社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

プログラミング演習における受講生支援のための コーディング過程可視化システムの提案

齊藤　俊† 山田　誠†† 井垣　宏††† 楠本　真二††† 井上　亮文†
星　徹†

† 〒 192-0982 東京工科大学コンピュータサイエンス学部 東京都八王子市片倉町 1404-1
†† 〒 192-0982 東京工科大学大学院バイオ・情報メディア研究科 東京都八王子市片倉町 1404-1
††† 〒 560-8531 大阪大学大学院情報科学研究科 大阪府豊中市待兼山町 1-3

あらまし　本稿では，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可視化するシステム 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を用いることで，受講生のコーディング，コンパイル，実行，提出といった過程を記録する．C3PV はこのコーディング過程から，課題の進み具合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い時間悩んでいる受講生や全体の進捗に対して遅れているといった支援すべき受講生を C3PV によって確認し，実際に支援につなげることができる．実際に提案システムを学部 1 年生が受講する Java プログラミング演習に適用し，コーディング過程の可視化および支援すべき受講生の検出が可能であることを確認した．
キーワード　プログラミング演習，教育支援，進捗管理，開発者メトリクス

Programming Process Visualization for Supporting Students in Programming Exercise

Syun SAITO†, Makoto YAMADA††, Hiroshi IGAKI†††, Shinji KUSUMOTO†††, Akifumi INOUE†, and Tohru HOSHI†

† School of Computer and Science, Tokyo University of Technology
1404-1, Katakura, Hachioji, Tokyo, 〒 192-0982 Japan
†† Graduate School of Bionics, Computer and Media Science, Tokyo University of Technology
1404-1, katakura, Hachioji, Tokyo, 〒 192-0982 Japan
††† Graduate School of Information Science and Technology, Osaka University
1-3, machikaneyama, Toyonaka, Osaka, 〒 560-8531 Japan

Abstract　In this paper, we propose a Coding Process Visualizer in Programming Practice (C3PV). The C3PV records coding processes of students in the view point of LOC, operation records of our online editor, interval of time required for correcting errors. In the C3PV, such records indicate some students whose development progresses are delayed relatively. With using the C3PV, teachers can acquire students who need support. In the experiments of this paper, we confirmed that 88% of students indicated by C3PV have needed actual support by teachers to proceed with their subjects.
Key words　Programming Exercise, Education Support, Progress Management, Deverlopper Metrics

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伴い，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．

通常プログラミング演習では，受講生は各自のペースで課題のコーディングを行い，講師が受講生の質問に対応するという形式で行われる．受講生による質問はコーディング中に疑問に思った点や解決できないエラーが発生したときに行われることが多い．

一方で，質問をせずにコーディングが遅延したり停滞したりする受講生も数多く存在する．そのため，講師は受講生による能動的な問い合わ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．

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 Assintant) 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．そのため，コーディング中の行動や提出されたソースコードを用いて受講生の学習状況を推測する研究が複数行われている [1] [2] [3].

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システム 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，ランキングにして表示する．C3PV がランキング上位に該当する受講生をサポートが必要な受講生として可視化し，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．

プログラミング演習で C3PV を用い，C3PV が遅延や停滞を示した受講生に講師/TA が対応する実験を行った．実験では，講師/TA が対応した 45 の事例の約 84% において受講生がサポートを必要としていた．結果として，C3PV を用いることで，サポートを必要とする受講生を効率良く検出し，対応することが可能であることが確認できた．

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. 準　　備

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．本稿では，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 プログラミング演習における課題

通常，講師や TA が受講生ごとのコーディング進捗度合いを知るのは，時間や対応人数の制約から，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限られている．

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 [1] を開発した．CAPES は受講生の解答内容に応じた課題の出題や指導を行うことが可能である．しかしながら，提出されたソースコードの答案から，課題に対する正否だけでなく，受講生の理解度や学習状況を講師が把握することは非常に困難である [4]．結果として，理解不足で遅延している受講生や，エラーが解決できずに停滞しているといったコーディング中の受講生を支援することも難しい．

受講生のコーディング過程を対象とした教育支援システムとして，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 [2]．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．

そこで本研究では，これらの課題の改善を目的として，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簡単

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，受講生が容易に利用できることが望ましい．また，リアルタイムに講師が受講生の遅延状況を検知するためには，課題ごとの提出時刻といった時間に関するメトリクスだけでなく，LOC やコンパイルエラーの情報といった多様なメトリクスを収集する環境が必要である．さらに，多様な言語への対応を考慮すると，収集されるメトリクスは特定の言語に依存しないほうが良い．

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，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 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

