情報処理学会論文誌　教育とコンピュータ　Vol.1　No.4　38–47 (Dec. 2015)

論　文

不正コピー検出手法を備えたオンラインジャッジシステムの開発

岩本 舞[1,a)]　中村 真人[2,b)]　小島 俊輔[3,c)]　中嶋 卓雄[4,d)]

受付日 2014年11月26日，再受付日 2015年3月23日，
採録日 2015年9月5日

概要：プログラミングの学習では，学生が様々な課題に取り組むことが重要である．しかしながら，教員が課題を採点するのに時間がかかるため，多くの課題を課することは難しい．そこで，本研究では，学生に多くの課題を解いてもらうために，オンラインジャッジシステムを開発した．またこのようなシステムでは，他人のソースコードを流用し提出する行為が問題となる．そのため，オンラインジャッジに不正コピーを検出する機能を実装した．以前の筆者らの研究では，トークン列長を判定基準とする手法を提案したが，単純な長さをしきい値としており，誤検知が多かった．そこで，本稿ではトークン列の複雑度と完全トークン列を用いた検出手法を提案する．実験の結果，学生が提出した 3 種類の課題プログラムにおいて，適合率 P が大幅に上昇した．また，本システムを使用した学生にアンケートを実施したところ，89%の学生が，プログラムを書けるようになったと回答した．

キーワード：プログラミング教育，オンラインジャッジ，不正コピー

Development of an Online Judge System for Learning with Detecting Illicit Copies

Mai Iwamoto[1,a)]　Masato Nakamura[2,b)]　Shunsuke Oshima[3,c)]　Takuo Nakashima[4,d)]

Received: November 26, 2014, Revised: March 23, 2015,
Accepted: September 5, 2015

Abstract: Learning computer programming requires students to deal with different variations of assignments. Instructors, on the other hand, have difficulty giving students sufficient volume of assignments due to time constraints assessing students' work, which is highly time-consuming. This study presents an online judge system to check the students' assignments so that students will have more opportunities to tackle different assignments. A system of this kind, however, can be problematic in that students may copy other source codes and submit it as their own work. We implemented a function to detect illicit copies into the system. In previous researches of ours, such methods based on the detection standard of the token length have been proposed, with many misdetections occurred. This paper proposes the detection method using the token sequence complexity and the complete token sequence. As the results of experiments, our new method was successful in improving the accuracy rate on three different kinds of assignments submitted by students. Moreover, we sent out questionnaires to students about the system, to be reported from 89% of them out of the entire respondents that they became capable of programming after using the new system.

Keywords: programming education, online judge, illicit copy

1. はじめに

一般に，学校の授業は教科書を用い，教員が教科書の内容を解説する形式で行われる．しかし，プログラミングの技術を習得するためには，教員の指導を受けて本を読む

1 熊本高等専門学校技術・教育支援センター
Center for Technical and Educational Support, Kumamoto National College of Technology, Yatsushiro, Kumamoto 866–8501, Japan
2 株式会社リクルートホールディングス
Recruit Holdings Co.,Ltd., Chiyoda, Tokyo 100-6640 Japan
3 熊本高等専門学校 ICT 活用学習支援センター
ICT Center for Learning Support, Kumamoto National College of Technology, Yatsushiro, Kumamoto 866–8501, Japan
4 東海大学
Tokai University, Toroku, Kumamoto 862–8652, Japan

a) m-iwamoto@kumamoto-nct.ac.jp
b) masato_nakamura@r.recruit.co.jp
c) oshima@kumamoto-nct.ac.jp
d) taku@ktmail.tokai-u.jp

だけでは不十分で，学生が様々な課題に取り組み，ソースコードを自らの力で書くことが重要である．よって，授業形式も，単に教員が解説して終わるのではなく，学生が意欲的に課題に取り組めるようにすることが望ましい．

従来，本校のプログラミング系講義では，学生にプログラミングの課題を課する場合，紙媒体でレポートを提出させ，理解していることを確認するために口頭試問を行うという形式をとっていた．しかし，従来の課題形式には，以下の 2 つの問題がある．

第 1 に，教員が課題を評価する時間がかかるため，多くの課題を課することが難しいという問題がある．そのため，教員の負担を軽減し，さらに，学生がすばやく自分のプログラムに対するフィードバックが得られる環境が必要である．第 2 に，学生が他の学生の解答をコピーして提出するという問題である．以後，コピーして提出された解答を**不正コピー**と記す．従来の形式では，教員は 100 人以上の学生から提出されるソースコードを見比べる必要があり，不正コピーを発見することは難しい．そのため，学生が理解しているか否かを確かめる口頭試問を行っていたが，これが評価に時間がかかり多くの課題を課せない原因となっていた．

以上の問題を解決するため，我々はオンラインジャッジシステムを導入した．オンラインジャッジとはユーザが課題に対する解答ソースコードを提出すると，システムが自動でソースコードをコンパイルし，解答の正誤を判定するシステムである．本システムでは，先の問題を解決するため，一般的なオンラインジャッジの機能に加え，インデントなどの基本的なコーディングスタイルの誤りを指摘する機能を実装した．さらに，オンラインジャッジシステム上に類似のソースコードを検知するシステムを導入し，存在を周知することで，不正コピーの抑止を図った．また不正コピーを常習的に行う学生（以下，不正コピー常習者）を特定できるよう教員を支援することで，授業を理解できておらず課題に手を付けられない，まったくやる気がないなど，教員によるサポートが必要な学生に対する指導を可能とした．

2. 関連研究

2.1 オンラインジャッジ

オンラインジャッジシステムは会津大学の AOJ（AIZU ONLINE JUDGE）[13] や topcoder [14] が有名だが，教員による課題の追加や変更，成績管理の仕組みがないため実際に授業で利用することは難しく，また，システムを教育機関の都合に合わせて改変することができない．さらに，既存のシステムはプログラミング上級者をユーザとして想定しているため，応用的な課題が多い．したがって，初級者向けの講義で使用するのは難しい．

また，これらのシステムは基本的にプログラミングを趣味としている人々のものであるため，不正コピー検出などの機能は備わっていない．そこで本研究では，授業で使用するのに必要な機能を備えたオンラインジャッジシステムを開発する．

