情報処理学会論文誌 Vol.54 No.1 330–339 (Jan. 2013)

プログラミング演習における進捗状況把握のためのコーディング過程可視化システムC3PVの提案

井垣 宏[1,a)] 齊藤 俊[2,b)] 井上 亮文[2,c)] 中村 亮太[3,d)] 楠本 真二[1,e)]

受付日 2012年4月20日, 採録日 2012年10月10日

概要：我々は受講生のプログラミング演習時におけるコーディング過程を記録し，可視化して講師に提示するシステム C3PV を提案する．本システムは，ウェブ上で動作するオンラインエディタとコーディング過程ビューから構成されている．オンラインエディタは受講生のコーディングプロセスにおける，文字入力，コンパイル，実行，提出といったすべての行動を記録する．コーディング過程ビューは課題の進み具合いや受講者の相対的な進捗遅れを可視化して講師に提示する．講師はあるエラーに関して長時間悩んでいる受講生や全体の進捗と比較して遅れている受講生を C3PV によって確認し，個別指導といった支援につなげることができる．本研究では実際に C3PV を学部 1 年生が受講する Java プログラミング演習に適用し，C3PV によって可視化されたコーディング過程を利用した受講生対応を行った．その結果，コーディング過程ビューに基づいて対応した 45 件中 38 件（約 84%）において，実際に受講生がサポートを必要としていたことが確認できた．

キーワード：プログラミング演習，コーディング，進捗管理，ソフトウェア工学教育

Programming Process Visualization for Supporting Students in Programming Exercise

Hiroshi Igaki[1,a)] Shun Saito[2,b)] Akifumi Inoue[2,c)] Ryota Nakamura[3,d)] Shinji Kusumoto[1,e)]

Received: April 20, 2012, Accepted: October 10, 2012

Abstract: In this paper, we propose a coding process visualizer for programming practice. Our system named “C3PV” consists of an online editor and a coding process viewer. The online editor can collect all coding process performed by each student. The coding process viewer can visualize whether the students are doing well about their programming exercises. This viewer enables a lecturer and teaching assistants to identify the students who are falling behind the rest of the class. We confirmed that the system could determine whether the student needed our help with 84 percent accuracy by some practical experiments.

Keywords: programming exercise, coding, progress management, software engineering education

1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
2 東京工科大学コンピュータサイエンス学部
School of Compute Science, Tokyo University of Technology, Hachioji, Tokyo 192–0982, Japan
3 東京工科大学片柳研究所
Katayanagi Institute, Tokyo University of Technology, Hachioji, Tokyo 192–0982, Japan
a) igaki@ist.osaka-u.ac.jp
b) syuns@hil.cs.teu.ac.jp
c) akifumi@cs.teu.ac.jp
d) rnakamura@media.teu.ac.jp
e) kusumoto@ist.osaka-u.ac.jp

1. はじめに

日常で利用されるソフトウェアおよびソフトウェアが組み込まれた製品の数や種類の増大にともない，優秀なソフトウェア開発者の育成が急務となりつつある．そのため，受講生が与えられた課題に基づいてコーディングを実際に行うプログラミング演習と呼ばれる形式の授業が多くの教育機関で行われるようになっている [2]．

通常プログラミング演習は，開始時に講師が受講生に課題を与える．受講生は必要に応じて講師に質問をしつつ，

与えられた課題のコーディングを行う．コーディングが終了した後，受講生は講師にソースコードを提出し，チェックを受ける．

このようにプログラミング演習では各受講生が個別に課題に取り組むため，受講生間のスキル差によって進捗状況に大きな差が生まれることが多い．実際に，講義や演習についていけてなくても質問をせず，順調に課題を進めている受講生と比較して遅延したり停滞したりする受講生も数多く存在する [5]．そのため，講師は受講生による能動的な問合せに対応するだけでなく，理解不足や諦め等，多種の要因に起因するコーディングの遅延や停滞状態にある受講生を把握し，状況に応じた指導を行うことが重要となる [11]．特に初学者を対象としたプログラミング演習の場合，理解不足等の状況を早期に発見することは非常に重要である．

しかしながらプログラミング演習では，受講生の数に比して少数の講師や TA（Teaching Assistant）が対応するため，受講生のコーディング状況の把握が困難な場合も多い．プログラミング演習における受講生の状況把握を目的とした既存研究は数多く行われているが，そのほとんどが課題提出後にその内容を分析するものであるため，演習中に問題のある受講生を把握することを想定していない [4], [10], [12]．

そこで本研究では，受講生のコーティング過程を分析し，コーディング遅延や停滞を可視化する C3PV（Coding Process Visualizer in Programming Practice）を提案する．C3PV は受講生ごとに，総 LOC（Lines Of Code），課題ごとのコーディング時間，単位時間あたりのエディタ操作数，課題ごとのエラー継続時間の 4 種類のメトリクスをリアルタイムに計測し，ランキング等の可視化処理を行い，講師に提示する．C3PV が相対的に状況が悪い受講生をサポートが必要な受講生として提示することで，限られた数の講師や TA でも容易に受講生の学習状況を把握し，その状況に即した指導を行うことが可能となる．

本論文の構成を以下に示す．2 章では準備としてプログラミング演習の説明と従来研究によるプログラミング演習のための教育支援システムの課題，3 章では提案手法，4 章では実装方法，5 章では評価について述べ，6 章をまとめとする．

2. プログラミング演習の特徴と問題点

2.1 プログラミング演習

プログラミング演習とは，講師がある単元について解説を行った後，受講生がその内容に応じた課題を独力で解く形式の授業である．

プログラミング演習において，各受講生は与えられた課題のコーディングを行い，作成したソースコードの正しさをコンパイルや実行によって確認後，そのソースコードを提出する．本論文では，受講生によるソースコードの編集から課題提出までの一連の行為を**コーディング過程**と呼ぶ．講師はコーディング過程における受講生からの質問への対応や，提出された課題のチェックによるフィードバックを行うことで受講生のコーディングスキル育成を目指す．

