卒　業　研　究　報　告　書

平成28年度

研究題目

Online Judge Systemを用いた
プログラミング講義の支援

指導教員　　上　野　秀　剛　　講師

氏　　名　　大　野　　優

平成29年2月16日　提出

奈良工業高等専門学校　　情報工学科

Online Judge Systemを用いた プログラミング講義の支援

上野研究室　大野　優

プログラミング講義の担当教員は，講義の演習課題を採点する際に手動でプログラムを実行し，その結果を目視で確認する必要がある．そのため，プログラミング講義の担当教員の負担が大きいことが問題である．本問題の解決策として，Online Judge System(OJS)の自動採点機能に焦点を当てた．OJSは，テストケースを用意することでプログラムの自動採点が可能となるシステムである．OJSによる採点の方法は従来の手動による採点方法に比べて採点作業の所要時間を短縮できることが考えられる．

そこで本研究では，OJSを実際にプログラミング講義に導入することによって，教員の負担が軽減できるかどうかを検証することを目的とする．

実験では，被験者（教員）にプログラミング講義に必要な作業に要する時間を，「OJSを導入しない従来の講義方法」と「OJSを導入した講義方法」の2つの方法で計測してもらう.「OJSを導入しない従来の講義方法」の作業時間は演習課題・テストケースの作成に要する作業時間とし,「OJSを導入した講義方法」の作業時間は提出チェックとソースコードの実行・結果確認による採点作業時間の合計とする．

実験の結果，OJSを導入した講義方法で要する作業時間の方がOJSを導入しない従来の講義方法に比べて長くなったが，長期的視点で見た場合，OJSのスケーラビリティ，採点の正確性の保証と手動による採点方法で発生するヒューマンエラーを考慮すると，OJSの講義への導入は有用であると考えられる．

また，OJSの講義への導入が受講者にとって有用であるかどうかを調査するために，講義アンケートを実施した．調査項目は，OJSの使いやすさ，OJSの即時フィードバック，OJSによる講義の総合評価の3点である．アンケートの結果，3点全ての項目において，いずれも高評価であり，受講者にとってOJSがプログラミング講義に有用であることが示唆された．

目 次

1 序論

日本国内では，将来のIT技術者の養成・確保および論理的思考力や問題解決能力の向上を目的に,「義務教育段階からのプログラミング教育等のIT教育を推進する」として，2020年度までに，小・中学校でのプログラミング教育を必須化するという内容が政府の成長戦略として挙げられている[1], [2]. このように現在，プログラミング教育は重要となっている.

現在，プログラミング教育における課題点として，プログラミング講義の担当教員の負担が大きいという点が挙げられる. なぜなら，講義の演習課題の採点は，手動でプログラムを実行し，その結果を目視で確認する必要があるからである. 講義1回分あたりに必要な採点の数は,「受講者数」×「演習課題数」である. したがって，受講者数や演習課題数が増えるにつれて，担当教員の負担も大きくなってしまう.

本研究では，プログラミングコンテストやAIZU ONLINE JUDGE[3], paiza[4] 等のプログラミングの自学自習Webサイトで利用されている，Online Judge System（以下，OJSとする）による自動採点機能に焦点を当てる. OJSは，テストケースを用意するだけで，解答のプログラムの自動採点が可能となる. ここで，テストケースとは，作成したプログラムが要件を満たしているかテストするための入力と，入力に対して望まれる出力の組みである. したがって，OJSは受講者数や演習課題数が増えても，与えられたテストケースの正解数によって，プログラムの正誤判定を全て自動で判断することが出来るスケーラブルなシステムである.

本研究では，OJSを従来使われていたプログラミングコンテストや自学自習の場面ではなく，プログラミング講義に導入することによって，教員の負担が軽減できるかどうかを検証することを目的とする.

本論文の構成は以下の通りである. 2章では関連研究について述べる. 3章ではOJSの概要を述べる. 4章ではプログラミング講義における作業内容について述べる. 5章では実験概要・計測手順を示し，6章で実験結果を述べ，考察を行う. 7章では結論として，まとめと今後の課題について述べる.

2 関連研究

これまでに，OJSに関する研究は，技術要件からプログラミング講義の支援等，様々な方面で複数行われている[5]~[7].

古谷らは，学習管理を行うWebサービス「moodle」と連携可能な3種類のOJSのプラグイン(Online Judge Plugin for Moodle, upchecker, CodeRunner)について，プログラムの採点に要する時間・UIの比較等の5つの要件で比較を行った[5]. 比較の結果，CodeRunnerが最も要件を満たしていることが確認された.

Huiらは，大学独自のOJSを導入した講義用システムである，YOJ(Youxue Online Judge)を構築した[6]. YOJは，解答ソースコードのコンパイル・実行・正誤判定を行うOnline Judgeモジュール等，7つのモジュールで構成されている. 各々のモジュールを拡張することによって，並列処理等による実行・評価を行うことができる.

岩本らは，(1)コーディングスタイルチェッカー，(2)不正コピー検出の2つの機能を備えたOJSを構築した[7]. 本OJSをプログラミング講義に導入し，講義の受講者125名にアンケート調査を実施した. アンケートの結果，機能(1)については，96%の学生が「評価する」と回答し，機能(2)については，80%の学生が「評価する」と回答した.

上記の研究は，OJSを利用しているという点において本研究と関連がある. しかし，OJSをプログラミング講義へ導入することによる，教員の負担軽減に関する研究は行われていない. そこで，本研究では被験者（教員）にプログラミング講義に必要な作業に要する時間を計測する. 計測の結果から，OJSをプログラミング講義へ導入することによって，教員の負担がどれだけ軽減できるかどうかを明らかにする.

本研究から，プログラミング講義へのOJSの導入による担当教員の負担軽減が明らかになれば，教員は少ない負担でプログラミング講義を実施することができる. 将来的には大学の講義だけでなく，小・中学校，高等学校でのプログラミング教育を行う際にもOJSの導入により教員の負担軽減が可能であるといえる.