2.2 不正コピー検出

ソースコードにおけるコピーを検出する方法として，コードクローン検出手法がある．従来のコードクローン研究では，主に産業界の大規模なコードクローンを検出する手法が提案されてきた．これらのコードクローン検出手法は，木やグラフを基礎とした構文レベルの手法と，テキストやトークンを比較する字句レベルの手法に分類することができる．文献 [17] では，様々な手法によるコードクローン検出方法を比較・検証している．

構文レベルの手法として，文献 [8] では，依存グラフによりコードクローンを同定する．文献 [2] では，抽象構文木を用いたアルゴリズムによりコードクローンを発見する．構文レベルの手法は，コードクローンを構造的なまとまりとして検出できる．

また字句レベルの手法として，文献 [9] は，判定に 21 種類の関数メトリクスを使用しており，それらを 4 つに分類して比較・検討している．また，文献 [1], [7] では，巨大なソフトウェアにおいて，コードクローンを高速に発見することを目的としたツールが開発されている．比較前にユーザ定義名を特殊文字に置き換えるため，定義名が異なる場合でもコードクローンとして検出が可能である．行単位での比較には接尾辞木検索アルゴリズムが用いられており，線形時間でコードクローンが検出可能である．文献 [15] では，動的計画法に基づいた手法により，以前のテキスト比較ツールより高精度に 2 つのプログラム間のコードクローンを検出することができる．文献 [16] では，ソフトウェア間の最大クローン長と部分類似度に着目し，どの程度の値であれば流用があるといえるかを実験的に導出している．

さらに，文献 [3] では，文献 [1], [2], [7], [8] など 6 つのコードクローン検出方法を，8 つの C や Java で記述されたプログラムを用いて検証している．その結果，字句レベルの技術 [1], [7] はかなり高い再現率となることを示した．

本研究で提案するアルゴリズムは，文献 [7], [9] で用いられたトークンベースのコードクローン検出手法に分類される．トークンベースの手法を選択した理由は，不正コピーにおける特徴を検出する機能を有しているためである．しかし，これまでの字句レベル手法の研究は，類似するコード片を発見するものであり，不正コピーか否かについては言及しない．たとえば，学生が課題演習で提出するソースコードにおいて，単純なコードは類似する可能性が高く，偶然似たものをコピーと判定してしまう．また，学生は不正コピーを隠すため，ステートメントの入れ替えなどの細工を施す場合があり，従来の手法では不正コピーを見逃す

```
printf("名前を入力してください\n");
scanf("%s",name);
printf("年齢を入力してください\n");
scanf("%d",&age);
printf("電話番号を入力してください\n");
scanf("%s",phone);
```

図 1　単純なステートメントの繰返しの例
Fig. 1　An example of iteration using simple statements.

可能性があった．さらに，制御ロジックが変化する行の入れ替えをしただけのソースコードを提出する場合もあり，接尾辞木や抽象構文木を利用した手法においても，正しい判定が困難となる．そこで筆者らの研究 [5], [6] では，学生の不正コピーの検出手法を提案している．学生による不正コピーには，大きく分けて以下の 5 つの特徴がある．

TypeV　ユーザ定義名，数値定数，文字列などの付け替え
TypeC　コメント行の追加・削除・変更
TypeI　インデントの編集
TypeS　関数・ステートメント行の入れ替え
TypeD　行の入れ替えによりロジックが変更されたコピー

また，以下のような，コードクローンだが不正コピーとして検出するべきではない特徴もある．

TypeT　教員が雛形ソースコードを提供するタイプの課題
TypeP　単純なプログラム

文献 [5] では，ソースコードのトークン列中から同一となるトークン列を検索し，トークン列長をしきい値としてコピーされた部分を判定し，全体に占める割合を不正コピー率とする手法を提案した．文献 [5] の手法では，TypeV，TypeC，TypeI，TypeS，TypeD といった特徴を検出し，TypeT の特徴を持つソースコードを検出しない．しかし文献 [5] の手法では，トークン列長をコピー判定のしきい値としており，**図 1** のような誰が書いても同じ書式となる単純なプログラムも，トークン列長がしきい値以上であれば検出され，誤検知が増える要因となっていた．そこで筆者らは従来手法 [5] をベースに，TypeP の特徴を持つソースコードを検出しない，より誤検知の少ない検出アルゴリズム [6] を提案した．本システムでは文献 [6] の手法を用いて不正コピー検出を行っている．

3.　不正コピー検出手法

ここでは，不正コピーを検知するための手法について説明する．検知までの大まかな処理の流れを以下に示す．なお，本稿では，列を ()，集合を {} と表記する．またトークン列 $\mathbb{P} = (p_1, p_2, \cdots, p_m)$ において，α 番目のトークン p_α を起点とし，β 番目（$\alpha \leq \beta$）のトークン p_β まで続く部分トークン列を $(p_k | \alpha \leq k \leq \beta)$ と記述し，トークン列 $\mathbb{P}$ の要素からなる集合を，$\{\mathbb{P}\}$ のように記述する．

Step1　前処理
ソースコードのコメントおよびインデントを削除し，雛形ソースコードとの差分を抽出する．雛型ソースコードとは，学生にそれを書き換えさせることを目的として教員から提示されたソースコードである．たとえば，ある一定の動きをするソースコードを提示し，その拡張や改変を行う演習を課した場合，そのまま不正コピーであるか否か判定すると，雛型ソースコードの部分が不正コピーであると見なされるため，差分を抽出してから不正コピー検出を行う必要がある．差分抽出にはレーベンシュタイン距離 [12] を用いる．さらに抽出した差分について，トークン解析を行う．