2.2 初学者を対象としたプログラミング演習における課題

近年，プログラミング演習における学業不振者等の増加が問題になっている．プログラミング演習では受講生自身が自律的にコーディングを行うことが求められる．そのため，自身で問題解決が困難である初学者にとって，コーディング中のトラブルや理解不足からくる行き詰まり状態の継続は，コーディングそのものに対するモチベーションを悪化させてしまう可能性がある．そのため，このような受講生を早い段階で見つけ出し，講師によるアドバイス等の具体的な支援を行うことが望まれている [9]．

通常，講師は受講生からの質問に対応するときや提出されたソースコードを確認するときに，受講生ごとのコーディング進捗度合いや理解度に関する情報を獲得する．しかしながら，講師や TA（ティーチング・アシスタント）の数が限られていることから，理解度に問題のある受講生を演習時間内に早期に効率良く発見し，対応することは非常に困難である．

そこで本研究では，下記要求に対応したプログラミング演習のための教育支援システムを提案する．

R1 開発環境のセットアップが容易で講師・受講生の負荷が小さい．

R2 多様な遅延や停滞状況の検知に対応したコーディング過程の可視化．

R3 コーディング対象の言語に非依存．

コーディング過程において，受講生はエディタを用いてコーディングを進める．そのための開発環境はセットアップが簡単で，講師および受講生が容易に利用できることが望ましい．また，コーディング過程における受講生の状況を早期に把握するためには，課題ごとの提出時刻や提出されたソースコードといったコーディング過程終了後を対象とした分析ではなく，単位時間ごとの実装行数（LOC）やコンパイルエラーの情報といったコーディング過程内を対象とした分析・メトリクスの可視化が必要となる．さらに，多様な言語への対応を考慮すると，分析・可視化の内容が特定の言語に依存しないことが望ましい．

以上をふまえて，次節では我々の提案するコーディング過程可視化システムについて詳述する．

3. コーディング過程可視化システム：C3PV

3.1 キーアイデア

我々は受講生のコーディング過程を記録し，相対的な遅延状況を可視化して講師に提示する教育支援システム

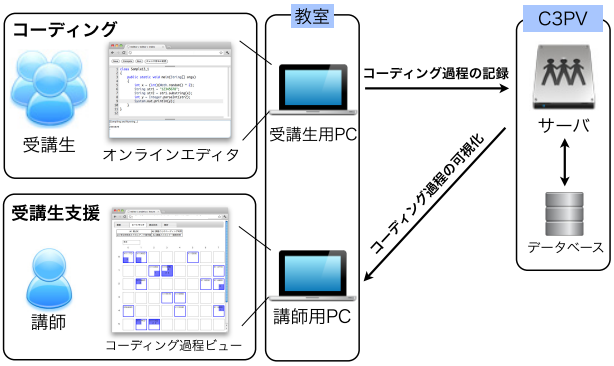


図 1 コーディング過程可視化システム C3PV の概要
Fig. 1 Coding process visualizer in programming practice (C3PV).

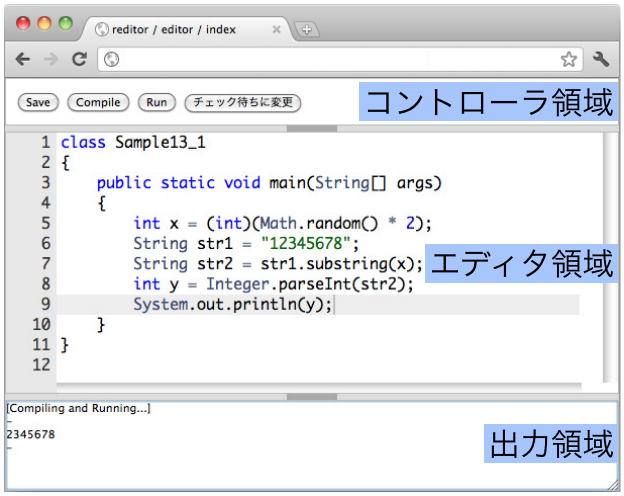


図 2 オンラインエディタを用いた Java プログラムのコーディング
Fig. 2 Coding example of Java with C3PV's online editor.

C3PV を提案する．**図 1** にシステム概要を示す．C3PV は受講生にオンラインエディタによる開発環境を提供する．受講生のコーディング過程は，オンラインエディタを通じて記録される．記録されたコーディング課程はコーディング過程ビューを通じて可視化され，講師に提示される．

C3PV におけるオンラインエディタとコーディング過程ビューは受講生および講師の PC 環境にブラウザが入っていれば利用できるため，環境のセットアップは非常に容易である．C3PV は受講生のオンラインエディタ上のコーディング過程から各種のメトリクスを収集する．その結果をコーディング過程ビューに可視化することで，受講生ごとの多様な遅延状況に講師が対応できるようになる．

以降では，C3PV の各構成要素と収集・可視化する 4 種類のメトリクスについて述べる．

3.2 オンラインエディタを用いたコーディング過程の記録

受講生は**図 2** に示すオンラインエディタにブラウザからアクセスし，コーディングを行う．オンラインエディタはコントローラ領域とエディタ領域，出力領域から構成されている．受講生はエディタ領域にソースコードを記述し，Save（保存），Compile（コンパイル），Run（実行），チェック待ちに変更（課題提出）といった操作をコントローラ領域で選択する．受講生が Compile あるいは Run を選択すると，出力領域にコンパイル結果あるいは実行結果が表示される．エディタ領域では，Ace [1] と呼ばれるオープンソースのコードエディタを利用することで，インデントやハイライトといったエディタに求められる基本的な機能が実現されている．C3PV はこのオンラインエディタを通じて，受講生 1 人 1 人のコーディング過程を記録する．次に，C3PV が収集するコーディング課程におけるログの種類について述べる．

表 1 エディタ領域ログの種別
Table 1 Categories of C3PV's editor area log.