3 Online Judge System(OJS)

3.1 概要

Online Judge System(OJS) は，提出されたプログラムの正確性とその効率の自動判定を行うWebサービスである．OJSのユーザは，24時間インターネットからアクセスして，自分のペースで問題を解くことが出来る[8]. 国内では主に，ACM-ICPCのプログラミングコンテスト，会津大学のAIZU ONLINE JUDGE，自学自習サイトpaiza等が知られている．また，OJSで取り扱うプログラミング言語として，C言語やJava，Python，Ruby，PHP等が挙げられる．

3.2 動作原理

OJSの動作原理を図1に示す. OJSはユーザからのHTTPリクエストを処理してWebページを表示するWebサーバと，回答のソースコードの正誤を判定するJudgeサーバの2つで成り立っている[9]. 図1中の手順1~7の詳細を以下に示す.

1. 教員はあらかじめ演習問題と，演習問題の正誤判定に必要なテストケースをWebサーバにアップロードする．
2. ユーザはWebサーバに掲載されている問題を解き，回答のソースコードを提出する．
3. Webサーバは送信されたソースコードと，テストケースをJudgeサーバへ転送する．
4. Judgeサーバはソースコードをコンパイルする．
5. 送信されたソースコードを実行し，問題の要件を満たす出力を行なっているかテストケースを用いて確認する．
6. Judgeサーバは回答のソースコードの判定結果をWebサーバに返却する．判定結果はテストケースの正誤判定に加えて，コンパイル時のエラー，実行時エラー等も結果として報告される．
7. WebサーバはJudgeサーバから受け取った判定結果を掲載し，ユーザへ通知する．

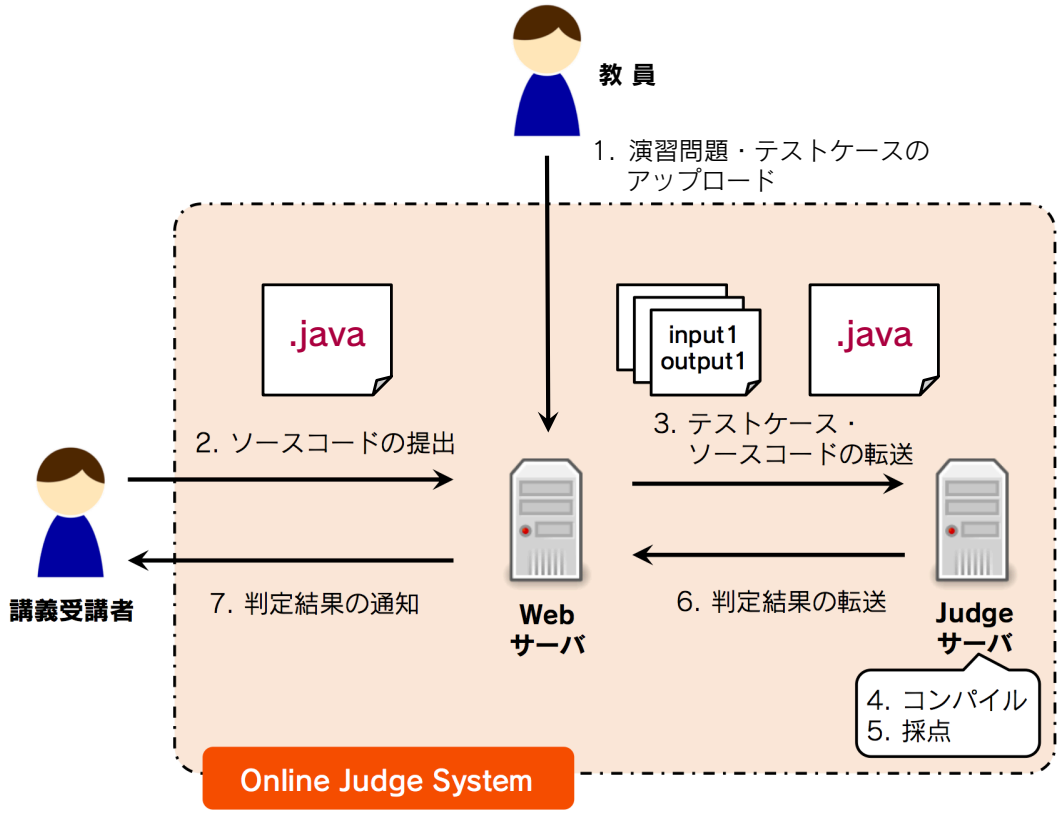

図 1 OJS の動作原理

3.3 OJS の効用

OJS の効用としてスケーラビリティの担保，ヒューマンエラーの防止，主観による評価基準の曖昧性排除の 3 点が挙げられる．

3.3.1 スケーラビリティの担保

OJS は採点すべきプログラムの数が増えても，採点に要する作業時間が増えることはない．なぜなら，OJS は採点すべきプログラムの数が増えても，テストケースに対して正解した数をもとに自動ですべてのプログラムを採点するからである．受講者数や講義 1 回分の演習課題数が増えることで採点すべきプログラムの数が増えたとしても，プログラムはすべて OJS によって採点される．したがって，採点に要する作業時間が増えることはなく，教員の負担増加を防ぐことができる．

3.3.2 ヒューマンエラーの防止

OJS を用いない従来の採点方法は，プログラムを手動で実行して結果を目視で確認するが，確認漏れによるヒューマンエラーが発生する恐れがある．ヒューマ

ンエラーを防ぐためには採点後の再確認に代表される追加の作業が必要なため，教員の負担が増加する．

OJSによる採点の場合，テストケースに基づいて自動で採点されるため，ヒューマンエラーを発生させることなくプログラムを採点できる．

3.3.3 評価基準の曖昧性排除

従来の採点方法の場合，教員による評定のばらつきに起因する曖昧性が発生することがある．評価基準の曖昧性の原因として，大量のプログラムを評価していくうちに，評定者の評価基準が不安定になることが挙げられる[10].

