ビジュアルプログラミング対応版 プロセス観察システムの開発と学習効果分析

田中 良樹[1,a)]　松澤 芳昭[2,b)]　酒井 三四郎[3,c)]

概要：プログラミングプロセス観察支援ツール (PPV: Programming Process Visualizer) を拡張し，ブロック型言語への対応を行った．本拡張における設計目標は，1）実際の授業現場で運用可能なデータ量でのログ保存と分析場面での性能を両立させること，2）ブロック言語開発者のための分析支援機能を追加すること，の2点である．この目標達成のために，画像データとイベントログデータのハイブリッド方式のプロセス復元方式と，ブロックの操作種類によるフィルタリング機能を実装した．大学文科系向けのプログラミング入門講義で課される課題への適用実験では，試行錯誤のプロセス数が十分に大きい場合を想定したとしても，ログデータサイズは 100MB 以内，かつスムーズなプロセス復元を達成できた．次に，本ツールを利用して，開発者が BlockEditor の一機能に関する教育効果の分析を行った．学生 9 名のデータを用いた分析の結果，開発者が現状のブロック言語の問題を明確化し，具体的な機能改善案の導出を行うことができた．

キーワード：プログラミング教育，プロセス分析，ビジュアルプログラミング言語，学習効果，教育情報システム評価

1. はじめに

言語特有の構文エラーは，プログラミング学習者がアルゴリズムを構築する学習に集中することの妨げになる可能性がある．プログラミング学習者がまず習得するべき能力はプログラミング言語そのものではなく，プログラムを構成する要素の概念の理解と，それらを組み合わせてアルゴリズムを構築する能力である．

構文エラーが発生しないように設計されたビジュアルプログラミング言語と，テキスト記述型言語の相互変換を用いたプログラミング教育環境（BlockEditor）が松澤らによって提案されている [1]．OpenBlocks フレームワークを用いて開発された BlockEditor は，ブロック型の命令を組み合わせることでプログラムを記述する．BlockEditor で記述されたビジュアルプログラミング言語は Java と相互変換が可能である．

[1] 静岡大学大学院 総合科学技術研究科 情報学専攻
Department of Informatics , Graduate School of Integrated Science and Technology , Shizuoka University
[2] 青山学院大学社会情報学部
School of Social Informatics, Aoyama Gakuin University
[3] 静岡大学情報学部
Faculty of Informatics, Shizuoka University
[a)] tanaka@sakailab.info
[b)] matsuzawa@si.aoyama.ac.jp
[c)] sakai@inf.shizuoka.ac.jp

しかし，BlockEditor による学習が学習者の理解度の向上に寄与したというデータはまだ得られていない．現在明らかになっているのは，BlockEditor の利用率の分析によって，ブロック型言語からテキスト記述型言語へのシームレスな移行が支持されていること，それによって，BlockEditor はブロック言語からテキスト記述型言語へ移行する際の足場かけになっていること，である．

そこで本研究では，プログラミングのプロセスを明らかにするというアプローチで，BlockEditor が学習者の理解度の向上に寄与していることを明らかにしようと考えた．学習者のプログラミングプロセス分析と，学習者へのインタビューを併用することで，BlockEditor の機能の学習支援効果を検証しようと試みる．

本研究の目的は，松澤らによって提案された，テキスト記述型言語のプログラミングプロセスの観察・分析を支援するツール ProgrammingProcesVisualizer[2] に機能を追加して，ブロック型言語のプログラミングプロセスを観察・分析するツールの提案を行うこと，提案ツールを用いて，ブロック型言語の機能に対する学習支援効果を検証することの 2 点である．

2. 先行研究

ブロック型の命令を組み合わせることでプログラムを実

装できるビジュアルプログラミング言語のプログラミングプロセス分析に関する直接の先行研究は確認できなかった．本章では，ビジュアルプログラミング言語の教育における研究と，テキスト型言語のプログラミングプロセス分析に関する先行研究について述べる．

ビジュアルプログラミング言語を用いた教育支援システムとして，Squeak eToys[3] や Scratch[4] などがある．これらのシステムはプログラミング入門教育において利用されている．ビジュアルプログラミング言語の教育効果を論じた研究はいくつかみられる [5][6] が，主流は事前事後テストの結果によって効果を論じる研究方法であり，プロセスに言及している研究は少数である．

少数ではあるが，テキストプログラミング教育の研究分野では， プロセスの重要性が指摘されている [7]．プログラミングのプロセス分析を目的としたコーディング過程のリプレイツールの提案も行われてきている [8][9].