Step2　同一トークン列の取り出し
不正コピーであるか否かを判定したい 2 つのソースコードのトークン列を用意する．ここでは，$\mathbb{P}$ をコピー元，$\mathbb{Q}$ をコピー先のプログラムのトークン列とする．ここで $p_i \in \{\mathbb{P}\}$，$q_j \in \{\mathbb{Q}\}$ はそれぞれソーコードの先頭から i 番目，j 番目のトークンを示し，それぞれ写像関数 τ_p，τ_q によって $\mathbb{T}$ の該当するトークンに写像される．$\mathbb{T}$ へ写像した結果が同一となる元の要素が $\mathbb{P}$，$\mathbb{Q}$ の双方に存在するとき，その 2 つの要素を**同一トークン**と表記する．ただしこの段階では，ユーザ定義名や数値の付け替えである TypeV の不正コピーに対応するため，ユーザ定義名は同じトークンであると見なし，数値についても同様に扱う．本研究では，このような同一トークンが連続したトークン列を**同一トークン列**と表記する．このとき，

$$\mathbb{P} = (p_1, p_2, \cdots, p_m) \tag{1}$$

$$\mathbb{Q} = (q_1, q_2, \cdots, q_n) \tag{2}$$

$$\mathbb{T} = \{ \text{すべてのトークン} \} \tag{3}$$

$$\tau_p : \{\mathbb{P}\} \to \mathbb{T} \tag{4}$$

$$\tau_q : \{\mathbb{Q}\} \to \mathbb{T} \tag{5}$$

とする．$\mathbb{P}$ の i 番目，$\mathbb{Q}$ の j 番目のトークンを起点とする同一トークン列長を求める関数 $L(i, j)$ を以下のように定義する．

$$L(i,j) = \min\{l | 0 \leq l, \tau_p(p_{i+l}) \neq \tau_q(q_{j+l})\} \tag{6}$$

式 (6) より，$\mathbb{P}$ の i 番目，$\mathbb{Q}$ の j 番目のトークンを起点とする同一トークン列 $\mathbb{C}_{ij}$ は次のように定義できる．

$$\mathbb{C}_{ij} = (q_{j+r} | 0 \leq r < L(i,j)) \tag{7}$$

例を**図 2** に示す．ここで，図中の黒点はそれぞれのトークン列で同一トークンが使われていることを示す．このとき，黒点が斜めに連なっている部分が同一トークン列である．

編集距離を用いた同一トークン列の発見手法では，**図 3**

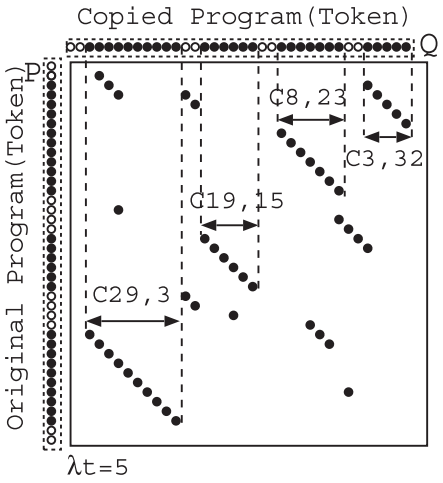


図 2　同一トークン列の検出

Fig. 2　Detection of the same token sequence.

```
ID:a - NUM ) printf ( STR ) ;
}while ( ID:no != ID:a && ID:b < ID:a ) ;
printf(STR,
```

図 3　同一トークン列の例

Fig. 3　An example of the same token sequence.

```
printf ( STR ) ;
}while ( ID:no != ID:a && ID:b < ID:a ) ;
```

図 4　完全トークン列の例

Fig. 4　An example of complete token sequence.

のように，同一トークン列がステートメントの途中から開始または終了する場合がある．しかし，実際にはステートメントが不完全な状態でコピーされるとは考えにくい．そこで，ステートメントの途中から始まる同一トークン列や，ステートメントの途中で終わる同一トークン列を削除し，ステートメントとして成立している部分だけを判定の対象とする**完全トークン列**手法（Complete Token Sequence Method，以下 CTSM と略記）を提案する．完全トークン列のみを判定の対象とすることで，不完全なトークン列によって複雑度がしきい値 λ_t を超えることを防ぎ，誤検知を減らす狙いがある．たとえば，図 3 に示すような同一トークン列は，CTSM を適用することで**図 4** になる．

$\mathbb{E}$ はステートメントの区切り記号の集合である．今回の実験では，C 言語で記述されたソースコードを対象としたため，$\mathbb{E}$ ={';', '}', ')'} とした．C 言語では，')' が関数や演算の優先順位の指定で使用されており，完結していない命令文を取り出す可能性がある．しかしながら `if()`，`for()`，`while()` で使用されており，条件式などの特徴的なステートメントを不正コピー検出に含めるため，区切り記号の集合 $\mathbb{E}$ に含めた．どのような区切り記号が適当かは今後さらなる検証が必要である．ここで，$\mathbb{K}_{ij}$ は q_{j-1} と $\mathbb{C}_{ij}$ からなるトークン列に含まれる区切り記号の位置の集合であり，以下のように定義する．

$$\mathbb{K}_{ij} = \{k | j-1 \leq k < j + L(i,j), \\ (k = 0 \text{ または} \tau_q(q_k) \in \mathbb{E})\} \tag{8}$$

このとき，完全トークン列 $\mathbb{C}'_{ij}$ を次式で定義する．

$$\mathbb{C}'_{ij} = (q_k | \min(\mathbb{K}_{ij}) < k \leq \max(\mathbb{K}_{ij})) \tag{9}$$

ここで，min および max は，それぞれ集合に含まれるスカラー値の最小・最大値を返す記号として用いた．なお，式 (8) において $j-1 \leq k$ としたのは，ステートメントが完全にコピーされている場合，$j \leq k$ とすると最初のステートメントが消失するためである．

このようにして作成した完全同一トークン列 $\mathbb{C}'_{ij}$ について，コピートークン列か否か判定する．

Step3　コピートークン集合の作成

Step2 で取り出した各々の同一トークン列について，コピーされたものかどうかを判定する．本研究では，同一トークン列 $\mathbb{C}_{ij}$ のうち，コピーであると判定された同一トークン列を**コピートークン列**，すべてのコピートークン列の和集合を**コピートークン集合** $\mathbb{C}$ と表記する．$\mathbb{C}_{ij}$ がコピートークン列であるか否かの判定には，トークン列の複雑度に基づく手法（Token Complexity Method，以下 TCM と略記）を提案する．TCM では，トークン列の複雑度という概念を導入し，単純な構造のプログラム不正コピーと判定しないようにした．