曖昧な評価基準は不公平な採点につながるため，評価基準は恒常的かつ安定したものである必要がある．

4 プログラミング講義

本章では，奈良高専情報工学科で行われている従来のプログラミング講義の形態や流れ，および2016年度のプログラミングIIの後期にて導入するOJSである，Sharif Judgeについて述べる．

4.1 奈良高専情報工学科でのプログラミングIIの講義

本節では，奈良高専情報工学科で行われている従来のプログラミング講義について述べる．

本学科では，2~4年次にプログラミング講義が必須科目として各々用意されている．本研究で対象とする講義は，3年次に行うプログラミングIIである．本講義では，前期･後期でテーマが分かれており，前期ではオブジェクト指向の基礎に関する内容，後期では前期の応用および実用的なソフトウェアの設計について講義を行う．本研究で対象とする講義は後期分とし，担当教員は上野教員である．

本学科におけるプログラミング講義では，1人1台のパソコンが用意されているコンピュータ演習室で行われる．また，講義資料のダウンロードや演習課題の提出等に用いるLMS(Learning Management System)として，学内のみアクセス可能であるe-Learningシステムを利用する．

プログラミング講義1回分の流れとして，大きく3つに分けることができ,「講義前の準備」,「講義の実施」,「講義後の演習状況の確認」である．各々の詳細を以下に示す.

1. 講義前の準備
   教員は，講義の準備として，講義に必要となる資料及び演習課題の作成を行う．作成した後，LMSのWebサーバへアップロードする．

2. 講義の実施
   講義はアップロードされた講義資料を用いて行われる．講義の最初に資料に基づいた教員の説明があり，その後，説明内容に関連した複数の演習が受講者に課される．受講者は各自のパソコンにインストールされた統合開発環境であるeclipse[11]を使用して演習を行う．演習の課題は次回の講義までにe-Learningシステムに提出する必要があり，作成したソースコードはzip形式に圧縮して提出する．

3. 演習状況の確認
   教員は，演習課題の提出締切日が経過した時点で，演習課題の採点を行う．また，演習状況の結果を講義の成績へ反映させる．

4.2 Sharif Judge

Sharif Judge[12] は，PHP と bash で構築されたオープンソースの OJS であり，プログラムの採点に対応している言語は，C 言語，C++，Java，Python である．また，本研究で使用する Sharif Judge は，日本語化や複数のソースコードを提出できるように拡張されている．

本研究では，Sharif Judge を 2016 年度のプログラミング II の後期にて導入する．

5 実験

5.1 教員の負担軽減

5.1.1 実験概要

本実験では，プログラミング講義に要する作業の所要時間を計測する．実験の対象とする講義は情報工学科3年次に行うプログラミングIIで，担当教員は上野教員である．また，導入するOJSは4.2節で述べたSharif Judgeである．実験期間は，2016年度の後期とし，この期間に実施された2回分の講義（11月8日，11月15日講義分）を計測の対象とする．

被験者は，本講義の担当教員1名で，被験者が所有しているPCを用いて計測を行う．

5.1.2 作業内容

本実験では，4.1節のプログラミング講義の流れをもとに，講義に必要な作業時間を，「OJSを導入しない従来の講義方法」と「OJSを導入した講義方法」の2つの方法で計測する．しかし，「講義の実施」はどちらの方法でも，所要時間は変わらない為，計測の対象外とする．したがって，本研究では「講義前の準備」および「講義後の演習状況の確認」の2つの行程を計測の対象とする．以下，5.1.2.1項，5.1.2.2項において，「OJSを導入しない従来の講義方法」と「OJSを導入した講義方法」の各々の作業内容について述べる．

5.1.2.1 OJSを導入しない従来の講義方法

OJSを導入しない従来の講義方法の流れを図2に示す．OJSを導入しない従来の講義方法において，講義の前後で行うべき作業内容は，全部で5点ある．図2中の作業1~5の内容を各々以下に示す．