ProgrammingProcessVisualizer（PPV）は松澤らによって開発された Java における，プログラミングプロセス分析ツールである．開発環境にセンサを埋め込み，操作ログを獲得し，リポジトリに記録する．センサが埋め込み可能な開発環境は Eclipse と，松澤らが独自開発している教育用エディタであり，センシングのレベルはタイピングレベルである．PPV はリポジトリに記録されたログデータを元に，プログラミングプロセスをアニメーションで復元し，分析を可能にする．しかし，これらのツールはブロック型言語のプログラミングプロセス分析には対応していない．

プログラミング演習におけるコーディング過程可視化システムである C3PV が井垣らによって提案されている [10]. C3PV は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，可視化を行うシステムである．C3PV は学習者のプログラミングをリアルタイムで把握できること，様々なテキスト記述型言語での対応が可能であることを主張している．しかし，C3PV はブロック型言語に対応していない．さらに，学習者のプログラミングプロセス分析ではなく，サポートが必要な学生を確認するために提案されたシステムであり，目的が異なる．

プログラミング初学者のコーディングの傾向が伏田らによって分析されている [11]．この研究では，学習者のプログラムをトークンに分割し，学習者のプログラムが回答のプログラムに近付く過程を観察すること，プログラミング熟練者による目視による定性的分析が行われている．分析の結果，学習者は修正するべきでない部分の編集を行うこと，他のプログラミング言語の知識に引きずられることなどが明らかになっている．しかし，学習者のエラー修正に対する分析にとどまっており，学習者の試行錯誤のプロセスの分析は行われていない．

3. BlockImageVisualizer の設計

BlockImageVisualizer（以下 BIVi）は BlockEditor におけるプロセス（以下，プロセスはプログラミングプロセスを指す）を復元する目的で提案するツールである．

提案ツールの使用目的は，1) 学習者がプロセスを振り返り，自学自習，または指導者の補助を受けて学習効果を高めること，および 2) 開発者による BockEditor の機能の有用性調査の支援を行うこと，である．

3.1 プロセスデータの定義

プログラミングプロセスの復元に用いるプロセスデータの定義を行う．

イベントログとは，プログラミングを行う際の操作ログである．BlockEditor から取得するイベントログには，ブロックの種類（変数宣言や while ループなど）と，ブロックに対する操作（追加や連結など）の情報が時系列順に保存されている．

XML ファイルとは，BlockEditor 上の各ブロックの状態を保存するために用いる XML 形式のファイルのことである．画像データとは，BlockEditor 上の全てのブロックを含んだキャプチャ画像のことである．これらのデータを総称してプロセスデータとする．

3.2 プログラミングプロセスの復元方式

ブロック型言語におけるプロセスの復元の方法として，1)「イベントログによる差分復元方式」，2)「XML ファイルを用いた復元方式」，3)「画像データとイベントログデータのハイブリッド方式」の 3 つを考えた．

「イベントログによる差分復元方式」はイベントログを利用することで，実際に行われたプログラミングの手順と同様にプログラムを構築し，プログラミングプロセスを復元する方式である．「XML ファイルによる復元方式」は，XML ファイルを用いてブロックの状態を復元するプロセス復元方式である．「画像データとイベントログデータのハイブリッド方式」は画像データとイベントログデータを併用する，ハイブリッド型のプロセス復元方式である．