ソースコードの複雑さについて，文献 [11] では，ルーチンのディシジョンポイントの個数で計測できるとしている．また文献 [10] では，ソースコードの複雑さを実行可能なパスの数としている．これらの文献では，プログラムの難読化を防ぐ目的で複雑さを評価しているため，本研究における複雑度の概念とは意味合いが異なる．そこで本研究では，次式のように定義した複雑度を提案する．

$$\text{Complexity}(\mathbb{C}_{ij}) = \left|\{\tau_q(q) | q \in \{\mathbb{C}_{ij}\}\}\right| \\ + K\left|\{q | q \in \{\mathbb{C}_{ij}\}, \tau_q(q) \in \mathbb{R}\}\right| \tag{10}$$

ここで $\mathbb{R}$ は繰返しトークンの集合であり，C 言語では $\mathbb{R}$ ={`for`, `while`} となる．第 1 項は使用されたトークンの種類数，第 2 項は繰返しトークンの総数である．複雑度の式では，定義名の違うユーザ定義名を別のトークンとして数える．たとえば，

```
a=a*10+b;
```

というステートメントには，`ID:a`, `=`, `*`, `NUM`, `+`, `ID:b`, `;` の 7 種類のトークンがある．

表 1　実験で使用したデータセット
Table 1　Datasets used in this experiment.

	課題形式	オーダー	雛形の有無（行数）	平均行数（雛形・コメント除く）	提出件数	不正コピー疑い件数
課題 1	自由拡張型	-	あり（29 行）	30.7	119	27
課題 2	課題解決型	$O(n^2)$	なし	32.7	116	44
課題 3	課題解決型	$O(n^3)$	なし	36.3	85	36

繰返しトークンは他のトークンに比べ複雑度を増加させる傾向が強いと考えられるため，重み付けの定数 K を 0 より大きな値とする．

TCM では，複雑度のしきい値を λ_t とし，コピートークン集合 $\mathbb{C}$ を次式のように定義する．λ_t が大きいほど，同一トークン列を誤ってコピートークン列とする可能性は小さくなる．

$$\mathbb{C} = \bigcup_{\mathrm{Complexity}(\mathbb{C}_{ij}) \geq \lambda_t} \{\mathbb{C}_{ij}\} \tag{11}$$

Step4　コピー率による不正コピー判定

Step3 で作成したコピートークン集合 $\mathbb{C}$ が全体のトークン数に占める割合がしきい値 λ_c（$0 \leq \lambda_c \leq 1$）を超えたとき，そのプログラムは不正コピーであったと判定する．すなわち，プログラムのコピー率 $\mathrm{CR}(\mathbb{C}, \mathbb{Q})$ を式 (12) のように定義したとき，その値が式 (13) を満たせば，そのプログラムは不正コピーであると判定する．λ_c が 1 に近づくことは，プログラムを不正コピーと見なす条件が厳しくなることを意味する．

$$\mathrm{CR}(\mathbb{C}, \mathbb{Q}) = \frac{|\mathbb{C}|}{|\mathbb{Q}|} \tag{12}$$

$$\mathrm{CR}(\mathbb{C}, \mathbb{Q}) \geq \lambda_c \tag{13}$$

4.　不正コピー検出精度

4.1　評価基準

コードクローン研究では，一般的に誤検知を用いた評価基準が使用される．ここでは，不正コピーを検出した True-Positive（以下 TP），不正コピーでないものを検出した False-Positive（以下 FP），不正コピーを検出できなかった False-Negative（以下 FN）を実験により求めた．

ここで，FP，FN を客観的に評価するための一般的な尺度となる再現率 Recall（以下 R），適合率 Precision（以下 P）を導入する．R，P は，一般に次式で定義される．

$$R = \frac{tp}{tp + fn} \tag{14}$$

$$P = \frac{tp}{tp + fp} \tag{15}$$

tp，fn，fp はそれぞれ TP，FN，FP の数である．R，P はそれぞれ 0 以上 1 以下の値をとり，1 に近いほど判定が正確であったことを意味する．

不正コピー検出機能は，教員による不正コピー常習者の特定支援を目的として実装する．一例として，$R = 0.8$ とし，ある 1 人の学生が不正コピーを 10 回行った場合，本システムが 5 回以上不正コピーとして検出する確率は二項分布により 99.4%となる．よって，個々の不正コピーの検出率は 80%でも十分に不正コピー常習者を特定できると考え，$R \geq 0.8$ に設定し，P の向上を目指す．

表 2　従来手法と提案手法の比較
Table 2　Comparison of the previous method and the proposed method.

	従来手法			提案手法		
	λ_c	R	P	λ_c	R	P
課題 1	0.69	0.85	0.50	0.53	0.81	0.73
課題 2	0.78	0.89	0.27	0.72	0.84	0.76
課題 3	0.80	0.81	0.36	0.72	0.81	0.76

4.2　実験データ

今回の実験では，熊本高等専門学校八代キャンパスの 3 年次のクラスで実施したプログラミング演習の中から特徴が異なる 3 つの課題を選出し，それぞれに対して学生が提出したプログラムを実験データとした．**表 1** にそれぞれのデータセットの詳細を示す．なお，課題形式は，ある一定の入力に対し一意に定まる出力が要求される課題を「課題解決型」，学生が各々自由にプログラムを作成もしくは改変する課題を「自由拡張型」と表記する．また，不正コピー疑い件数について，ソースコード A がソースコード B の不正コピーである場合と，ソースコード B がソースコード A の不正コピーである場合は区別される．これは，2 つのソースコードが同一の部分を含んでいても，同一でない部分の量によってオリジナリティが異なるためである．

4.3　結果

F 値は R と P の調和平均であり，F 値 $= \frac{1}{(1/R+1/P)/2}$ により求められる．今回の実験では，文献 [6] で行った実験をふまえ，F 値が最も高くなった $\lambda_t = 14$，$K = 2$ に固定して，$R \geq 0.80$ で最大となる P およびそのときの λ_c を求めた．結果を**表 2** に示す．課題 1 では 23%，課題 2 では 49%，課題 3 では 40%の P の向上が見られた．従来手法に比べ，今回提案した手法では，誤った検出が大幅に減少したことが分かる．特に，課題解決型である課題 2，3 では，大きく P が向上した．これは，入出力などのどうしても同じになってしまうプログラムが，TCM を使用する

表 3　1 クラスの全学生を対象とした不正コピー検出表
Table 3　Table of the result of illicit copy detection against all students in a class.

