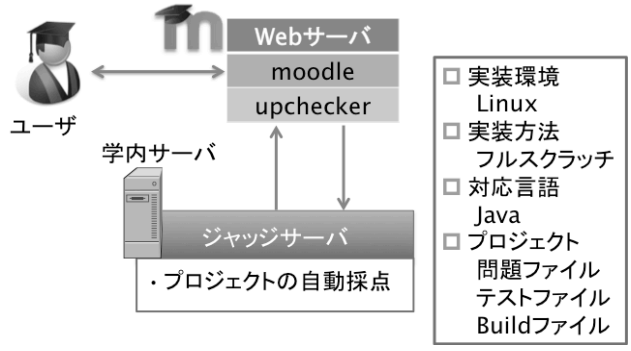


図 6　upchecker のシステム構成図

### 5.3 CodeRunner

カンタベリー大学のリチャード・ロブが開発したプラグイン。Web ページ上のエディタでコーディングを行うことで解答を行う。エディタには行数の表示やシンタックスハイライトなどのコーディング支援機能が備わっている。問題の解答範囲はメソッドとクラスの 2 種類があり、目的に応じて使い分けることが可能である。ジャッジサーバには Jobe[16]を利用する。Jobe の実装は専用のインストーラを利用することで容易に行うことができる。

また、ジャッジサーバの設定を変更することで、Junit に対応した解答が可能である。CodeRunner のシステム構成図を図 7 に示す。

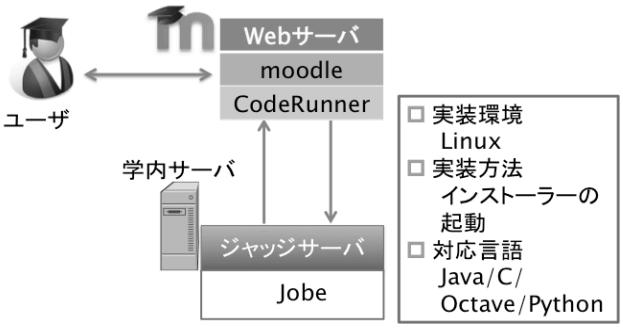


図 7　CodeRunner のシステム構成図

### 5.4 各 OJS の特徴

各 OJS の特徴を表 1 に示す。セキュリティの項目はプログラムの実行を sandbox 内で行っているか表したものである。sandbox は外部から受け取ったプログラムを保護された領域で動作させるセキュリティモデルである。実行したプログラムが暴走したりウイルスを動作させようとしても sandbox 外のシステムに影響を与えない仕組みにすることで、システムの安全性を保証している。

表 1　各 OJS の特徴

| | OJPM | upchecker | CodeRunner |
|---|---|---|---|
| 解答方法 | ファイル提出 | ファイル提出 | コーディング |
| 問題範囲 | クラス | クラス | クラス<br>メソッド |
| プログラミング言語 | Java含む 60 言語 | Java | Java<br>C<br>Octave<br>Python |
| テストファースト(Junit 互換) | × | ○ | △ |
| セキュリティ | ○ | × | ○ |
| ジャッジサーバ | Ideone | 自作サーバ | Jobe |
| サーバの用意 | × | ○ | ○ |
| 実装方法または利用方法 | Ideone アカウントの作成 | フルスクラッチ | インストーラーを利用 |
| 実装のコスト | ◎ | × | ○ |

## 6. システムの試験運用

OJS と LMS を連携した学習環境が教員の採点作業の効率化、生徒の学習効率の向上が可能か確認するために各 OJS の試験運用を行った。

試験運用として問題作成、授業での実施を行う。問題作成は各 OJS、授業での実施は CodeRunner のみで行った。授業での実施は問題の解答、提出、結果確認を行う。また、利用した生徒から採点時間に関するアンケートを実施することで、即時性のあるフィードバックができているか確認する。授業での実施環境を以下に示す。
対象：情報工学科 3 年生（38 人）
試験時間：1 時間、（これ以外に冬期休業中課題）
問題数：4 問（授業で実施される程度の難易度）