各方式の長所と短所は以下の通りである．

イベントログによる差分復元方式

長所　プロセスデータを小さくできる．

短所　実装コストが大きい，プロセスの復元に時間がかかる．

XML ファイルによる復元方式

長所　画像データに比べ，プロセス復元に必要なデータのサイズを小さくすることが可能．

短所　実装コストは中程度，プロセスの復元に時間がかかる．

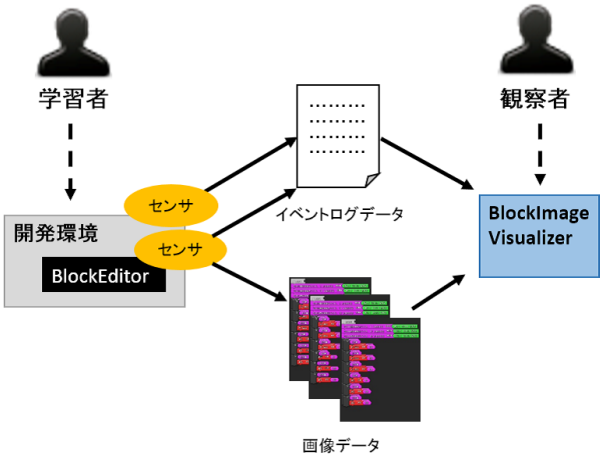

図 1　提案システム全体のシステム構成図

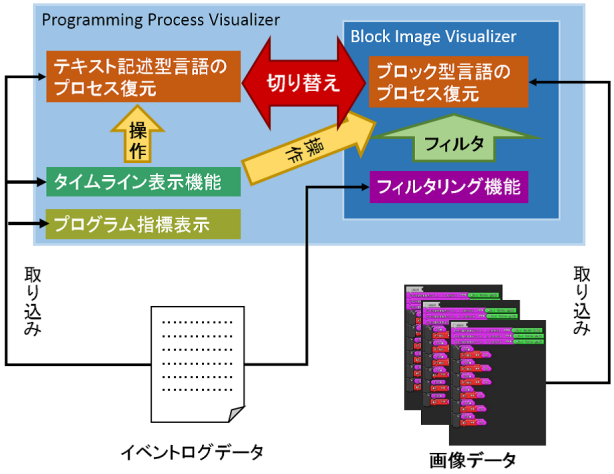

図 2　BlockImageVisualizer 内部のシステム構成

画像データとイベントログデータのハイブリッド方式

長所　様々なビジュアルプログラミング言語にも対応可能，プロセスの復元にかかる時間を最小限にすることが可能．

短所　プロセス復元に必要なデータサイズが 3 方式中，最も大きい．

提案ツールではスムーズな分析を行うこと，実装コストを抑えることを優先し，「画像データとイベントログデータのハイブリッド方式」（以下ハイブリッド方式）を採用した．

加えてハイブリッド方式はプログラミングプロセスの復元に，ビジュアルプログラミング環境の内部のシステムを用いないため，多種多様なビジュアル言語に対しても，プロセスデータの取得が低コストで実装可能であるということも利点である．多種多様なビジュアル言語とは，例えば，フローチャートやクラス図，オブジェクト図などである．

3.3　提案システムの概要

提案システム全体の構成図を図 1 に示す．開発環境とは，松澤らによって独自に開発された教育用のエディタであり，BlockEditor は開発環境上で動作する．

学習者が Java でプログラミングを行った場合，開発環境に埋め込まれたセンサによりイベントログを取得する．学習者が BlockEditor でプログラミングを行った場合，BlockEditor に埋め込まれたセンサによりイベントログと画像データをそれぞれ取得する．

観察者はイベントログデータと画像データを BlockImageVisualizer に取り込み，プロセスの観察を行う．

3.3.1　画像データ

画像データとは BlockEditor 上のすべてのブロックを含む画像ファイルである．画像ファイルは jpg 形式である．

3.3.2　イベント

基本イベントとして，ブロック同士を連結させる動作 (connect)，ブロックの連結を解除させる動作 (disconnect)，そして，新たなブロックを追加する動作（add）の 3 種類を定義し，記録している．必要に応じて開発者がイベントを追加することも可能である．

3.4　BlockImageVisualizer

提案ツールはテキスト記述型言語のプログラミングプロセスを復元し観察する ProgrammingProcessVisualizer（PPV）に機能を追加して実装を行った．そのため PPV 元来の機能はすべて継承し，利用することができる．