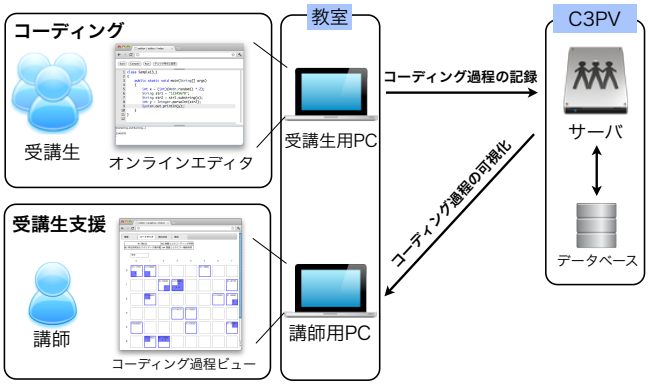


図 1　コーディング過程可視化システム C3PV の概要

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，コーディング過程ビューとして可視化され，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簡単である．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．総 LOC や課題ごとのコーディング時間といったメトリクスはコーディング対象のプログラミング言語に依存しないため様々な言語の演習に適用可能である．

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

本稿で提案する C3PV において受講生は**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) 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 [5] と呼ばれる OSS 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を実現している．C3PV はこのオンラインエディタを通じて，受講生一人ひとりのコーディング過程を記録する．

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．この例では，SID 2 に対応するソースコードについて，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

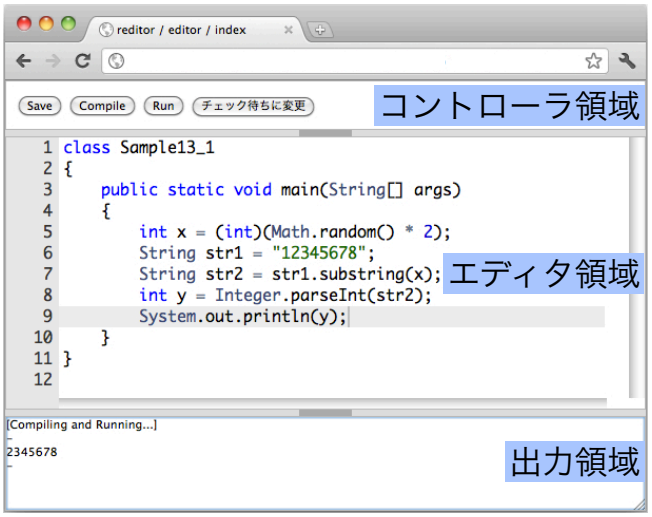


図 2　オンラインエディタを用いた Java プログラムのコーディング

表 1　エディタ領域ログの種別

ログ種別名	内容
insertText	一行以内の文字を入力
removeText	一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

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

表 3　コントロール領域ログの種別

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 以降に対象課題が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

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　コントロール領域ログ

C3PV は受講生がコントロール領域において行う操作とその時刻をコントロール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，それ以外は全て受講生の行った振る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，エディタのページがブラウザが閉じられたり，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 close が記録される．

表 4　コントロール領域ログの例

SID	ユーザ 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			
4	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
5	syuns	run	2012-2-1 15:31:20	Java	3: ...world!");	hello, world!	
6	syuns	save	2012-2-1 15:31:25	Java	同上		
7	syuns	submit	2012-2-1 15:31:30	Java	同上		
8	syuns	close	2012-2-1 15:31:35	Java	同上		

表 4 に C3PV が記録するコントロール領域ログの例を示す．前提として，講師は「hello, world!」と表示する Java の課題を課している．この例では，初めに，ユーザ 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元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た（close）．

3.3　コーディング過程メトリクス

受講生の遅延状況可視化を目的として，以下の 4 種類のコーディング過程メトリクスを提案する．

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

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．そのため，課題ごとの行数ではどの受講生が遅れているかはわからない．また，課題ごとに行数が異なる可能性があるので，異なる課題を解いている受講生間での比較を行うことも難しい．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ことで，M1 がより少ない場合を全体的に進捗が遅れている受講生として検知できる．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．コントロール領域ログとして 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．例えば，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

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

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が，実際には 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．

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

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

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 位であることがわかる．すなわち，総 LOC が他の受講生より少なく，現在取り組んでいる課題に時間を要し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

C3PV は edubaseCloud [6] の仮想マシン (Xeon 2.4GHz CPU 8 コア，メモリ 24GB) 内に実装されている．OS は CentOS5.6 である．ウェブサーバおよび DBMS はそれぞれ Apache/2.2.3，MySQL/5.0.77 である．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，特にオンラインエディタ部分には Ace/0.2.0 [5] を利用して開発を行った．

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 [7] 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．

表 5　実 験 環 境

	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

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．受講生はログイン後，システムを利用した着席処理と課題選択を行ってエディタを開き課題に取り組む．講師はログイン後，シートマップを開いて受講生の遅延を確認しながら遅延が顕著な受講生の支援にあたる．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．