### 6.1 問題作成

問題作成は moodle 上で行う。moodle では問題の実施、教材の公開などの学習活動を行うためにモジュールと呼ばれるコンテンツを追加する必要がある。OJPM は課題モジュールを利用し、upchecker と CodeRunner は小テストモジュールを利用して問題を公開する。それぞれの違いとして、課題モジュールはファイルなどの提出物を提出させる課題を実施するものであり、小テストモジュールは穴埋め問題や計算問題といった問題をテスト形式で公開するものである。小テストモジュールで利用する問題は問題バンクで管理しており、問題の再利用、他の moodle サーバの問題をインポートすることが可能である。

また、プログラムの自動採点を行うために各プラグインには独自の設定項目があるため、問題作成を行う手間に差がでる。OJPM は使用する言語の他に CPU 使用時間やメモリ制限などの細かい設定、Ideone にアクセスするための ID とパスワードの入力と解答の正誤判定をするためのテストケースの作成が必要である。upchecker では利用するジャッジサーバの URL と POST 変数名、採点形式の指定を行うだけの設定となっている。CodeRunner では使用する言語の設定と問題範囲の設定、テストケースの作成を行う必要がある。

各 OJS の問題作成の手順を表 2、OJPM のオンラインジャッジの設定画面を図 8、upchecker のジャッジサーバの設定画面を図 9、CodeRnner のテストケース作成画面を図 10 に示す。

表 2　問題作成の手順

| 手順 | OJPM | upchecker | CodeRunner |
|---|---|---|---|
| 1 | 課題モジュールを追加 | 小テストモジュールを追加 | |
| 2 | 課題の設定 | 小テストの設定 | |
| 3 | オンラインジャッジの設定（図 8） | 小テストで利用する問題の選択 upchecker | 小テストで利用する問題の選択 CodeRunner |
| 4 | テストケース作成 | 問題内容の設定 | 問題内容の設定 |
| 5 | 問題作成 | ジャッジサーバの指定<br>採点形式の設定（図 9） | テストケース作成（図 10） |
| 6 | | 問題作成 | 問題作成 |


図 8　オンラインジャッジの設定画面(OJPM)


図 9　ジャッジサーバの設定画面（upchecker）

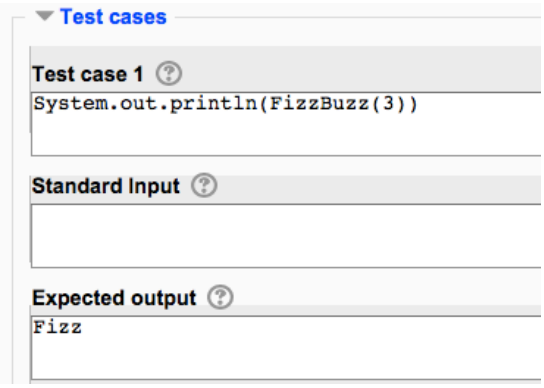


図 10　テストケース作成画面（CodeRunner）

### 6.2 解答提出

学習者は 6.1 章で作成した問題を受講し、プログラムを作成する。解答方法は各 OJS で異なる。OJPM と upchecker はファイルの提出、CodeRunner は Web ページ上でコーディングすることで解答を行う。提出後、テストケースの結果と採点結果が表示される。解答画面を図 11、採点結果画面を図 12 に示す。