ログ種別名	内容
insertText	1 行以内の文字を入力
removeText	1 行以内の文字を削除
insertLines	複数行の文字列を入力
removeLines	複数行の文字列を削除

表 2 エディタ領域ログの例
Table 2 Examples of editor area log.

ID	時刻	文字列	ログ種別	開始行	開始列	終端行	終端列
1	2012-2-1 15:22.476	int x = 4; int y = 3;	insertLines	1	0	3	0
2	2012-2-1 15:31.793	3	removeText	2	8	2	9
3	2012-2-1 15:32.404	5	insertText	2	8	2	9
4	2012-2-1 15:39.643	int x = 4; int y = 5;	removeLines	1	0	3	0

3.2.1 エディタ領域ログ

C3PV は受講生がエディタ領域において行う操作とその時刻をエディタ領域ログとして記録する．ここで収集されるログの種別を**表 1** に示す．エディタ領域において受講生はアルファベットの文字入力や削除，コピー&ペースト，カット&ペーストを行う．C3PV は受講生による文字（列）の入力/削除を検知し，エディタ領域ログとして記録する．**表 2** に実際に C3PV が収集したエディタ領域ログの例を示す．ここで ID はログの主キーを示し，ログイン ID は受講生を示している．このエディタ領域ログからは受講生が初めに int 型の変数 x，y を初期化した 2 行の文を貼り付け（insertLines），y の値を 3 から 5 に書き換えた後（removeText と insertText），その 2 行の文を切り取っている（removeLines）ことを確認できる．

3.2.2 コントローラ領域ログ

C3PV は受講生がコントローラ領域において行う操作とその時刻をコントローラ領域ログとして記録する．ここで収集されるログの種別を**表 3** に示す．autosave のみ C3PV による自動保存を表し，それ以外はすべて受講生の行った振舞いとして記録される．自動保存は newopen もしくは open が記録されてから close が記録されるまでの間，1 分

おきに呼び出される．また，close は受講生によってエディタのページが閉じられたり，別のページへ遷移したりすることによって表示されなくなったときに記録される．

表 4 に C3PV が記録するコントローラ領域ログの例を示す．この例では講師が「hello, world!」と表示する Java の課題を課しているものとする．この例では，初めに，ログイン ID syuns が課題をエディタで開き（newopen），その課題のコーディング中に C3PV が自動的にソースコードの保存を行った（autosave）．次に，syuns は課題を完成させ [Compile] ボタンをクリックしてコンパイルを行った（compile）．ここでエラーが発生したため，エラーメッセージをもとにソースコードを修正し [Run] ボタンをクリックしてコンパイルと同時に実行した（run）．syuns は出力を確認後，ソースコードを [Save] ボタンをクリックして保存（save），[チェック待ちに変更] ボタンをクリックしてソースコードを提出後（submit）した．最後に，次の課題を選択するためにブラウザの戻るボタンをクリックし，エディタ画面を閉じた（close）．

3.3 コーディング過程メトリクス

受講生のコーディング状況可視化を目的として，C3PV はエディタ領域ログとコントローラ領域ログを利用して，以下に示す 4 種類のメトリクスを計測する．

M1：総 LOC
M2：課題ごとのコーディング時間
M3：単位時間あたりのエディタ操作数
M4：課題ごとのエラー継続時間

表 3 コントローラ領域ログの種別
Table 3 Categories of C3PV's controller area log.

ログ種別名	内容
newopen	対象の課題が初めてエディタで開かれた
open	newopen された対象課題が再度エディタで開かれた
save	[Save] ボタンがクリックされた
autosave	C3PV がソースコードを自動保存した
close	ブラウザを閉じる等して受講生がエディタから離れた
compile	[Compile] ボタンがクリックされた
run	[Run] ボタンがクリックされた
submit	[チェック待ちに変更] ボタンがクリックされた

M1：総 LOC

総 LOC は，受講生がプログラミング演習中にエディタ領域に入力したソースコードの総行数を表す．プログラミング演習において講師は複数の課題を受講生に課する．通常，課題ごとのプログラム行数は異なることが多いため，どの受講生が他の受講生と比べて遅れているかを課題ごとの行数で検知することは困難である．また，受講生ごとに解き始めた課題が異なるような事例では，課題ごとの行数を比較することができない．そこで我々は，総 LOC として演習開始時から受講生によってコーディングされた課題の合計行数を計測する．総 LOC を受講生同士で比較することで，進捗が相対的に遅れている受講生を検知することが可能となる．ただし，欠席等で課題を 1 つも開いていない受講生は検知の対象としない．

M2：課題ごとのコーディング時間

課題ごとのコーディング時間は，受講生がコーディングに取り組む課題別の時間を表す．コントローラ領域ログとして newopen もしくは open が検知されてから，submit が行われるまでの時間を計測することで，M2 がより長い場合を特定課題のコーディングに行き詰まっている受講生として検知できる．ただし，いずれかの課題をエディタで開いており，かつその課題の提出が終わっていない受講生を対象とする．

M3：単位時間あたりのエディタ操作数

単位時間あたりのエディタ操作数は，受講生が一定時間内に文字入力/削除やコピー&ペーストを行った課題別の数を表す．表 2 に示すエディタ領域ログの 1 レコードを 1 ステップとして計測することで，M3 がより少ない場合をエディタの操作をせず考え込んでいたり別のことをしていたりする受講生として検知できる．ただし，いずれかの課題をエディタで開いており，かつその課題の提出が終わっていない受講生を対象とする．たとえば，次の (1)〜(4) のエディタ操作数は 4 である．

(1) エディタに「a」を打つ．
(2) エディタの「a」を消す．
(3) エディタにクリップボードの「class」を貼り付ける．
(4) エディタの「class」を選択して消す．

表 4 コントローラ領域ログの例
Table 4 Examples of controller area log.