User	Assignment No 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A				EPbdjo		N/S	e								
B			D	Cl		D					NW	PRScl		D	D
C				EFGJLRZgikl						H					
D															
E				FJL								W			
F		K	bjl	KLS								Un	W		HSl
G				FKLS							JLTc	JLT	JLQT	d	c
H				KSYbcefghjo										W	Sl
I	DU	DU	BD	D	DUX	BD	FUWZci		U	CH	Uh		U	CDHWb	Uabg
J	Fc	F	c	FGKLS	c	T			c						
K													R		
L	Jc	X	BJTbce	KS	Jc	JT	J	J	T	J	J	J	J	J	J
M	B	B		CEFGJKLRSTYZbgiklo		BDIXn			B	B	BNWe	di	B	FY	Gc
N	FJ	H		CEFGJKLMRSTYZbdghiklo								F			
O	N/S														
P				E	Ge	BDIMXn	FIUWZcin		IUn	JL	GJLQTc	Rl	IUn	CU	IUabdgn
Q	JLc					JLT	JL		Jc	T	GJLTc		JLT	JLT	GJLMTZc
R				EFGJKLS						a		aglo			l
S				K											
T	FJNJLX		BJbce	CEFGJKLRSYZabcdgiklo			C	JLj			JL	JL	JL	JL	JLZ
U	D	D		SVXYc	D	BDeg	FWZci	In		CHIn	h			Cb	abg
V				Kc								FUn		o	
W															aghjk
X	jk			Vc	DU	BDI		j	j	IUjn		j	abejo		Y
Y	W			FGJKLRS											
Z				EFGJKLRSYal										a	JL
a				EGR								Sgl			bk
b		a	Ojl	FJLPgo									KRao	C	
c				EFJKLS			Wo	g			JLT	S	N		
d				Ea		e						EW		CU	Wagjk
e	b			bfo	G	Dg					BN		abo	FXi	FHS
f		d		o			Zl	cg					K		
g				EFGJKLRSYZaclo		D					h			Gd	abjk
h	N/S		N/S	FGJKLMRSTYZcfgiko					N/S			P			jk
i				EFGJKLRSYZbdglo							N/S			Xj	
j	k			Pafo				JL		IUn	IUhn			F	ak
k		HN		EFGJKLRSYZabgil		BDIMXn			Y					Sd	
l			j	EFGJLR								g			
m	c			EYadh							IUghjn		Y		
n	DIUDIU		BDI	DI	DIUX	BDIX	FIUWZci	I	IU	CHI	IUh	U	IU	f	IUabg
o	Fb			FJKLP	c							Sagl	KRa		ag

blank: No detection, N/S: No submission

ことによって除去されたからだと考えられる．しかしながら，λ_c が課題 1 と 2，3 では大きく異なっているため，課題ごとに適当な λ_c を決定するという課題は残る．

4.4　システムにおける不正コピー検出結果の活用

本節では，システムにおける不正コピー検出表を例示することで，3 章で述べた不正コピー検出がどのように機能するのかを説明する．**表 3** は，本システムにおいて教員のみが閲覧可能な不正コピー検出表のうち，1 クラスの学生 41 人が実際に提出した課題 15 問を抜粋して示したものである．なお，分かりやすさを重視し，ユーザ数と課題数を限定してユーザ間の不正コピー関係を一覧表とした．表 3 の縦軸はユーザ名，横軸は課題番号であり，表中にそれぞれのユーザの提出したプログラムが誰のコピーであるかを表示している．なお，本機能では提出時間が遅いものを不正コピーと見なす．図中の空白は不正コピーが検出されなかったことを，`N/S` は未提出を意味する．

表 3 の User I，U，n を横方向に見ると，複数回にわたって User D のソースコードのコピーと判定されており，常習的に User D から提供されたソースコードを提出している可能性が高いことが分かる．同様に User L，Q，T は User J から，User M は User B からソースコードの提供を受けている可能性が高い．他のユーザは，時々不正コピーとして検出されているが，頻度が低く，また同一人物からのコピーでないため，False-Positive である可能性が高い．また課題 4 は，教員が講義で例示したソースコードに少し変更を加えるだけで正解となる課題だったため，多くのユーザが相互に不正コピーと判定されており，大部分が False-Positive であると考えられる．

5.　システム構築

5.1　オンラインジャッジシステム

表 4 に開発環境を，**図 5** にシステムの概要を示す．一般的なオンラインジャッジシステムはユーザが提出したソースコードを受け取り，コンパイルして実行する．その過程で多くの文字列処理を行うため，本システムは文字列を簡単に処理できる Python で実装した．また，Web インタフェースを実装するために Python の Web フレームワークである Django を使用した．Django を使用することによって MVC モデルに則ってシステムを構築することができ，システムの変更・修正が容易になる．アカウントとパスワードの照会は，学内 LDAP（Lightweight Directory Access Protocol）サーバを用いる．既存の認証サーバを用いることで，新規登録やパスワード変更の手間を省いた．

本システムには，一般的なオンラインジャッジシステムが備える機能を実装する．本システムで扱うプログラミング言語は，本校の授業で使用されている C 言語とする．ユーザはシステムに学内用のアカウントとパスワードを入力してログインする．ユーザの権限は学生・教員・管理者に分けられている．

学生ユーザは，課題を閲覧・提出することができる．本システムでは，学生が提出したソースコードを保存し，次回学生が同じ課題を開いたときに再度表示する．これにより，学生はソースコードを保存しておく必要がなく，使用する端末を変更しても，続きから課題に取り組める．システムは提出されたプログラムをコンパイル・実行し，コンパイルエラーや正誤判定の結果を学生にフィードバックする．正誤判定は，境界値のチェックを含む複数の入力でテ

表 4　開発環境
Table 4　Environmental development.