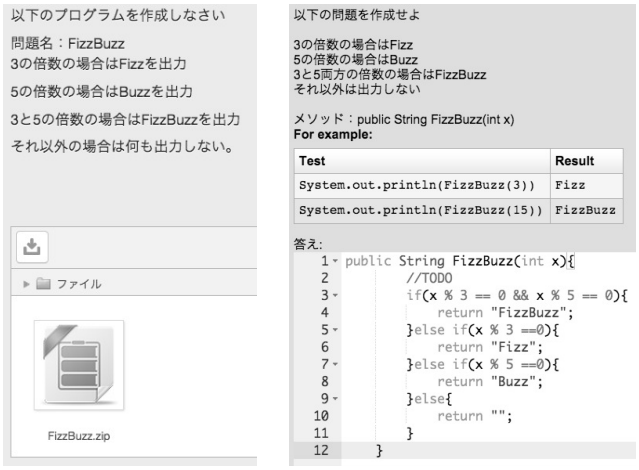


図 11　解答画面（左：upchecker　右：CodeRunner）

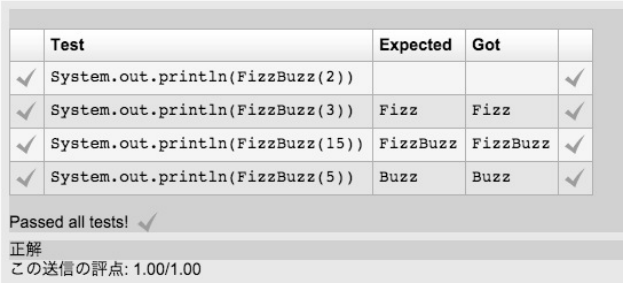


図 12　採点結果画面（CodeRunner）

### 6.3 試験運用の結果

アンケートの結果を図 13 に示す。ほぼ全ての生徒が 1 分以内に採点結果が返ってくると答えた。このことから、生徒に即時性のあるフィードバックをすることができ、学習効率の向上が期待できる。また、教員は問題作成時、OJS の設定をする必要があるが、課題を採点する必要がなくなり、作業の効率化ができた。

作成した教員からの意見を示す。

- Eclipse のプロジェクトをそのまま利用できる upchecker が機能的に好ましい
- 他の形式についても、スクリプト等を用いることで変換できるため、それほどの負担ではない


図 13　採点時間に関するアンケート結果

## 7. 各 OJS の比較

各 OJS のユーザインターフェースの評価、採点時間の比較を行う。ユーザインターフェースの評価として、教師が行う問題作成、生徒が行う解答提出の処理コストを測定する。処理コストはクリックの回数、キー入力の箇所、ページ遷移の回数とした。問題作成、解答提出の評価の際、利用する問題は実際に授業で利用されているものを用いた。利用した問題を図 14 に示す。

問題名：FizzBuzz
メソッド FizzBuzz の引数の値が
- 3 の倍数の場合は Fizz
- 5 の倍数の場合は Buzz
- 3 と 5 両方の倍数の場合には FizzBuzz
- それ以外には何も出力しない

テストケース数：4 つ

図 14　問題内容

### 7.1 ユーザインターフェースの比較

問題作成の処理コスト、解答提出の処理コストを測定した。それぞれの測定結果を表 3、表 4 に示す。

表 3　問題作成の処理コスト

| | OJPM | upchecker | CodeRunner |
|---|---|---|---|
| クリック数 | 20+n | 13 | 14 |
| キー入力 | 4+2n | 9 | 6+2n |
| ページ遷移数 | 6 | 8 | 8 |

n=テストケースの数

表 4　解答提出の処理コスト

| | OJPM | upchecker | CodeRunner |
|---|---|---|---|
| クリック数 | 7 | 11 | 4 |
| キー入力 | ファイル名変更 (1 カ所) | パス入力 (1 カ所) | コピー& ペースト (1 回) |
| ページ遷移数 | 4 | 6 | 3 |

表 3 の結果から upchecker 以外の OJS はテストケースの数に比例して問題作成の処理コストが増加した。upchecker は提出するプロジェクトフォルダ内のテストファイルを利用して採点するため、問題作成時にテストケースを作成しないためである。

表 4 の結果では CodeRunner が最も容易に解答提出が可能だということが分かった。OJPM はクリック回数、ページ遷移回数が少ないが、クラス名を変更する必要がある。また、upchecker は他と比べおよそ 2 倍の処理コストがかかった。upchecker は解答提出の際にプロジェクトフォルダを zip ファイル形式でエクスポートする必要があり、この作業が処理コストが多い原因となっている。