5. 評 価 実 験

5.1 準　　備

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．

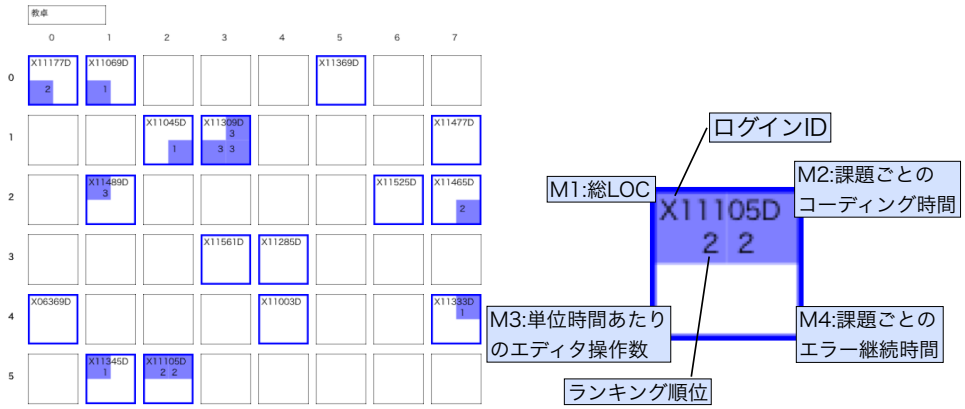
演習開始後 30 分は相対的な差が付きにくいため，実験の対象とはしていない．30 分経過後，講師と TA は図 4 に示すシートマップを閲覧しながら，M1∼M4 の各メトリクスに基づく相対的な遅延を示す受講生を確認し，一人ずつ直接対応を行った．

講師/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/TA 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

5.2 結果および考察

表 6 に C3PV のメトリクスランキングおよび受講生による挙手に基づいて講師/TA が対応した回数とその内訳を示す．この表では，講師/TA が各実験において対応した回数の合計を「のべ対応回数」，このうち受講生の挙手によって対応した回数を「挙手回数」，シートマップで支援の必要ありと提示されたものの実際にはその必要がなかった回数を「サポート必要なし」として記述している．なお，挙手回数の項目にある括弧は受講生が挙手をした際に，C3PV がシートマップにその受講生が遅延していると提示しなかった数を示している．



(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

表 6　講師/TA 対応内訳

	のべ対応回数	挙手回数	サポート必要なし
1 回目	31	9(1)	6
2 回目	35	12(2)	1

表 7　C3PV が検知した遅延状況詳細

	のべ対応回数				挙手回数			
	M1	M2	M3	M4	M1	M2	M3	M4
1 回目	17(2)	20(5)	12(2)	10(0)	3	4	1	3
2 回目	15(0)	12(0)	9(0)	22(1)	3	5	4	6

結果として，実験 1 回目ではのべ対応回数 31 回のうちから C3PV がシートマップに提示しなかった 1 件を除いた 30 件中 24 件 (80%)，実験 2 回目では同様に 33 件中 32 件 (97%) において，提案システム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．

検知できなかった 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 件とも課題を解く初期の段階に，課題内容について問い合わせを行うものであり，遅延に由来するものではなかった．

サポート必要なしとなったケースでは，実験 1 回目の 6 件については，全てが演習開始後 1 時間以内 (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一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．

表 7 に講師/TA が対応した事例のうち，C3PV が遅延していると提示していたものの対応するメトリクス分類を示す．なお，括弧内はサポートの必要がなかった回数を表している．本実験では，M1〜M4 のうち M3 を除く全てのメトリクスにおいて，単独で受講生の遅延状況を示す事例が存在した．すなわち，M3(単位時間あたりのエディタ操作数) のみは他のメトリクスと同時に遅延していることを示す事例がほとんどであった．

以上の結果より，本実験において受講生の遅延状況を検知して対応するには，本稿で提案した M1，M2，M4 のメトリクスがあれば十分であると判断できる．今後 M3 については，操作の意図 (コピー&ペーストの内容等) など，より踏み込んだ分析に利用することを検討したい．

6. 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少なくとも 8 割以上の検出率で発見でき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．

今後は単独での有用性が確認できなかったメトリクス M3(単位時間あたりのエディタ操作数) を用いた，受講生のより詳細なコーディング過程の可視化について検討していきたい．

文　献

[1] 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.02，pp.280–292，Feb. 2008.
[2] 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，March 2007.
[3] 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，Feb. 2005.
[4] 知見邦彦，坂庭裕希子，櫨山淳雄，宮寺庸造，“失敗知識を利用したプログラミング学習環境の構築，” 電子情報通信学会技術研究報告. ET, 教育工学，vol.104，no.48，pp.7–12，2004.
[5] “Ace(ajax.org cloud9 editor)，” http://ace.ajax.org/. 参照 Feb. 21,2012.
[6] “edubaseCloud，” http://edubase.jp/cloud/. 参照 Feb. 21,2012.
[7] “Ideone，” http://ideone.com/. 参照 Feb. 21,2012.