1. 講義資料のアップロード

   講義の準備として，講義に必要となる資料及び演習課題の作成を行い，LMSのWebサーバへアップロードする．

2. 提出チェック

   LMSにある課題の提出状況（未提出・提出済）を目視して，受講者が演習課題を提出したかどうかを確認する．

3. ソースコードのオープン

   ソースコードをeclipse上に開く．この時，ファイル・クラス名に誤りがあったり，文字化けが発生した場合，適宜修正を行う．

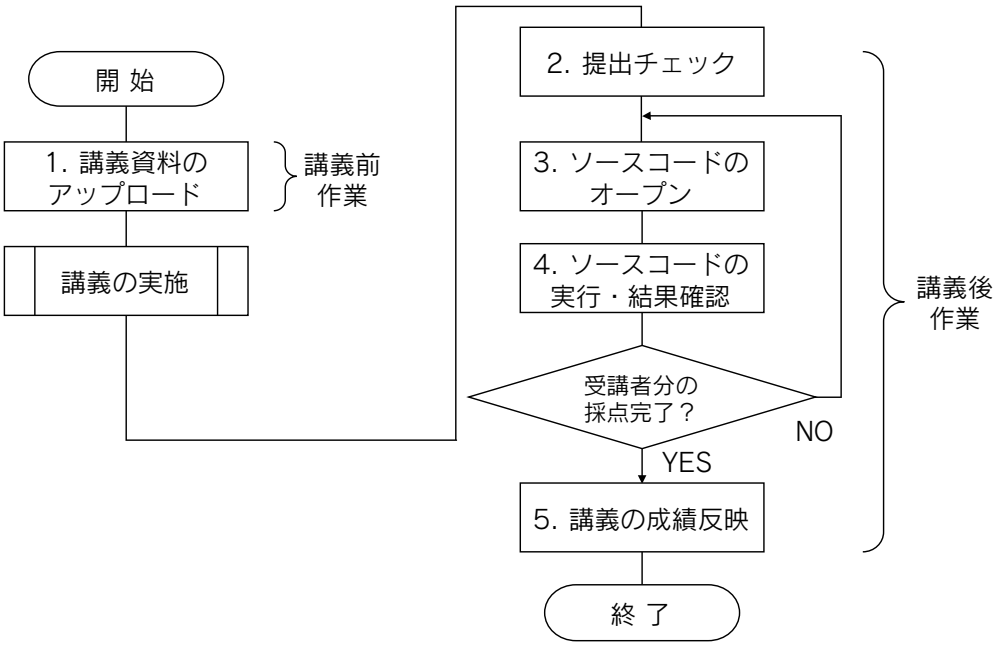

図 2 OJS を導入しない従来の講義方法の流れ

4. ソースコードの実行・結果確認

提出されたソースコードのアルゴリズムが正しいかどうか確認を行った後，実際にソースコードを実行し，ある入力に対して，正しい結果が出力されているかどうかを目視によって確認する．確認の結果に応じて，各受講者に対する演習課題の成績表に課題の成績を入力する．

5. 講義成績への反映

講義受講者数分の採点（作業2~4）後，演習課題の成績表に記録されている課題の成績を講義の成績表にコピー＆ペーストする．

5.1.2.2 OJS を導入した講義方法

OJS を導入した講義方法の流れを図 3 に示す．OJS を導入した講義方法において，講義の前後で行うべき作業内容は，全部で 3 点ある．図 3 中の作業 1~3 の内容を各々以下に示す．

1. 演習課題の作成

演習課題の作成では，文字コードによるトラブルの回避が可能になるように仕様を決定する必要がある．仕様の例として，コメントを除く日本語の部分を英語化する方法が挙げられる．演習課題作成後，OJS の Web サーバへアップロードする．

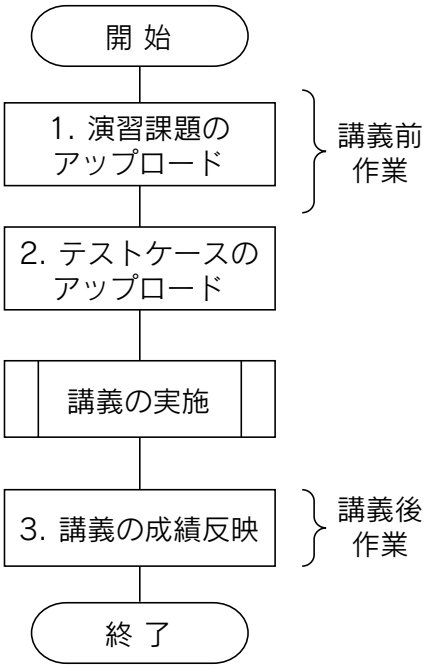

図 3 OJS を導入した講義方法の流れ

2. テストケースの作成

テストケースは，入力用と出力用のテキストファイルの組みによって構成されている．

テストケースの作成手順として，プログラムの入力値を個数分，1つずつ改行しながら入力用のテキストファイルに記し，同様にプログラムの出力値を個数分，1つずつ改行しながら出力用のテキストファイルに記す．テストケースの作成例を図4に示す．図は，与えられた2つの整数の和と積を表示させるプログラムのテストケースの作成例である．

図4に示すテストケースの場合，入力用のテキストファイルに，入力となる2つの整数値である8と9，出力用テキストファイルに入力に対応する出力である，8と9の和(17)と積(72)を記せばよい．

以上の手順で作成したファイルを複数個作成することによって，テストケースが完成する．

また，OJSによる正誤判定で質の良い判定を行う点で重要となるのは，適切なテストケースの入力値を用意することである．適切なテストケースの特徴として，以下の2点が挙げられる．

- パラメータの境界条件

  プログラムの仕様における様々なパラメータの境界値をテストケースに含めること．

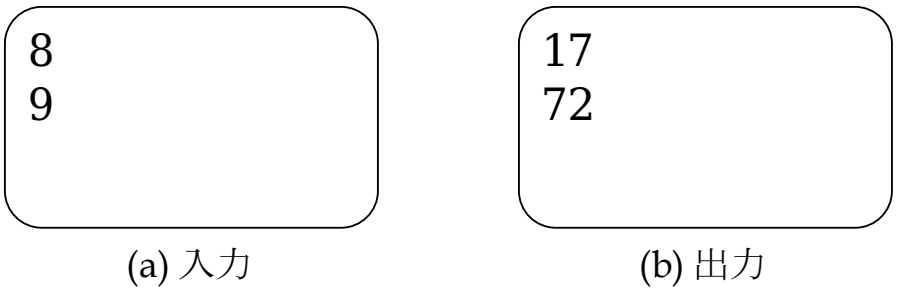

図4テストケースの作成例

- 不具合検出
  不正な値が入力された場合に対して，不具合（エラー）が正しく起こること．

教員は，以上のことを留意した上で，課題の仕様におけるパラメータとその境界条件，および不正な値が入力された場合に対して起こる不具合を考えてからテストケースを作成する必要がある．

3. 講義への成績反映
   Sharif JudgeのWebサーバから，Excel形式の演習課題の成績表をダウンロードする．ダウンロードされた成績表には，提出者の氏名，提出日時，課題の成績が記録されている．本作業では，演習課題の成績表に記録されている課題の成績を講義の成績表にコピー＆ペーストする．

5.1.2.3 計測の対象とする作業内容

本実験で計測する作業内容は以下の3つである．

1. 演習課題・テストケースの作成時間(*make*)
2. 提出チェック時間(*check*)
3. ソースコードの動作確認・成績反映時間(*execution*)