### 7.2 採点時間の比較

課題を提出してから結果が返ってくるまでの時間の計測を行った。採点時間の測定結果を図 15 に示す。

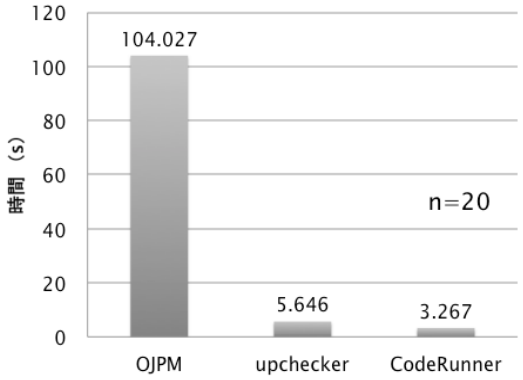


図 15　採点時間

図 15 の結果から upcheker と CodeRunner は数秒で程度で採点結果を返すのに対して、OJPM は 100 秒以上時間がかかった。この原因として、Ideone の利用が考えられる。Ideone は世界中のユーザが利用しており、ネットワークやサーバの負荷に影響を受けるためである。

## 8. OJS の要件対応状況

各 OJS が 2 章で述べた要件をどれだけ満たしているか確認する。各 OJS と要件の比較を表 5 に示す。

表 5　各 OJS と要件の比較

| | 要件 | OJPM | upchecker | CodeRunner |
|---|---|---|---|---|
| 採点時間 | ○ | × | ○ | ○ |
| 問題作成の処理コスト | ○ | △ | ○ | △ |
| 解答提出の処理コスト | ○ | ○ | × | ◎ |
| テストファースト | ○ | × | ○ | △ |
| セキュリティ | ○ | ○ | × | ○ |

表 5 の結果から、CodeRunner が最も要件を満たしていることが確認された。

## 9. まとめ

プログラム課題の問題であった、教員の採点作業の手間、生徒の学習効率の低下を改善するために OJS と moodle を連携した学習環境を実現した。

moodle と連携可能な 3 種類の OJS の実装・試験運用・比較を行い、試験運用の結果から教員の採点作業の効率化、生徒の学習効率の向上が確認できた。また、比較結果から、本校が求める要件を満たした OJS が CodeRunner であることが分かった。冬期休業中の課題として、CodeRunner による課題を課したが、多くの学生が自主的にプログラミングをこなしている。

今後の課題として、CodeRunner をベースに問題の蓄積を行い、本校の授業で利用し、学習効果の測定を行う予定である。

## 参考文献

[1] Business Journal：IT 技術者不足, http://biz-journal.jp/2014/06/post_5239.html
[2] 文部科学省高等教育局専門教育課, “文部科学省における情報技術者育成の取組および今後の施策について” (2012).
[3] Paiza 開発日記：プログラミング教育を強化した国で何が起きているのか？世界の教育事情, http://paiza.hatenablog.com/entry/2015/02/24/プログラミング教育を強化した国で何が起きてい
[4] moodle：https://moodle.org/
[5] paiza 開発日誌：【Java が恐ろしく強い】転職時に希望するプログラミング言語ランキング http://paiza.hatenablog.com/
[6] Kent Beck, “XP エクストリーム・プログラミング入門―ソフトウェア開発の究極の手法” (2000).
[7] 平岡利規, 森山真光, “プログラミングの学習を支援する Web サービスアプリケーション -単体テストを用いたプログラムの自動採点サービス-“ 電子情報通信学会 2014 年総合大会 (2014)
[8] 稗方和夫, “プログラミング学習支援システム JavaEP の開発と導入”, SIG-KST-2011-02-03 (2011)
[9] Top Coder：http://www.topcoder.com/
[10] AIZU ONLINE JUDEGE：http://judge.u-aizu.ac.jp/onlinejudge/
[11] 文部科学省先導的大学改革推進委託事業, “ICT 活用教育の推進に関する調査研究” (2011).
[12] Online Judge Plugin for Moodle：https://github.com/hit-moodle/onlinejudge/wiki
[13] 伊藤恵, 美馬義亮, 大西昭夫, “コース管理システムと授業固有の課題チェック機能のwebサービスによる連携”, 情報処理学会論文誌, Vol.52, No.12 (2011).
[14] CodeRunner：https://github.com/trampgeek/CodeRunner
[15] Ideone：https://ideone.com/
[16] Jobe：https://github.com/trampgeek/jobe