3.4.1　ブロック型言語のプロセス復元方法

BIVi 内部のシステム構成を図 2 に示す．BIVi は PPV に実装されている機能である「タイムライン表示機能」に対して操作を行うことでタイムライン上に示された時刻のプログラミングプロセスを復元する．指定された時刻において，テキスト記述型言語でプログラムが行われていればソースコードを，ブロック型言語で記述されていれば画像データをそれぞれ表示する．

ハイブリッド方式では，イベントログデータに記録されたイベントログと画像データが紐付けされており，イベントログデータの時刻を利用して，その時刻の画像データを表示する．

3.4.2　フィルタリング機能

フィルタリング機能はイベントの種類ごとにログにフィルタをかけ，観察を行うことを目的とした機能である．フィルタリング機能が初期設定にて対応しているイベントログの種類は add，connect，disconnect の 3 種類である．イベントログと画像データが紐付けされているため，任意のイベントログのみを抽出することでフィルタリング機能を実装した．

この機能により，特定のイベントに関連するプログラミングプロセスの分析が容易になる． 例えば，学習者が既に作成したプログラムの誤りを修正する場合，学習者は作成されたブロックの連結を解除して誤った部分を取り除き，

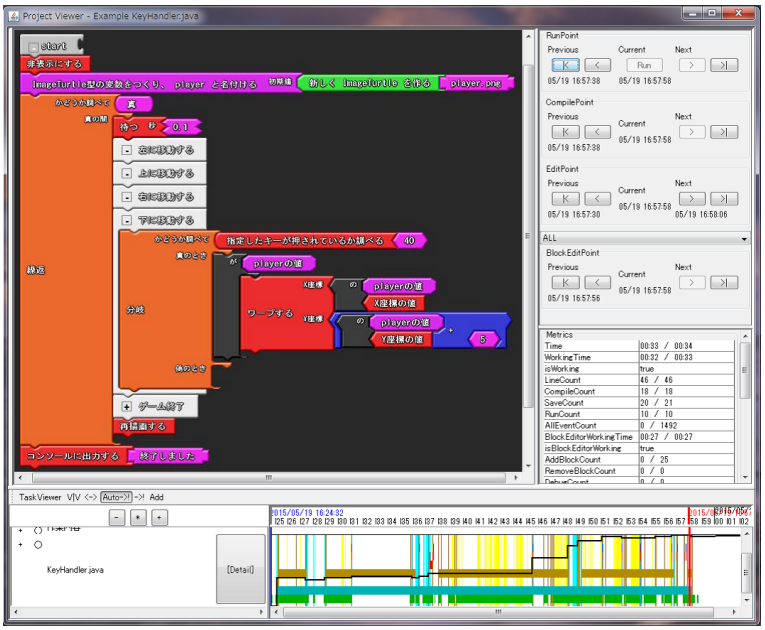

図 3　プロジェクト・ビュー・ウィンドウ

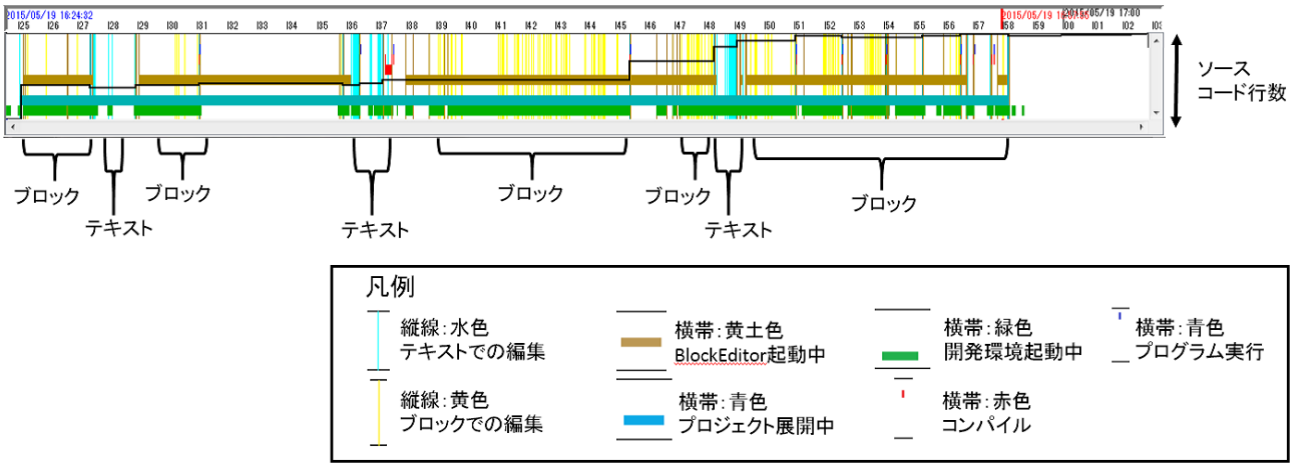

図 4　タイムラインパネル

正しいブロックを追加するという操作を行うことが想定される．このような場合，学習者がブロックの連結を解除する操作を行った部分のみを抽出し，その近辺のプログラミングプロセスを分析することで，どのようにエラーの修正を行ったのかを容易に分析することができる．

3.5　インタフェース

BIVi の主要なウインドウである「プロジェクト・ビュー・ウインドウ」を図 3 に示す．このウインドウは「プログラム表示パネル（左上）」，「操作ボタン，指標表示パネル（右上）」，「タイムラインパネル（下部）」の 3 つの区画から構成される．

「操作ボタン，指標表示パネル」上のボタン，もしくは，「タイムラインパネル」上のタイムラインを操作することで，任意の地点のプログラムの状態を「プログラム表示パネル」上に表示する．任意の地点においてテキストで編集が行われていればソースコードが，ブロックで編集が行われていれば，ブロックの状態が，それぞれ「プログラム表示パネル」上に表示される．