ID	ログイン ID	ログ種別	時刻	言語名	ソースコード	出力	エラー
1	syuns	newopen	2012-2-1 15:30:00	Java			
2	syuns	autosave	2012-2-1 15:31:00	Java	3: ...world!")		
3	syuns	compile	2012-2-1 15:31:05	Java	3: ...world!")		3: ';' expected
4	syuns	run	2012-2-1 15:31:20	Java	3: ...world!");	hello, world!	
5	syuns	save	2012-2-1 15:31:25	Java	3: ...world!");		
6	syuns	submit	2012-2-1 15:31:30	Java	3: ...world!");		
7	syuns	close	2012-2-1 15:31:35	Java	3: ...world!");		

M4：課題ごとのエラー継続時間

課題ごとのエラー継続時間は，受講生がコンパイル/実行時にエラーが発生してから以降のコンパイル/実行によってエラーが発生しなくなるまでの時間を表す．エラー継続時間を計測することで，M4 がより長い場合をエラーが解決できない受講生として検知できる．ただし，いずれかの課題をエディタで開いており，かつその課題の提出が終わっていない受講生を対象とする．

3.4 コーディング過程ビューによる可視化

C3PV はコーディング過程ビューを用いて，計測したメトリクスの可視化を行う．コーディング過程ビューはメトリクスランキングおよびシートマップから構成される．メトリクスランキングでは M1～M4 の各コーディング過程メトリクスがソート可能なリストとして表示される．**図 3** にメトリクスごとのランキングテーブルを示す．図 3 (a) は受講生のログイン ID，受講生の座席情報および総 LOC を表示している．図 3 (b) は課題名とその課題のコーディング時間が表示されており，課題名で並べ替えることで，任意の課題において相対的に長時間取り組んでいる受講生が誰であるかを講師が確認できる．図 3 (c) では，5 分あたりのエディタ操作数ランキングが表示されている．なお，単位時間は 5，10，15 で切り替えられるようになっている．図 3 (d) はエラー継続時間（秒）を課題ごとに提示している．

ログインID	行列	総LOC
X11345D	5-1	61
X11105D	5-2	143
X11489D	2-1	175
X11003D	4-4	184
X11477D	1-7	189

(a)M1:総LOCランキング

課題名	ログインID	行列	課題ごとのコーディング時間(s)	アクティブ(0/1)
Report13_2	X11333D	4-7	2390	1
Report13_2	X11105D	5-2	1584	1
Report13_2	X11309D	1-3	1051	1
Report13_2	X11561D	3-3	1041	1
Report13_2	X06369D	4-0	1022	1

(b)M2:課題ごとのコーディング時間ランキング

課題名	ログインID	行列	単位時間あたりのエディタ操作数	アクティブ(0/1)
Report13_2	X11069D	0-1	4	1
Report13_2	X11177D	0-0	14	1
Report13_2	X11309D	1-3	40	1
Report13_2	X11465D	2-7	70	1
Report13_2	X11333D	4-7	79	1

(c)M3:単位時間あたりのエディタ操作数ランキング

課題名	ログインID	行列	課題ごとのエラー継続時間(s)	アクティブ(0/1)
Report13_2	X11045D	1-2	434	1
Report13_2	X11465D	2-7	426	1
Report13_2	X11309D	1-3	297	1
Report13_2	X11333D	4-7	252	1
Report13_2	X11105D	5-2	17	1

(d)M4:課題ごとのエラー継続時間ランキング

図 3 メトリクスランキングの表示例
Fig. 3 Example of metrics ranking view.

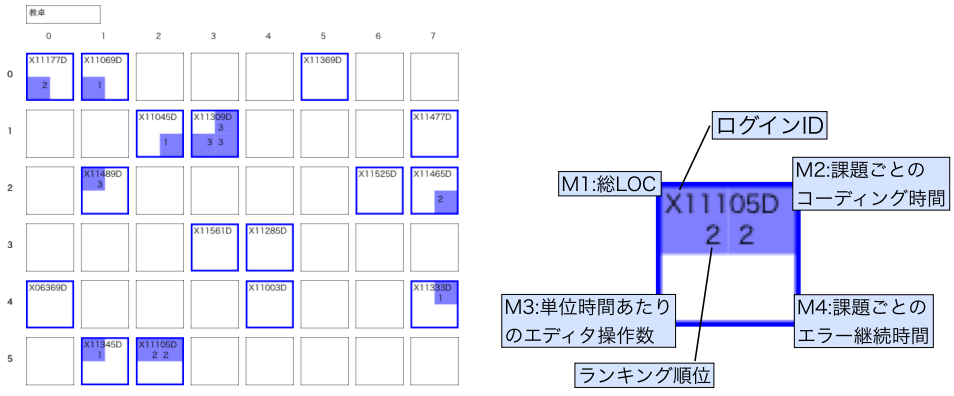
シートマップはメトリクスランキングにおける上位，すなわち何らかの基準で相対的に遅延している受講生を実際に座っている座席情報に基づいて強調表示するビューである．**図 4** (a) が実際に表示されるシートマップである．枠が太線で囲まれているマス目は受講生が実際に座っている座席を示しており，ログイン ID が表示される．各マス目は図 4 (b) に示すように 4 分割されており，M1～M4 の各メトリクスにおいて上位に入る受講生の対応するマス目内領域が青く表示される．この例では，ログイン ID が X11105D の受講生が M1，M2 の 2 種類のメトリクスにおいてランキング 2 位であることが分かる．すなわち，総 LOC が全受講生の中で 2 番目に少なく，かつ現在取り組んでいる課題に全受講生の中で 2 番目に時間をかけているということが判断できる．

このように講師はシートマップを見ることで，どの席に座っている受講生が相対的に遅延しているかを M1～M4 のメトリクスベースで判断することができる．

4. 実装

4.1 実装環境

図 1 に示す C3PV のサーバおよびデータベースは



(a) シートマップによる M1～M4 の可視化の様子　(b) M1～M4 とシートの対応関係

図 4 シートマップによる遅れの可視化
Fig. 4 Visualization example of sheetmap view.