また，以上の3つの作業について，各々の作業の開始時刻・終了時間を変数として表1に定義する．

表1作業開始時刻・終了時刻

	作業開始時刻	作業終了時刻
演習課題・テストケースの作成	$make_{start}$	$make_{finish}$
提出チェック	$check_{start}$	$check_{finish}$
ソースコードの動作確認・成績反映	$execution_{start}$	$execution_{finish}$

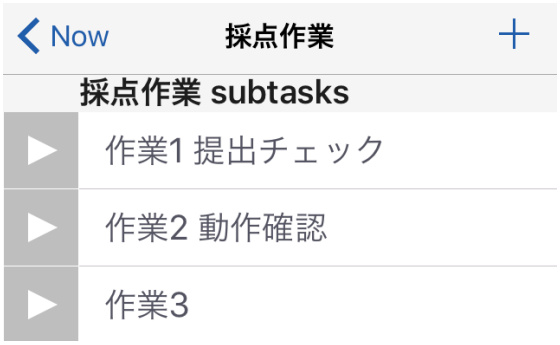

図 5 Now Then Pro のスクリーンショット

```
ブラウザ= iexplore.exe|firefox.exe|Safari.exe|chrome.exe
プログラミング= eclipse.exe|netbeans.exe
テキスト閲覧・編集= EmEditor.exe|TeraPad.exe|notepad.exe|
WINWORD.EXE|Acrobat.exe|acrodist.exe|Kami.exe
プレゼン編集= POWERPNT.EXE
データ分析= EXCEL.EXE|Rgui.exe|paswstat.exe|javaw.exe:Weka|
ファイル操作= explorer.exe
ネットワーク= FFFTP.exe|WinSCP.exe|Skype.exe
マルチメディア= PStarter.exe|mspaint.exe|OIS.EXE|wmplayer.exe|
dllhost.exe:Windows
```

図 6 TaskPit の設定ファイル

5.1.3 計測に用いるソフトウェア

本実験の計測に用いるソフトウェアは，Now Then Pro[13] と TaskPitciteTaskPit の 2 つである．

Now Then Pro は，Angry Aztec 社が作成した iOS 対応の作業計測用のソフトウェアである．Now Then Pro のスクリーンショットを図5に示す．あらかじめ計測する作業の名称をリストに追加し，リストから作業名が書かれたボタンをタップして計測する作業を切り替える．計測した作業時間の履歴は csv ファイル形式で出力され，行った作業の作業名および開始時間･終了時間が記録される．本実験では，Now Then Pro がインストールされている iPad を用いる．

TaskPit は，Windows 対応の作業計測･判別を自動化したソフトウェアである．あらかじめ図6に示す設定ファイルにソフトウェアの実行ファイル名を記入することで，作業時間を計測できる．TaskPit は，アクティブになっているソフトウェアのウィンドウ名と設定ファイルに記入されている作業を関連づけて，作業時間や打鍵数，マウスクリック数を記録する．計測した記録は csv ファイル形式で出力され，作業名，ソフトウェアのウィンドウ名，作業時間や打鍵数，マウスクリック数が記録される．

5.1.4 計測手順

計測の手順を以下に示す.

1. 計測の準備
   計測前に，実験者はあらかじめNow Then Proに「演習課題・テストケースの作成」,「提出チェック」,「ソースコードの動作確認・成績反映」の3つが書かれた作業リストを作成し，被験者へiPadを渡す. また，TaskPitを起動する.

2. 「演習課題・テストケースの作成」の作業時間の計測
   被験者はiPadを被験者のPCの近くに置き，Now Then Proを起動する. 被験者はNow Then Proの作業リストにある「演習課題・テストケースの作成」をタップしてから作業を開始する($make_{start}$). 演習課題・テストケースをSharif JudgeのWebサーバへアップロードした時点で，Now Then Proの作業リストにある「演習課題・テストケースの作成」をもう1度タップして計測を終了する($make_{finish}$).

3. 「提出チェック」・「ソースコードの動作確認・成績反映」の作業の計測準備
   本計測において採点作業に用いるソースコードはSharif JudgeのWebサーバに提出されているソースコードを用いる. 計測前に被験者は，Sharif JudgeのWebサーバに提出されているソースコードをダウンロードする.

4. 「提出チェック」の作業時間の計測
   被験者はNow Then Proの作業リストにある「提出チェック」をタップしてから作業を開始する($check_{start}$).「提出チェック」作業が終了した時点で，被験者はNow Then Proの作業リストにある「提出チェック」をタップしてから作業を終了する($check_{finish}$).

5. 「ソースコードの動作確認・成績反映」の作業時間の計測
   被験者はNow Then Proの作業リストにある「ソースコードの動作確認・成績反映」をタップしてから作業を開始する($execution_{start}$). 受講者数分のソースコードの動作確認が終了し，演習課題の成績を講義の成績表へ反映させた時点で，被験者は「ソースコードの動作確認・成績反映」をもう1度タップして作業を終了する($execution_{finish}$).

6. 計測後処理
   計測後，被験者はiPadとTaskPitの計測ログを実験者へ渡す. 実験者は，iPadのNow Then Proの計測ログを出力する.

5.1.5 レビューと評価

本実験の評価は，Now Then Pro と TaskPit の計測ログを用いて行う.