フィルタリング機能操作パネルは「操作ボタン，指標表示パネル」内に設置した．プルダウンメニューでイベントログの種類を選択し，ボタンを押すことで，特定の操作が行われた時のブロックの状態が表示される．

図 3 に表示されている「タイムラインパネル」を拡大したものを図 4 に示す．x 軸は時間経過を表し，y 軸はソースコードの行数を示す．図 4 中の凡例に示したように，学習者のプログラミングプロセスが色分けして表示される．図 4 の例では，テキスト記述型言語で編集されたことを示す水色の縦線の集まりと，ブロック型言語で編集されたことを示す黄色の縦線の集まりが交互に現れていることか

ら，ブロック言語とテキスト記述型言語を行き来して行われる，「まぜ書き」と呼ばれるコーディングが行われていることが観察できる．

4. 実験方法

本実験の目的は，1）画像データを用いたハイブリッド方式により，プロセス復元に時間をかけることなく，スムーズにプロセス分析が可能で，かつ一定のデータ容量に収まること，2) ツール開発者（BlockEditor 開発者）にとって，提案ツールを用いた機能分析が有用であること，の 2 つを評価することである．そのため，以下の 2 つの実験を行う．

実験 I ハイブリッド方式における画像データサイズの測定

実験 II BlockEditor の機能に対する有用性の調査

4.1 実験 I : ハイブリッド方式における画像データサイズの測定

4.1.1 対象

本学におけるプログラミング入門講義の演習課題を用いて実験を行った．2012 年度における対象講義の BlockEditor 利用率と平均行数から，BlockEditor 利用率が 50%以上であり，かつ平均行数が 50 行以上であった課題を対象に実験を行った．課題プログラムは Q2-5「MyFavorite.java」，Q3-3「FigureShop.java」，Q3-4「Omikuji.java」，Q4-6「HundredBoxes.java」，Q5-3「TurtleRace.java」であった．

4.1.2 実験方法

対象プログラムの作成は，第 1 著者が行った．BlockEditor を用いて，対象となる課題 5 題のプログラミングを行い，プログラム作成に要した時間，出力された画像データの枚数，画像データ全体のデータサイズ，課題完成時の行数（lines of code），ブロック数を測定した．

プログラムの作成に要した時間は，最初にブロックに対して操作を行ってから最後にブロックに対して操作を行った時間で計測した．完成時のブロック数は課題終了時の BlockEditor 上のブロックの数を計測した．完成時行数については，課題完成時のブロック言語を Java に変換した際に出力される Java ファイルの LOC を計測した．