edubaseCloud [7] の仮想マシン内に実装されている．ここで我々が利用した仮想マシンには edubaseCloud の機能を利用し，Xeon 2.4 GHz CPU 8 コア，メモリ 24 GB が割り当てられている．OS は CentOS5.6 で，ウェブサーバおよび DBMS はそれぞれ Apache/2.2.3，MySQL/5.0.77 を用いた．実装言語および実装フレームワークは，サーバサイドが PHP/5.3.3 および ZendFramework/1.11.0，クライアントサイドが JavaScript，jQuery/1.7.0 となっており，オンラインエディタ部分には Ace/0.2.0 を利用して開発を行った．

ここで edubaseCloud とは国立情報学研究所によって運用されている IaaS と呼ばれる種類のクラウドサービスである．一般に IaaS は，OS をインストールするための CPU やメモリからなるコンピュータ基盤をサービスとして提供する．本実装においても edubaseCloud 上に C3PV を配置することで，ハードウェアを新たに用意することなく安価にシステムを構築することが可能となった．また，OS が利用する CPU のコア数やメモリといったリソースの変更や OS 単位でのシステムの複製が可能であるため，多人数で利用した際の C3PV にかかる負荷を非常に容易に分散させることが可能となった．

4.2 Ideone サービスを用いたコンパイルおよび実行

受講生がコーディングしたソースコードのコンパイルおよび実行には，Ideone.com [3] のウェブ API を使用した．Ideone は 40 種類以上のプログラミング言語に対応したオンラインコンパイラである．SOAP 形式での呼び出しに対応しており，ソースコードを送信することで，コンパイルおよび実行結果を受け取ることができる．

4.3 C3PV を用いたプログラミング演習の流れ

ユーザは C3PV を利用する際，講義開始時に配布されるログイン ID とパスワードを使ってログインする．C3PV は，ユーザの役割を未ログインユーザ，受講生，講師の 3 つに分類することにより，各ページへのアクセスを制限している．

受講生は各自のブラウザにより C3PV にログインし，着席処理を行う．ここで着席処理とは，受講生が現在座っている教室内における座席の行番号と列番号を C3PV に登録することを指す．ログインおよび着席処理が終了すると，受講生はブラウザを通じて C3PV が提供するオンラインエディタにアクセスできるようになる．その後受講生は，解くべき課題の選択を行い，オンラインエディタを用いて選択した課題に取り組む．受講生が課題に取り組んでいる間，表 3 に示すコントローラ領域ログのログ種別に対応するイベントがオンラインエディタにおいて発生するたびに，C3PV がエディタ領域ログおよびコントローラ領域ログを内部のデータベースに保存する．さらに 3.2.2 項で述べたとおり，C3PV は newopen もしくは open が記録されてから close が記録されるまでの間 1 分ごとに，オンラインエディタのエディタ領域に記述されたソースコードを自動保存する．自動保存が行われるたびに，ログ種別 autosave が記録される．

また，compile および run は受講生がコントロール領域の同名のボタンをクリックしたときに記録されるログ種別である．これらのログ種別が記録されると同時に，C3PV は受講生のエディタ領域に記述されているソースコードを Ideone.com の compile あるいは run といった対応するウェブ API に送信する．ウェブ API から返ってきた compile や run の結果は，対応する受講生のオンラインエディタにおける出力領域に表示される．受講生は出力領域に表示された compile や run の結果を確認し，ソースコードの修正や課題の提出（submit）を行う．

講師は C3PV に講師としてログインすることで，コーディング過程ビューを閲覧することが可能となる．講師が閲覧中のコーディング過程ビューに表示されるメトリクスランキングやシートマップの情報は，C3PV によって 1 分ごとに更新される．講師は 1 分ごとに更新されるシートマップ等の情報に基づいて受講生の遅延を確認し，遅延が顕著な受講生の支援にあたる．なお，プログラミング演習ではメトリクスの詳細やシートマップの内容を講師が受講生に通知することは想定していない．

C3PV を利用するにあたって講師が行うべき事前準備は，受講生ごとのログイン ID およびパスワードの登録と，そのプログラミング演習で受講生が解くべき課題の登録のみである．

5. 評価実験

5.1 準備

C3PV が提示する可視化情報が演習中の講師にとって，遅れている受講生の発見や対応に有効であるかを評価するための実験を行った．本実験の環境を**表 5** に示す．プログラミング基礎 II は学部 1 年生が受講する Java のプログラミング講義であり，A クラスと B クラスの演習内容は基本的に同じものである．

演習開始後 30 分は相対的な差がつきにくいため，実験の対象とはしていない．30 分経過後，講師と TA は図 4 に

表 5 実験環境
Table 5 Experimental environment.

	実験 1 回目	実験 2 回目
講義名	プログラミング基礎 II	
プログラミング言語	Java	
講義時間	2 コマ（90 分 × 2）	
クラス	A	B
講師/TA	1/3 名	1/4 名
受講生	16 名	21 名

示すシートマップを閲覧しながら，M1～M4 の各メトリクスに基づく相対的な遅延を示す受講生を確認し，1 人ずつ直接対応を行った．なお，本実験では講師と TA の数を考えて，ランキング上位 3 名がシートマップ上に強調表示される設定とした．また，1 度対応した受講生には受講生の挙手等がなければ 10 分間は対応しないものとする．

講師と TA が直接受講生に対応した際には，対応した受講生とシートマップとの関係を明らかにするため，下記 6 項目を紙に記録した．

記録 1 講師らが受講生に対応した時刻
記録 2 対応した受講生のログイン ID
記録 3 演習課題名
記録 4 対応した受講生が挙手をしていたか
記録 5 受講生は実際に支援を必要としていたか
記録 6 支援を必要としていた場合，その内容は何か