計測後，実験者は被験者と共に計測したログをレビューしながら，表1で定義した6つの作業時刻を決定し，それを基に5.1.2.3項で示した3つの作業時間 *make*, *check*, *execution* を各々(1)式，(2)式，(3)式を用いて算出する.

$$make = make_{finish} - make_{start} \tag{1}$$

$$check = check_{finish} - check_{start} \tag{2}$$

$$execution = execution_{finish} - execution_{start} \tag{3}$$

3つの作業時間から，OJSを導入しない従来の講義方法とOJSを導入した講義方法における作業時間 *noOJS*, *OJS* を各々(4)式，(5)式を用いて算出し，評価に用いる.

$$noOJS = check + execution \tag{4}$$

$$OJS = make \tag{5}$$

5.2 講義アンケートの実施

5.2.1 実施概要

OJSの導入が教員の負担軽減に有用であったとしても，受講者にとってOJSがプログラミング講義に有用でなければ，OJSの導入は価値がない．そこで，受講者にとってOJSがプログラミング講義に有用であるかどうかを調べるため，2016年度のプログラミングII受講者を対象に，OJSによる講義についてアンケートをオンライン上で実施した．実施時期は，後期中間分の講義が終了する8回目の講義受講後（2016年11月29日~12月11日）である.

5.2.2 調査項目とアンケート設問

本アンケートで調査する項目は，以下の3点である．OJSの使いやすさ，OJSの即時フィードバック機能，OJSによる講義の総合評価の3点である.

- 講義に取り組む上でのOJS(Sharif Judge)の使いやすさ
  OJSが使いづらいことによる講義の取り組みに対する受講者のフラストレーションが生じていないかどうかを調査する.

- OJSの即時フィードバック機能
  OJSによる採点が教員による手動採点の場合に比べてプログラムの正誤判定結果が明らかになることで生じる即時的効果が受講者にとって学習に役立つかどうかを調査する.

- OJSによる講義の総合評価

  総合的に，OJSの講義導入に価値があるかどうかを調査する．また，価値の有無の理由を自由記述による回答で明らかにする．

実施したアンケートの6つの設問を図7に示す．アンケートの設問は，4段階のリッカート尺度で回答する3つの設問と自由記述式で回答する3つの設問の合計6つの設問で構成されている．

[OJSの使いやすさに関する質問]
Q1. 本講義で用いたOJSは使いやすかったですか．以下の4つから選択してください．
4（使いやすい）
3（どちらかというと使いやすい）
2（どちらかというと使いづらい）
1（使いづらい）

Q2. 本講義で用いたOJSは，どういう所が使いやすい（あるいは使いづらい）ですか．
（自由記述式による回答）

[OJSの即時フィードバック機能に関する質問]
Q3. OJSの正誤判定機能による即時フィードバックは，プログラミング学習に役立つと思いますか．以下の4つから選択してください．
4（思う）
3（どちらかというと思う）
2（どちらかというと思わない）
1（思わない）

Q4. Q3に関して，具体的にどういう点で役立つ（あるいは役立たない）と思いますか．
（自由記述式による回答）

[OJSによる講義の総合評価に関する質問]
Q5. OJSによる講義は総合的に良いと思いますか．以下の4つから選択してください．
4（思う）
3（どちらかというと思う）
2（どちらかというと思わない）
1（思わない）

Q6. OJSによる講義について何か意見・感想があれば記述してください．
（自由記述式による回答）

図7 OJSによる講義に関するアンケート

6 結果と考察

6.1 教員の作業時間

実験結果を図8に示す. 図8には，2回分の計測における，OJSを導入しない従来の講義方法(*noOJS*)とOJSを導入した講義方法(*OJS*)の作業時間を示しており，単位は時間である.

1回目の計測結果は，OJSを導入しない従来の講義方法は3.57時間，OJSを導入した講義方法は0.76時間となった. 2回目の計測結果は，OJSを導入しない従来の講義方法は2.00時間，OJSを導入した講義方法は0.74時間となった. 以上より，1回目，2回目ともにOJSを導入した講義方法に比べ，OJSを導入しない従来の講義方法の方が作業時間が短いという結果になった.

OJSを導入しない従来の講義方法の作業計測に用いたソースコード（以下，手動採点で用いたソースコード）は，受講者がSharif JudgeのWebサーバに提出したソースコードを用いている. OJSは，提出されたソースコードにおいて，ファイル名やクラス名に誤りがあったり，文字化けが発生した場合，その時点で採点されない. この場合，ソースコードを提出した受講者自身がファイル名やクラス名，文字化けを修正して再提出する必要がある. 本実験において，手動採点で用いたソースコードはファイル名やクラス名に誤りはなく，かつ文字化けの発生も無い状態で採点が可能となるため，ファイル名やクラス名，文字化けを修正するための作業時間を考慮することができなかった. したがって，OJSを導入しない従来の講義方法における作業時間が短くなったと考えられる.

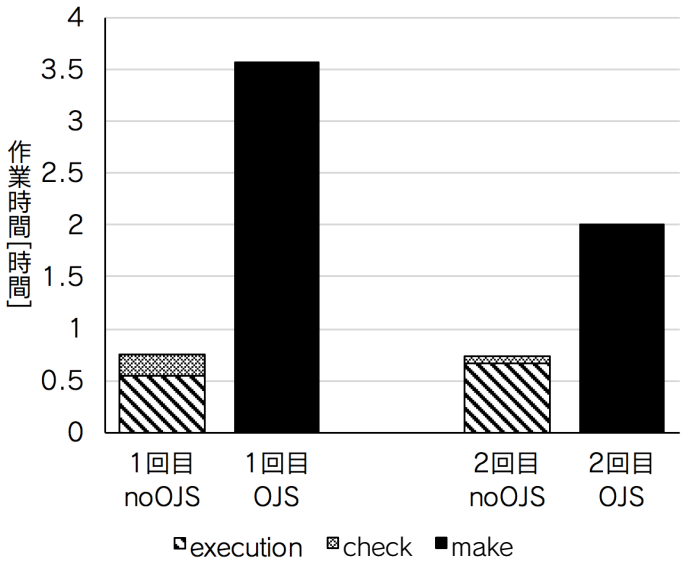

図8 作業時間

本実験は，OJSを初めて講義に導入したときに作業時間を計測し，教員の負担軽減を検証した．しかし，本実験結果は短期的視点によるものであるため，OJSの長期的視点での考察が必要となる．そこで，長期的視点で見た教員の負担軽減について述べる．