4.2 実験 II : BlockEditor の機能に対する有用性の調査

本実験では，提案ツールを用いて，BlockEditor の一機能の有用性調査を行った．実験の目的は，提案ツールを用いてツールの有用性分析が可能であることを明らかにすることである．

4.2.1 動的なスコープチェック機能

本実験で調査対象とする BlockEditor の機能は，動的なスコープチェック機能（VSC:VisualScopeChecker）である．VSC は意味エラーをユーザに提示する目的で実装された機能である．VSC の動作イメージを**図 5** に示す．VSC

図 5 vsc の動作イメージ

はブロックのスコープをチェックし，接続対象のブロックからみてスコープ外の変数を参照するブロックをユーザが接続しようとした場合に，ブロックの連結をせず，ブロックをはじき返すアニメーションを表示する．このとき，変数の宣言元と，連結されなかったブロックを強調表示する．

4.2.2 実験方法

本学のプログラミング入門講義受講中の学生 9 名（以下，学習者とする）に，スコープの理解を問うプログラミング課題を課した後，その解答過程を記録し，BIVi を用いてプロセスの分析を行った．学習者は全 15 回中 14 回目の講義の受講を終了しており，スコープの概念は学習済みであった．

本実験のプロセスを以下に示す．

(1) 学習者によるプログラミング課題の解答
  (a) 実験担当者（第 1 著者）が課題の説明を行う．
  (b) 学習者が BlockEditor の操作方法を十分理解していることを確認する．
  (c) 学習者が BlockEditor を用いて課題に解答する．自力で課題を達成できない場合は，出題者がヒントを段階的に提示することで課題の達成に導く．

(2) 実験担当者，学習者，開発者による解答プロセスの分析
  (a) 実験担当者が BIVi を用いて学習者のプログラミングプロセスを分析する．
  (b) 実験担当者が学習者に対し，BIVi を利用して学習者自身のプログラミングプロセスを提示しながらスコープの理解と VSC の動作についてのインタビューを行う．
  (c) BlockEditor 開発者で，上記二つの分析結果をレビューしながら，VSC の有用性と，改善案の議論を行う．