記録 1～記録 3 は講師らが対応した際の時刻やどの受講生がどの問題を解いているところであったかを記録する項目である．記録 4 の項目では講師らが C3PV のシートマップを見て対応したのか，受講生の挙手を見て対応したのかを記録する．記録 5，6 では講師らが対応した受講生が実際に支援や指導を必要とする状況にあったかどうか，あった場合はその具体的な支援内容は何であるかが記録された．

さらに実験終了後，受講生に対して以下の質問 1～4 のアンケートを行った．このアンケートを用いて，これまで使ってきた通常の開発環境と比べ，受講生が支援を受けやすいと感じたか等を記入してもらった．

質問 1 オンラインエディタにあってほしい機能，足りない機能はありましたか？ あればその機能を書いてください．（記述式）
質問 2 オンラインエディタを利用することで，講師/TA のサポートが受けやすくなりましたか？（記述式：はい，いいえのどちらか）
質問 3 これまで使ってきた通常の開発環境（エディタ）と比べて，こちらで指定したオンラインエディタを使うことに関する不満点や問題点はありましたか？（記述式）
質問 4 オンラインエディタがあって良かったことや悪かったことその他なにか気づいたことがあれば書いてください．（記述式）

5.2 実験結果

表 6 に C3PV のシートマップおよび受講生の挙手によって講師らが対応した回数とその内訳を示す．この表では，講師らが各実験において対応した回数の合計を「のべ対応回数」，このうち C3PV のシートマップを見て対応した回数を「シートマップ対応回数」，受講生の挙手によって対応した回数を「挙手対応回数」として示している．シートマップ対応回数と挙手対応回数に記述されている括弧内の数値はそれぞれサポートの必要がなかった回数とシートマップで強調表示されていなかった回数を示している．

表 6 講師/TA 対応回数内訳
Table 6 Total number of directed students.

	のべ対応回数	シートマップ対応回数	挙手対応回数
1 回目	31	22（6）	9（1）
2 回目	35	23（1）	12（2）

結果として，実験 1 回目ではのべ対応回数 31 回のうち C3PV がシートマップに強調表示しなかった 1 件を除いた 30 件中 24 件（80%），実験 2 回目ではのべ対応回数 35 件のうち C3PV がシートマップに強調表示しなかった 2 件を除いた 33 件中 32 件（97%）において，C3PV はサポートを必要とする受講生の存在を正しく講師に伝えることができた．

シートマップに強調表示されなかった（すなわちメトリクスランキングの上位 3 位以内に入らなかった）3 件のうち，実験 1 回目の 1 件は演習終了時刻が迫っていたため，エラーが発生した直後に挙手をした結果，メトリクスランキング上位にならなかったという事例であった．実験 2 回目の 2 件については，2 件とも演習初期の段階に，課題内容について問合せを行うものであり，遅延に由来するものではなかった．

また，シートマップに強調表示されたにもかかわらずサポートの必要がなかった 7 件のケースのうち，実験 1 回目の 6 件については，すべてが実験開始後 30 分に集中していた．C3PV が相対的な進捗遅れを検知する仕組みになっているため，この時点では差があまり大きくなかったことが原因の 1 つであると考えられる．実験 2 回目の 1 件については，エラーへの対応に詰まっていた受講生だったが，講師が対応に行く際にちょうど独力で解決できたというケースであった．

5.3 アンケート結果

受講生への質問 1～4 のアンケートの集計結果を**表 7** に

表 7 受講生へのアンケート
Table 7 Result of enquete.

実験 1 回目（有効回答数 16）		
	はい	いいえ
質問 1：足りない機能があるか？	5	11
質問 2：サポートが受けやすくなったか？	15	1
質問 3：指定したオンラインエディタを使うことに問題点があるか？	1	15
実験 2 回目（有効回答数 21）		
	はい	いいえ
質問 1：足りない機能があるか？	3	18
質問 2：サポートが受けやすくなったか？	16	5
質問 3：指定したオンラインエディタを使うことに問題点があるか？	9	12

示す．機能追加についての要望としては，実験 1 で 5 件，実験 2 では 3 件あった．追加の要望としては，eclipse にあるようなコードの予測変換や undo 機能についてのものが 2 件，出力領域のサイズ変更や結果表示の初期化に関するものが 2 件，自分の打ち込んだソースコードを一括で DL する機能や提出結果の一覧といった提出に関する機能が 2 件，実際にはあったが説明していなかったために要望に含まれていたオートセーブに関するものが 2 件あった．

C3PV によってサポートが受けやすくなったかという質問については，実験 1 回目のクラスでは 1 人を除く全員が，実験 2 回目では 21 人中 16 人が “はい” と回答した．ここで “いいえ” と回答した受講生を調査したところ，すべて成績が優秀で演習中の進捗も一定して他の受講生より進んでいる受講生であった．

こちらが指定したオンラインエディタを利用して開発を進めることに対して，実験 1 のクラスでは 1 人，実験 2 では 9 人が問題点があると回答した．実験 1 のクラスで指摘された唯一の問題点は着席処理の際に不具合が発生したという受講生によるものであった．この受講生は実験時に着席処理画面が表示されず，しばらく座席情報の登録ができない状態になっていた．最終的にその受講生も着席処理を完遂できたが，しばらく演習にとりかかれなかったためアンケート結果において問題点があると回答を行っていた．実験 2 のクラスでは問題点があると指摘した受講生が 9 人いたが，そのうち 8 人は，コンパイルに時間がかかった点を問題点としてあげていた．これは，C3PV が外部の Web サービスを利用してプログラムのコンパイルを行っており，実験を行った日にたまたまその Web サービスのレスポンスが悪かったことに起因している．残り 1 人の指摘した問題点は，ブラウザの戻るボタンを間違えてクリックすることでソースコードが消えてしまうという症状についてのものであった．

質問 4 では自由記述でオンラインエディタの利用についての感想を聞いた．この質問ではほとんどの受講生が提出が楽になったことや悩んでいるときに声をかけてもらえたという点を感想としてあげていた．