構成要素	バージョン
開発言語	Python 2.7.1，Perl 5.10.1
Web フレームワーク	Django 1.3.1
OS	CentOS 6.2
データベース	PostgreSQL 8.4.11
Web サーバ	gunicorn 0.17.2
リバースプロキシサーバ	nginx 1.2.0
評価用 C コンパイラ	gcc 4.4.7

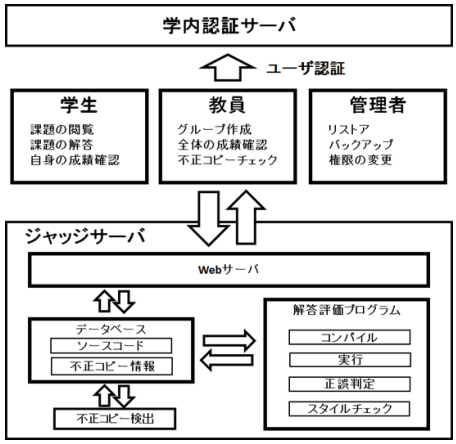


図 5　システム概要
Fig. 5　Outline of the proposed system.

ストしており，学生が自身でテストする場合と比べより厳密なものとなっている．さらに，自身の成績や，提出したソースコードを閲覧できるが，提出したソースコードの不正コピー判定結果は分からない．

教員ユーザは，システムに課題をアップロードできる．教員が**図 6** のようなファイルをアップロードすると，自動で**図 7** のような課題ページが作成される．また，自身が担当する講義を受講する学生のグループを作成し，成績を管理できる．さらに，学生が提出したソースコードや，過去に提出されたソースコードとの類似度を示す表を閲覧できる．

管理者ユーザは，教員ユーザや管理者ユーザの権限変更，システムのバックアップおよびリストアを行うことができる．

5.2　不正コピー検出

不正コピー検出プログラムは Perl で記述した．システムは，不正コピー検出プログラムを一定の時間間隔で起動する．不正コピー検出プログラムは提出されたソースコードが格納されたディレクトリを巡回し，新しいファイルを発見すると，他のユーザのソースコードと不正コピー率の計算を行う．不正コピー率があらかじめ定めたしきい値 λ_c を超えた場合，双方のユーザ ID と不正コピー率をデータベースに格納する．

```
<category>プログラミング基礎課題集</category>
<problem_title>hello world を出力せよ</problem_title>
<problem_text>整数 N を読み込み，N 回 hello world と出力するプログラムを書いてください．</problem_text>
<input_type>3</input_type>
<output_type>
hello world
hello world
hello world
</output_type>
<input_data1>1</input_data1>
<output_data1>hello world</output_data1>
```

図 6　課題ファイルの例
Fig. 6　Example of a assignment file.

図 7　システム出力の結果
Fig. 7　Result of the system output.

本システムでは，提案する手法による比較結果のほか，他の学生のプログラムを変更せずに提出した場合，ユーザ定義名および定数値のみを変更して提出した場合にそれぞれ有効な手法による判定結果を追加した（以下，判定 1，判定 2 と記す）．判定 1 は，判定したい 2 つのプログラムのトークン列について，編集距離が最も少なくなるように改変したものとして同一トークン列を特定し，不正コピー率の算出を行う．判定 2 は判定 1 に加えてトークン列中のユーザ定義名および定数値を抽象化して比較する．判定 1，判定 2 の不正コピー率はほとんど手を加えていないプログラムでは高くなり，偶然類似した可能性のあるプログラムでは低くなる．これにより，ほとんど手を加えていないプログラムと，偶然類似した可能性のあるプログラムを区別できる．

なお，現在のシステムでは，課題はすべて課題解決型であり，表 2 より課題解決型で有効だった $\lambda_c = 0.72$ を使用している．しかしながら，表 3 の課題 4 に見られるように，課題によっては False-Positive が多数発生しており，状況に応じた λ_c の調整が必要である．そこで，今後，課題ごとに λ_c を変化させられるよう改良する予定である．

また，学生は不正コピー検出結果を閲覧することはできないため，教員が伝えない限りは検出結果を知ることはない．また教員は複数課題の不正コピー検出状況，学生間の

不正コピー関係などを総合的に判断して不正コピー常習者を特定するため，一度の False-Positive により不正コピーをしていない学生を誤って指導することはない．検出結果を直接学生に開示せず，教員経由の指導とすることで，不正コピーをしていないにもかかわらず検出された学生がシステムに対し不信感を持つことを防いでいる．

6. アンケート

熊本高等専門学校八代キャンパス 3 年次（2013 年度）の学生を対象に，2013 年 6 月（本システムを 3 カ月間使用後，回答者 126 名．以下アンケート 1 と記す），2014 年 1 月（本システムを 10 カ月間使用後，回答者 122 名．以下アンケート 2 と記す）の 2 回に分けてアンケートを実施した．アンケート 1 では本システムについて，アンケート 2 では本システムの利用による学習成果について調査した．

アンケート 1 の内容を**図 8**，結果を**図 9** に示す．アンケートは，「思う，少し思う，あまり思わない，思わない」の 4 段階の選択式である．100%の学生が，本システムのいずれかの機能について有用であると回答し，そのうち 91%の学生はすべての設問で機能が有用であると答えた．また，81%の学生が，不正コピー検出機能は不正行為の抑止力になると考えていることが分かった．なお，アンケート 1 の詳細については文献 [4] が詳しい．

(1) 課題を Web 上で提出できる機能は有用だと思いますか
(2) 提出したソースコードの正誤がすぐに分かる機能は有用だと思いますか
(3) プログラムの書き方を教えてくれる機能（コーディングスタイルチェック機能）は有用だと思いますか
(4) コピーされたソースコードを検出する機能（不正コピー検出機能）により，理解していないプログラムを提出する人が減ると思いますか
(5) 本システムを利用することで，以前に比べプログラムが書けるようになる（なった）と思いますか
(6) 本システムを利用した感想を教えてください（自由記述）

図 8　提案システムに関するアンケート

Fig. 8　Questions in the questionnaire for the proposed system.