実験結果より，OJSを導入した講義方法の作業時間はOJSを導入しない従来の場合に比べ，1回目の計測では4.70倍，2回目の計測では2.70倍長いという結果になった．

本実験で計測の対象としたOJSを導入した講義方法における作業は，演習課題・テストケースの作成である．本実験では，初めてOJSを講義に導入したため，従来の講義方法で使用した演習課題をOJSで正誤判定できるように仕様を変更する必要があった．したがって，課題の仕様を変更するための作業時間が発生し，演習課題・テストケースの作成の作業時間が長くなったと考えられる．しかし，次年度以降は今年度の演習課題を再利用できるため，演習課題・テストケースの作成の作業が不要となり，OJSを導入した講義方法における作業時間を今年度と比べて短縮することができる．

一方，OJSを導入しない従来の講義方法の場合は，上記の作業は不要であるが，手動による採点作業が必要となる．3.3節で述べた通り，従来の手動による採点の場合，実行結果の確認漏れ等のヒューマンエラーの発生によって正確な採点結果が得られるとは限らない．また，ヒューマンエラーの対処として，一通り行った採点の確認作業を行うことによる作業時間が発生し，教員の負担増加が考えられる．

本実験の1回目の計測の場合，OJSを導入した講義方法の作業時間はOJSを導入しない従来の場合に比べ4.70倍長くかかるという結果になった．すなわち，5年以上OJSを継続して利用した場合，OJSを導入した講義方法の方がOJSを導入しない従来の講義方法より作業時間が短くなると解釈できる．

2回目の計測の場合についても，1回目の計測の場合と同様に，OJSを導入した講義方法の作業時間はOJSを導入しない従来の場合に比べ2.70倍長くかかるという結果になった．すなわち，3年以上OJSを継続して利用した場合，OJSを導入した講義方法の方がOJSを導入しない従来の講義方法より作業時間が短くなると解釈できる．

したがって，OJSを講義に導入することによる作業時間的な採算性は高いことが考えられるため，長期的視点では，OJSの導入は有用であるといえる．

6.2 講義アンケートの回答結果

アンケートの回答者数は，プログラミングII受講者39名中23名であった．アンケートの回答結果を表2に示す．表2の値は，図7のQ1, Q3, Q5における，4段階 (1~4) の各々の回答率である．

表2のQ1より，91.3%の学生が，OJSは使いやすい (4)，もしくは，どちらかというと使いやすい (3) と回答した．Q2の自由記述では，”即時採点が良い”という回答が多数あり，採点の即時性という点で受講者のフラストレーションの発生が生じることはなく，OJSは使いやすいことが示唆された．一方，否定的な意見として，”出力のフォーマットが違う場合，気がつきにくい”，”出力におけるスペースの数が一致しないと0点になり，使いづらい”等，OJSの採点方法に関する意見が見られた．

表2のQ3より，全員の学生が，OJSの即時フィードバック機能はプログラミング学習に役立つ (4)，もしくは，どちらかというと役立つ (3) と回答した．Q4の自由記述では，”間違いがすぐわかる”，”ブラックボックス的に円滑にテストを行えるので，プログラミングの開発速度が向上できると思う”等，全員の学生が肯定的な回答をしており，OJSの即時フィードバック機能はプログラミング学習に役立つことが示唆された．

表2のQ5より，全員の学生がOJSによる講義は総合的に評価できる (4)，もしくは，どちらかというと評価できる (3) と回答した．Q6の自由記述では，”授業が円滑に進む”，”講義中だけでなく，自宅での学習も格段にやりやすくなった”等，多数の学生が肯定的な回答をしており，OJSによる講義は総合的に良いことが示唆された．一方，否定的な意見として，”インデントやコメントが適切かどうか見てほしい”，”定期的にコードの書き方について指導してほしい”等，本講義で用いたOJSでは採点できないコーディングスタイルのチェック機能に関する意見や”提出ごとに採点してくれるため，テスト用のコードを書かなくなった”と動作テストの怠慢に関する意見が見られた．

以上のアンケート結果から，3点の調査項目についていずれも高評価であり，受講者にとってOJSがプログラミング講義に有用であることが示唆された．一方，自由記述の設問で，OJSに関する否定的な意見が幾つか見られたが，これについては，6.3節で述べる．

表2 アンケートの結果 (Q1, Q3, Q5)

指標	Q1（使いやすさ）	Q3（即時フィードバック機能）	Q5（総合評価）
1	0.0%	0.0%	0.0%
2	8.7%	0.0%	0.0%
3	30.4%	21.7%	4.3%
4	60.9%	78.3%	95.7%

6.3 OJSのトレードオフ

本節では，プログラミング講義でのOJSの導入によって生じるトレードオフについて述べる．

6.3.1 出力部分の文字列マッチングの厳密性

6.2節の講義アンケートの自由記述より，OJSの採点方法がシビアであるという意見が多く見られた．OJSによる正誤判定はテストケースの出力部分との文字列のマッチングによって行われる．そのため，課題に対する解答のアルゴリズムが合っていたとしても，日本語の文字化けや空白数の過不足，アルファベットの大文字・小文字の違いなど，フォーマットが仕様どおりでなければ不正解と判定される．実務上の場合，これらはバグとして扱われるうえに，システムによっては重大なトラブルを引き起こす可能性がある．よって，OJSによる正誤判定の厳密性は，実務上において重要であると考えられる．しかし，プログラムのアルゴリズムが合っていたとしても，受講者はOJSによって正解と判定されるまで，出力部分のフォーマットを修正し続ける必要がある．したがって，OJSによる正誤判定の厳密性が受講者の学習意欲を下げる可能性が考えられる．