6. 考察

6.1 利点と限界

5.2 節で述べた実験結果より，演習開始 1 時間後以降に限れば，C3PV の提示に従って直接対応したほぼすべてのケースにおいて実際に受講生がサポートを必要としていた．また，挙手対応のほとんどの事例において C3PV がシートマップ上に強調表示を行っていたことから，受講生が必ずしも挙手をしなくても，C3PV によって対応が必要な受講生として講師が知ることが可能であった．本実験におけるサポートの内訳としては，エラーの解決についての指導や課題の解き方の解説といった理解不足への対応がほとんどであったが，課題を解くのを諦めてほかのことをやっている受講生への注意といった事例も数件存在した．以上の結果より，受講生が演習に真面目に取り組んでいるかを確認しつつ，理解不足で悩んでいる受講生を効率的に早期に発見することが C3PV によって可能となったと考えられる．

また 5.3 節のアンケートからは，オンラインエディタの機能に 8 割近くの受講生が満足しており，8 割以上の受講生がサポートを受けやすくなったと感じており，7 割以上の受講生が C3PV のエディタを利用することに問題点を感じていないという結果が得られた．自由記述からもほとんどの受講生がサポート面についてポジティブなイメージを持っていたことが分かった．特に，開発過程がモニタリングされていることに対して疑問や問題に感じている受講生が 1 人もおらず，遅れている際に声をかけてもらえるといったメリットを感じているといったコメントが多かった．以上より受講生全体に C3PV の有効性が認識されていると考えている．

2.2 節で述べた R1〜R3 の各要求については以下に示す提案システムの 3 つの特徴によって対応できたといえる．

A1 C3PV のすべての機能にユーザはブラウザからアクセスできる．また，演習を行う際に必要な事前準備もログイン ID とパスワードおよび演習課題の登録のみであるため，講師や受講生に対するセットアップコストがほぼかからない．

A2 M1〜M4 の各メトリクスによって，多様な遅延や停滞状況の検知が実際に可能であることが本論文で行った実験により確認できた，

A3 M1〜M4 で定義されたすべてのメトリクスは特定の開発言語に依存するものではない．

アンケートにより得られた今後の課題に関する意見の中で，既存の統合開発環境に含まれている高度な機能の一部が C3PV のオンラインエディタには不足しているというものがあった．C3PV で実装されているオンラインエディタ（図 2）では，コントローラ領域の Save/Compile/Run といったソースコードの保存やコンパイル，実行に相当する機能とエディタ領域におけるコードエディタとしての機能が実装されている．コードエディタの機能としては，シンタックスハイライト，オートインデントやソースコード中の括弧の対応付けといった基本的な編集機能が，既存の統合開発環境同様 C3PV のオンラインエディタにおいても実装されている．一方でコードの予測変換や文法エラーの自動チェック・自動修正，無制限の undo といった高度な機能については実装されていない．これらの高度なコードエディタの機能については，本論文の提案システムが初学者向けであることから，実装することを想定していない．

出力領域のサイズ変更については，現時点で対応しているが，利用方法が受講生に伝わっていなかったために問題点として提示されていた．今後は提出状況の確認といった

UI のユーザビリティの改善とマニュアルの充実を検討している．外部のコンパイル用サービスのレスポンスについては，外部の Web サービスを利用しているため，我々では対応が困難である．今後は同種のサービスを自前で提供することでパフォーマンスの改善を図っていきたい．

C3PV は進捗が相対的に遅れている初学者を早期に発見し，講師がフォローすることを目的としている．そのため我々は C3PV によって，優秀な受講生の自律的な学習を支援するというよりは，潜在的に支援を必要とする受講生へのフォローによってプログラミングに対する苦手意識が生まれないようにすることを目指している．今後は，アンケートにもあった優秀な受講生への対応を想定し，優秀な受講生のピックアップや，早期に課題を終了させた受講生向けに新しい課題を出すといったより多くの受講生へのサポートも考慮していきたい．

6.2 関連研究

プログラミング演習における受講生の理解度把握の支援を目的とした研究は演習修了時に次の演習に向けて行われるものと演習中にコーディング課程を分析して支援を行うものの 2 種類が存在する．

宮地らはあらかじめ設定した正解プログラムを用いて答案プログラムの正誤判定を自動的に行う，アセンブラ言語に特化した教育支援システム CAPES [4] を開発している．CAPES は受講生の解答内容に基づく正誤判定結果により受講生の理解度を判定し，受講生ごとに異なった課題の出題を行うことができる．

田口らは受講生がこれまでに解いた演習内容と教員および受講生自身による評価に基づいて協調フィルタリング手法を適用し，受講生の理解度の推定と受講生が次に解くべき演習内容を提示する手法を提案している [6]．

倉澤らは受講生によるコンパイル作業履歴と事前に分類されたエラー要因から受講生の多くが実際に陥っているエラーの状況とその原因を推定し，講師にフィードバックを行う手法を提案している [10]．

これらの研究はいずれも受講生によって提出されたソースコードやコーディング後のコンパイル作業履歴を対象として分析を行うものであるため，コーディング過程内の受講生の状況を早期に把握することを目的としていない．また特定の言語に特化しているため，アセンブラ，C，Java，初学者向けの独自プログラミング言語 [8] といった多様な開発言語に対応することが困難である．一方で，ソースコードからの理解度分析や講師による評価のフィードバックといった機能は C3PV に組み込むことが可能である．今後，講師や受講生の負荷を増加させずに対応できる範囲で，演習後分析機能についても検討を進めたい．