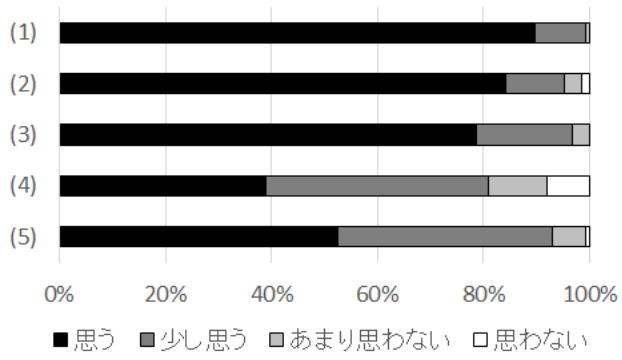


図 9　提案システムに関するアンケート結果

Fig. 9　Result of answers from the questionnaire for the our system.

アンケート 2 の内容を**図 10**，結果を**図 11** に示す．アンケートは基本的に「思う，少し思う，あまり思わない，思わない」4 段階の選択式で，(5) および (6) のみ，「100 時間以上，50–100 時間，10–50 時間，10 時間未満」の 4 段階から選択する．

(1) より，学生は授業内容を難しいと感じているが，(2), (3) から，自分で解こうという姿勢が感じられる．

(4) より，89%の学生が，課題に取り組んだことでプログラムを書けるようになったと回答している．これは，本システムを講義で使用したことで，学生に多くの課題を解かせると同時に不正コピーを抑止でき，学生たちが自力で多数のプログラム課題を解くという状況を作ることができたためだと考える．また，定期試験の成績をもとに，プログラムの記述力が向上したといえるかどうか，以下の方法で調査した．まず，年 4 回実施される定期試験において，1 回目（6 月実施）と 4 回目（2 月実施）の試験におけるプログラム記述問題（100 点満点）の得点のうち，1 回目の成績が 30 点以下だった学生グループに注目した．グループを，さらに，提出した課題のうち 8 割以上が不正コピーと判定されたグループ（5 人，平均 4.4 問提出）と 8 割未満のグループ（29 人，平均 24.1 問提出）に分け，それぞれのグループにおける得点の伸びの平均に有意な差があるか検証した．結果，前者のグループは平均 22.4 点（標準偏差

(1) 授業の内容は難しかったと思いますか
(2) プログラムを暗記するのではなく，理解するよう努力していると思いますか
(3) 課題を友人の力を借りずに解いたと思いますか
(4) 課題に取り組んだことでプログラムを書けるようになったと思いますか
(5) 通常の予習・復習・定期試験の勉強時間はどの程度ですか（課題は含まない）
(6) 課題に要した時間はどの程度ですか
(7) 本システムを 1 年間利用した感想を教えてください（自由記述）

図 10　学習成果に関するアンケート

Fig. 10　Questions in the questionnaire items on learning achievements.

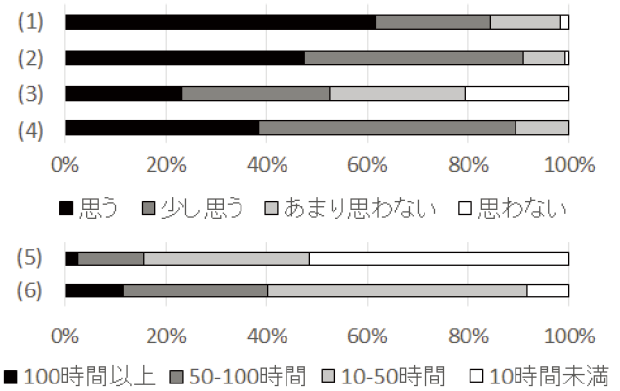


図 11　学習成果に関するアンケート結果

Fig. 11　Result of answers from the questionnaire on learning achievements.

14.9 点），後者のグループでは平均 46.0 点（標準偏差 21.9 点）の得点の向上が見られ，有意水準 1%で有意差検定を行った結果，有意な差が認められた．

さらに，(5) および (6) について，課題以外の学習時間が 10 時間未満の学生は 51%（63 人）だったのに対し，課題にかけた時間が 10 時間未満の学生は 8.2%（10 人）であった．よって，多くの課題（2013 年度は 50 問）を課することにより，自主的な学習の習慣がない学生でも，学習時間を確保できることが分かる．なお，課題にかけた時間が 10 時間未満の学生 10 人のうち 7 人は，4 回の定期試験の平均点が全学生の平均以上であることから，もともとプログラミングが得意な学生で，学習時間を多くとる必要がなかったと考えられる．

また自由記述では，「学習する意欲が上がった」，「自分でプログラムを書こうと努力できたのでよかった」，「大変だったが，勉強になった」，「だんだん解けるようになっていることが実感できた」，「自分で考えて解けたとき，達成感が得られた」，「難しい問題もあったが，解けたとき非常に嬉しかった」，「プログラムの実力がついた」といった肯定的な意見が 51%の学生からあげられた．一方で，「難しかった」，「課題が多く大変だった」，「不正コピー判定が怖かった」などの否定的な意見も 36%見られた．

また実際に講義で本システムを利用している教員からは，本システムを導入したことで採点の負担が軽減され，当初予定していたよりはるかに多い課題に取り組ませることができた，との意見があった．その背景として，これまでは 1 人で 120 人分のレポートを採点し，全員に 5〜10 分の口頭試問をしていたため，4，5 問程度の課題でも多大な労力を要していた．しかし，本システムを利用することにより，従来教員が目視で行っていたプログラムの動作チェックが自動化され，不正コピー常習者を重点的に指導できるようになった．教員の採点の負担が大きく軽減された結果，利用を開始した 2012 年度後期は 18 問，2013 年度は前期 25 問，後期 25 問と，学生に多くの課題を課することができた．