6.3.2 出力が不定であるプログラムの採点

OJSは，乱数や並列処理等によって，プログラムの出力が不定であるソースコードを採点することができない．なぜなら，テストケースは入力と出力が一意的かつ固定でなければならないからである．したがって，プログラムを実行しないと出力の値が定まらないため，テストケースによる正誤判定が不可能となる．

7 結論

本稿では，Online Judge System(OJS)をプログラミング講義へ導入することによって，教員の負担が軽減できるか検証することを目的に，教員が作業に要する時間を計測した．実験では，被験者がプログラミング講義に要する作業時間を，OJSを導入しない従来の講義方法とOJSを導入した講義方法の2つの方法で計測した．

計測の結果，OJSを導入した講義方法で要する作業時間の方がOJSを導入しない従来の講義方法に比べて1回目の計測では4.70倍，2回目の計測では2.70倍長かった．本実験では，ファイル名やクラス名，文字化けを修正するための作業時間を考慮できなかったため，OJSを導入しない従来の講義方法の作業時間が短くなったと考えられる．長期的視点で見た場合，OJSのスケーラビリティ，採点の正確性の保証と従来の手動による採点方法で発生するヒューマンエラーを考慮すると，OJSの講義への導入は有用であると考えられる．

また，OJSの講義への導入が受講者にとって有用であるかどうかを調査するために，OJSの使いやすさ，OJSの即時フィードバック，OJSによる講義の総合評価の3点について講義アンケートを実施した．アンケートの結果，3点全ての項目において，いずれも高評価であり，受講者にとってOJSがプログラミング講義に有用であることが示唆された．

本研究の今後の課題として，受講者への講義アンケートにおいて多く見られた意見である，OJSの採点方法の厳密性の緩和が挙げられる．空白数の過不足やアルファベットの大文字・小文字の違い等による不正解のような，OJSの採点方法の厳密性が受講者にストレスを与えてしまい，学習意欲を下げる可能性が考えられる．また，教員にとっても入出力のフォーマットを厳密化する作業が発生することが考えられる．したがって，OJSをプログラミング講義に導入するにあたり，OJSの採点方法に柔軟性を確保させる必要がある．今後，正誤判定に用いる手段として，単純な文字列のマッチングによる判定方法だけでなく，正規表現を用いた判定方法やテストケースの出力部分とプログラムの実行結果の2つを差分によって判定する方法等を用いて，OJSによる柔軟性のある採点を可能にしたい．

また，本研究で行った実験は被験者が1名で，対象講義がプログラミングIIの後期であった．そのため，本システムの汎用性を検証するため，一般的にプログラミング講義での作業とはどういうものなのかを再考した上で，実験設定を見直す必要がある．

謝辞

本論文の執筆および研究を進めるにあたり，多くの方々に協力していただきました．この場を借りて御礼申し上げます．ありがとうございました．

指導教員である上野秀剛講師には，お忙しい中，貴重な時間を割いて被験者実験に協力いただいたり，輪講や論文のチェックの指導をしていただき，的確なご指導をいただきました．ありがとうございました．

査読教員である松村寿枝准教授には，貴重なご意見・ご指摘をいただき，心からの感謝申し上げます．ありがとうございました．

講義アンケートの回答にご協力いただきました情報工学科3年生の皆さんに心からの感謝申し上げます．ありがとうございました．

日常の議論を通して，多くの指摘をいただいた上野研究室の皆様へも心から感謝の気持ちと御礼を申し上げます．ありがとうございました．

参考文献

[1] ”【2016年最新】プログラミング教育の今と未来!課題と取り組みまとめ — 侍エンジニア塾ブログ — プログラミング入門者向け学習情報サイト”, http://techacademy.jp/magazine/8525, (参照 2016-10-31).

[2] ”なぜ、小学校からの「プログラミング教育」が必要?| ベネッセ教育情報サイト”, http://benesse.jp/kyouiku/201606/20160603-1.html（2017-1-23 参照）

[3] ”AIZU ONLINE JUDGE”, http://judge.u-aizu.ac.jp/onlinejudge/index.jsp（2017-1-23 参照）

[4] ”ブラウザでプログラミング･実行ができる「オンライン実行環境」— paiza.IO”, https://paiza.io（2017-1-23 参照）

[5] 古谷勇樹, 林真史, 山本隆弘, 長尾和彦:”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”, 情報科学技術フォーラム講演論文集, Vol.14, No.3, pp.89-94(2015).

[6] Hui Sun, Bofang Li, Min Jiao:”YOJ: An online judge system designed for programming courses”, 9th International Conference on Computer Science Education (ICCSE2014), pp.812-816(2014).

[7] 岩本舞, 中村真人, 小島俊輔:”不正コピー検知機能を備えた学習用オンラインジャッジシステムの構築と評価”, 日本工学教育協会, Vol.62, No.3, pp.3_3-3_8(2014).

[8] 渡部有隆:”オンラインジャッジではじめる C/C++プログラミング入門”, マイナビ出版, pp.12-16(2014).

[9] ”オンラインジャッジの作り方/実装 - #define int ll”, http://d.hatena.ne.jp/wistery_k/20120428/1335621313, (参照 2016-12-2).

[10] 椿本弥生:”計量的手法を用いた大学におけるレポート採点支援に関する研究”, 東京工業大学大学院社会理工学研究科博士論文 (2008).

[11] ”Eclipse - The Eclipse Foundation open source community website.”, https://www.eclipse.org（2017-1-23 参照）

[12] ”GitHub - mjnaderi/Sharif-Judge/ A free and open source online judge system for programming courses”, https://github.com/mjnaderi/Sharif-Judge（2017-1-23 参照）

[13] ”Now Then Pro - The must-have time tracking app for your iPhone, iPod touch or iPad”, http://angryaztec.com/nowthen.html（2017-1-23 参照）

[14] ”/開発行動記録システム TaskPit/.webarchive”, http://www.taskpit.jpn.org/index.html（2017-1-23 参照）