受講生のコーディング過程を対象とした早期の理解度把握を支援するシステムとして，藤原らは受講生のコンパイル回数，実行回数，コンパイルエラー数，行数等の履歴を利用するものを提案している [12]．藤原らのシステムでは，講師がコーディング過程の各種履歴を組み合わせたパターンを定義しておくことで，過度に進捗が遅れている等の事前に決められたパターンに合致する受講生を検出することが可能である．しかしながら，収集された履歴の分析およびパターン化といった処理を事前に講師が行う必要があるため，講師の負荷が大きく，可視化によるフィードバックにも対応していない．また，定義したパターンに一致する受講生を複数人検出したとき，どの受講生を優先して支援するべきか判断できない．これは受講生数に対して講師や TA の数が少ない場合に重要な課題となる．実際に C3PV では，シートマップに強調表示する基準となるメトリクスランキングの数値を設定により変更することが可能である．今後はより多くのメトリクスの設定と実証実験を行うことで，有効なメトリクスの分析・検証と C3PV の実装へのフィードバックを行っていきたい．

7. おわりに

C3PV は，課題を作成する過程とランキングテーブルによる受講生間の相対的な比較から，プログラミング演習中に遅れておりサポートを必要とする受講生を早期に効率良く検出可能であることが確認できた．

本実験では Java を用いたが，Ideone を使用しているため様々なプログラミング言語に対応できる．また，言語に依存しない要素を使用して M1〜M4 を定義しているため，他の言語においても C3PV の適用が可能である．ユーザはブラウザを通じて C3PV のすべての機能にアクセスできるため，利用時のコストを非常に低くすることが可能となった．今後他の教育機関等様々なユーザに C3PV を利用してもらう際に，クラウド環境を利用した仮想マシン単位で C3PV を配布することにより，容易に C3PV の導入が可能となると考えている．今後の課題としては，エディタ環境および可視化環境の使い勝手の向上や，より多くのメトリクスの設定と実証実験を行うことによる有効なメトリクスの分析・検証，C3PV の実装へのフィードバックを行っていきたい．さらに，講義外で C3PV を利用する受講生の支援にも役立てていきたい．

謝辞 本研究は，日本学術振興会科学研究費補助金若手研究（B）（課題番号：24700030）の助成を得た．

参考文献

[1] Ajax.org Cloud9 Editor: Ace, available from ⟨http://ace.ajax.org/⟩.
[2] 独立行政法人情報処理推進機構：IT 人材白書 2011 (2011).
[3] Sphere Research Labs: Ideone.com, available from ⟨http://ideone.com/⟩.
[4] 宮地恵佑，高橋直久：構造誤り検出機能を有するアセンブラプログラミング演習支援システムの実現と評価，電子情

報通信学会論文誌，Vol.J91-D, No.2, pp.280–292 (2008).
[5] 堀口悟史，井垣 宏，井上亮文，山田 誠，星 徹，岡田謙一：講義資料閲覧ログを用いたプログラミング講義進捗管理手法の提案，情報処理学会論文誌，Vol.53, No.1, pp.61–71 (2012).
[6] 田口 浩，糸賀裕弥，毛利公一，山本哲男，島川博光：個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援，情報処理学会論文誌，Vol.48, No.2, pp.958–968 (2007).
[7] 国立情報学研究所：edubase Cloud, 入手先 ⟨http://edubase.jp/cloud/⟩.
[8] 西田知博，原田 章，中村亮太，宮本友介，松浦敏雄：初学者用プログラミング学習環境 PEN の実装と評価，情報処理学会論文誌，Vol.48, No.8, pp.2736–2747 (2007).
[9] 匂坂智子，渡辺成良：プログラミング初学者の学習方略と段階的理解度に関する調査および支援ルールの作成について（特集次世代情報教育の構築に向けて）—（プログラミング教育），教育システム情報学会誌，Vol.26, No.1, pp.5–15 (2009).
[10] 倉澤邦美，鈴木恵介，飯島正也，横山節雄，宮寺庸造：プログラミング演習における一斉指導のための学習状況把握支援システムの開発（collaboration と agent 技術/一般），電子情報通信学会技術研究報告 ET，教育工学，Vol.104, No.703, pp.19–24 (2005).
[11] 加藤利康，石川 孝：授業支援システムにおけるプログラミング演習のための学習状況把握支援機能の設計と評価，研究報告コンピュータと教育（CE），Vol.2012-CE-113, No.6, pp.1–8 (2012).
[12] 藤原理也，田口 浩，島田幸廣，高田秀志，島川博光：ストリームデータによる学習者のプログラミング状況把握，電子情報通信学会第 18 回データ工学ワークショップ，pp.1–6 (2007).

井垣 宏 （正会員）

2000 年神戸大学工学部電気電子工学科卒業．2002 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2005 年同大学院博士後期課程修了．2011 年大阪大学情報・特任准教授．博士（工学）．ソフトウェア工学教育，ソフトウェアプロセス等の研究に従事．IEEE, ACM，IEICE 各会員．

齊藤 俊

2011 年東京工科大学コンピュータサイエンス学部卒業．グループウェアおよび教育支援システムの研究に従事．

井上 亮文 （正会員）

1999 年慶應義塾大学理工学部計測工学科卒業．2005 年同大学院後期博士課程修了．博士（工学）．現在，東京工科大学コンピュータサイエンス学部講師．グループウェア，音楽情報処理の研究に従事．本会論文誌編集委員．ヒューマンインタフェース学会，ACM 各会員．

中村 亮太 （正会員）

2008 年慶應義塾大学大学院理工学研究科博士後期課程修了．博士（工学）．2007～2011 年東京工科大学メディア学部助教．2011 年より同大学片柳研究所研究員．ヒューマンコンピュータインタラクション，グループウェア，ネットワークサービス，生体情報活用等の研究に従事．DICOMO2005 & 2006 優秀論文賞受賞．2007 年山下記念研究賞受賞，GNWS2011 優秀論文賞受賞．ACM，日本教育工学会各会員．

楠本 真二 （正会員）

1988 年大阪大学基礎工学部卒業．1991 年同大学院博士課程中退．同年同大学基礎工学部助手．1996 年同講師．1999 年同助教授．2002 年同大学大学院情報科学研究科助教授．2005 年同教授．博士（工学）．ソフトウェアの生産性や品質の定量的評価に関する研究に従事．電子情報通信学会，IEEE，IFPUG 各会員．