7. まとめ

本研究では，学生がプログラミングの多くの課題に取り組み，プログラミング技術を向上させることを目的としたオンラインジャッジシステムの開発を行った．

本システムでは不正コピー検出機能により，学生の不正行為の抑制を図っている．本稿で提案した不正コピー検出手法では，3 つのデータセットでの検証において，従来の検出手法と比べ P が向上している．特に，課題解決型である課題 2，3 では，それぞれ 49%，40%の向上が見られた．しかしながら，課題 1 と 2，3 では不正コピーと判定する不正コピー率のしきい値が大きく異なっているため，課題ごとに適当な値を決定するという課題が残る．しかし，本システムを構築する以前は，教員の記憶や勘に頼った検出しかできなかったのに対し，本システムを使用することにより不正コピー常習者を特定できるようになり，それらの学生に対し重点的に指導を行うことが可能になった．また，不正コピーの検出精度向上による抑止効果については，学生は不正コピー検出結果を見ることができないため，抑止効果との直接の関わりはないものの，検出精度の向上により教員の不正コピー常習者の発見の精度が上がり，よりきめ細かな指導が可能になると考える．

また，本システムを使用した学生にアンケートを実施したところ，本システムでの課題提出について，ほとんどの学生が従来の提出方式より利便性が高いと回答した．学習成果の面からも，課題に積極的に取り組んだ学生はプログラムを書けるようになっていると回答している．また本システムを使用し多くの課題を課することにより，自主学習の習慣がない学生の学習時間を確保できることも分かった．また自由記述から，多くの学生が難しい課題を解くことへの楽しさや達成感を得られていることも確認できた．以上より，本システムの学習用システムとしての有用性を検証することができた．

なお，本システムでは正誤判定機能，コーディングスタイルチェック機能，不正コピー検出機能の 3 つの機能を実装しているが，機能別の評価は行っていない．また，不正コピー検出機能を実装することで，不正コピー行為がどの程度減少したのかについては，アンケート結果のみで実際には評価できていないため，これらは今後の課題とする．

参考文献

[1] Baker, B.S.: On finding duplication and near-duplication in large software system, *WCRE*, IEEE Computer Society, pp.86–95 (1995).

[2] Baxter, I.D., Yahin, A., Moura, L., Anna, M.S. and Bier, L.: Clone Detection Using Abstract Syntax Trees (1998).

[3] Bellon, S., Koschke, R., Antoniol, G., Krinke, J. and Merlo, E.: Comparison and Evaluation of Clone Detection Tools, *IEEE Trans. Softw. Eng.*, Vol.33, pp.577–591 (2007).

[4] Iwamoto, M., Nakamura, M. and Oshima, S.：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，工学教育 (2014).

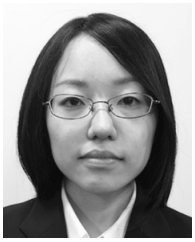
[5] Iwamoto, M., Oshima, S. and Nakashima, T.: Token-based Code Clone Detection Technique in a Student's Programming Exercise, *BWCCA*, IEEE, pp.650–655 (2012).

[6] Iwamoto, M., Oshima, S. and Nakashima, T.: A Token-based Illicit Copy Detection Method Using Complexity for a Program Exercise., *BWCCA*, IEEE, pp.575–580 (2013).

[7] Kamiya, T., Kusumoto, S. and Inoue, K.: CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code, *IEEE Trans. Softw. Eng.*, Vol.28, pp.654–670 (2002).

[8] Krinke, J.: Identifying Similar Code with Program Dependence Graphs, *8th Working Conference On Reverse Engineering*, Stuttgart, Germany, IEEE, pp.301–309

(2001).
[9] Mayrand, J., Leblanc, C. and Merlo, E.M.: Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics, *ICSM*, IEEE Computer Society, pp.244–253 (1996).
[10] McClure, C.: *The three Rs of software automation: Re-engineering, repository, reusability*, Prentice Hall (1992).
[11] McConnell, S.: *Code Complete*, Microsoft Press (1993).
[12] Navarro, G.: A guided tour to approximate string matching, *ACM Computer Surveys* (*CSUR*), Vol.33, No.1, pp.31–88 (2001).
[13] The University of Aizu: Aizu Online Judge, available from ⟨http://judge.u-aizu.ac.jp/onlinejudge/⟩ (2015).
[14] Topcoder, Inc.: topcoder, available from ⟨http://www.topcoder.com/⟩ (2015).
[15] Yang, W.: Identifying Syntactic Differences Between Two Programs, *Software — Practice and Experience*, Vol.21, pp.739–755 (1991).
[16] 岡原　聖，真鍋雄貴，山内寛己，門田暁人，松本健一：ソースコード流用のコードクローンメトリクスに基づく検出手法（ソフトウェア解析），電子情報通信学会技術研究報告，KBSE，知能ソフトウェア工学，Vol.109, No.307, pp.73–78 (2009).
[17] 肥後芳樹，楠本真二，井上克郎：コードクローン検出とその関連技術，電子情報通信学会論文誌 D，Vol.J91-D, No.6, pp.1465–1481 (2008).



岩本 舞　（正会員）

2013 年熊本高等専門学校情報電子工学科卒業．同年熊本高等専門学校八代キャンパス技術・教育支援センター所属．主としてコードクローン研究に従事．

中村 真人

2013 年熊本高等専門学校生産システム工学専攻情報システムコース修了．2015 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻修了．現在，株式会社リクルートホールディングス所属．ソフトウェアエンジニアとしてソフトウェア開発に従事．

小島 俊輔　（正会員）

1991 年熊本大学工学部電気情報工学科卒業．1993 年同大学院修士課程修了．同年八代高専情報電子工学科助手．2003 年同校助教授．現在，熊本高専 ICT 活用学習支援センター教授．主としてネットワークセキュリティに関する研究に従事．電子情報通信学会，ACM 各会員．博士（工学）．

中嶋 卓雄　（正会員）

1986 年熊本大学院修士課程修了．富士通（株），熊本大学工学部を経て，2001 年九州東海大応用情報学部．現在，東海大学九州キャンパス長（学長補佐）．自己相似性を持つネットワークの評価，システムパフォーマンスの評価，ネットワークセキュリティ等の研究に従事．情報処理学会山下記念研究賞（2006）受賞．ACM，IEEE-CS 各会員．博士（工学）．