実験に使用した課題プログラムを**図 6** に示す．課題は繰り返しを用いた「買い物プログラム」である．問題の説明は以下の通りである．

問題

このプログラムは商品を 10 個買うプログラムです．

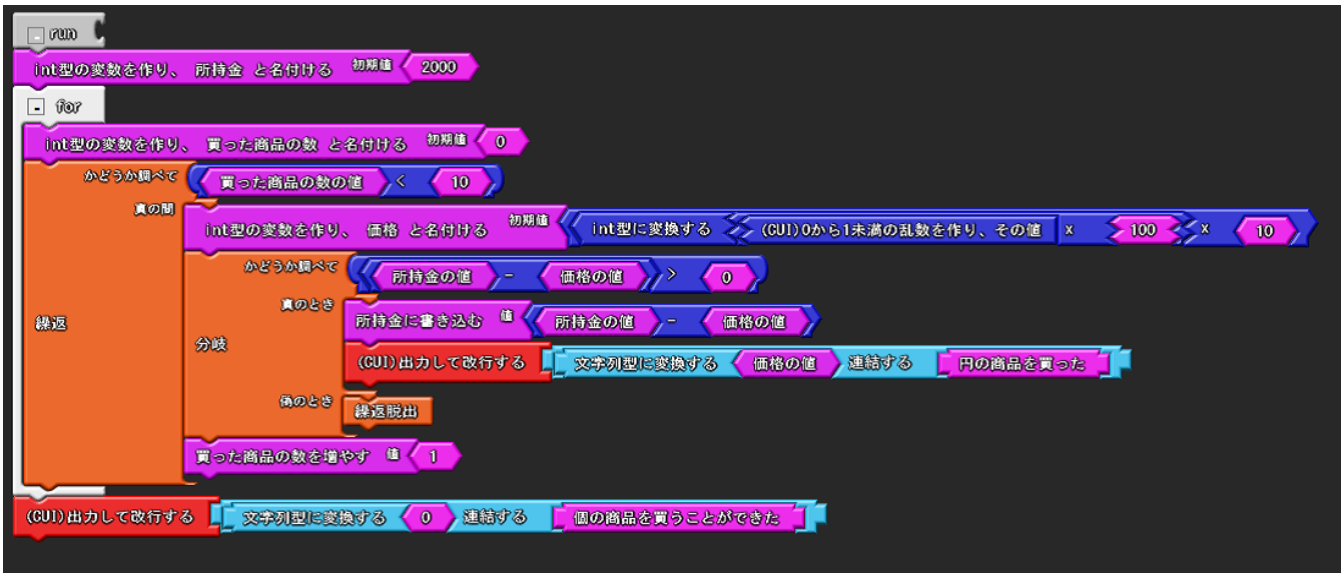

図 6 課題プログラム：買い物プログラム

所持金が無くなれば買うのをやめます．商品の価格は 0 円から 990 円まで毎回ランダムで決まります．既に購入した商品の価格分，所持金が減っていき，所持金が不足すれば購入をやめるところまでは完成していますが，買った商品の数を出力する部分が完成していません．買った商品の数を出力するようにプログラムを編集してください．

課題意図はスコープの理解を問うことである．課題の解答は BlockEditor 上でプログラムを修正することで行う．

VSC の動作前後におけるプログラミングプロセスを観察するため，センサが取得するイベントログの種類に，「VSC によってブロックが反発する動作 (connect_miss)」が追加されている．

学習者に提示するヒントは以下の 3 段階である．

ヒント 1 for と書いた構造化ブロックはどんな役割か

ヒント 2 「買った商品の数」という変数はどこで宣言されているか

ヒント 3 「買った商品の数」という変数を使うにはどうしたらいいか

5. 実験結果

5.1 実験 I：ハイブリッド方式における画像データサイズの測定

画像データサイズの測定結果を**表 1** に示す．完成時 LOC 数と完成時ブロック数の相関係数は 0.99 となり，強い相関があった．画像データ合計と作業時間の相関係数は-0.12 となり，ほとんど相関が無かった．このことから，Java ファイルの LOC 数が分かれば，出力される画像データの合計サイズを推定することが可能であることが分かった．

画像データ合計と LOC 数で単回帰分析を行った結果，画像データの合計を概算する以下の式が得られた．

表 2 取得したデータの分析結果

学習者	合否	connect_miss	time	ヒント回数
A	合	4	31 秒	0
B	合	3	1 分 11 秒	0
C	合	2	3 分 34 秒	2
D	合	3	2 分 50 秒	2
E	合	2	1 分 9 秒	1
F	合	2	2 分 43 秒	2
G	合	3	6 分 15 秒	3
H	合	2	2 分 35 秒	2
I	合	4	7 分 45 秒	1

$$TotalImageDataSize = 0.2LOC \text{ [MB]} \quad (1)$$

5.2 実験 II：BlockEditor の機能に対する有用性の調査

5.2.1 研究者によるプロセス分析結果

取得したデータを**表 2** に示す．学習者は全員課題に合格することができた．学習者 A,B は，ヒント無しで解答し，合格だった．このことからスコープの概念の理解が十分であったと考えられる．

VSC によってブロックが反発した後，観察することができた学習者の行動パターン 4 つを以下に示す．

(1) スコープの誤りに気付き，直ちにプログラムの修正を行う

(2) 手が止まり，ヒントを要求する

(3) ブロックの型が間違っていると考え，ブロックの型の変換を行おうとする

(4) 一度ブロックを分解し，もう一度組み直す

5.2.2 インタビュー結果

「ブロックが反発した時にどう思ったか」というインタビューを行った．インタビュー結果を**表 3** に示す．インタビュー結果から学習者 A，B は VSC が動作した後，変数が参照不可能であったということに加え，具体的な解決方法

表 1 画像データサイズの測定結果

実験対象	画像データ数	画像データ合計 (MB)	完成時ブロック数	作業時間	完成時 LOC 数
FigureShop.java	102	6.0	86	4 分 41 秒	49
Omikuji.java	112	2.1	44	2 分 30 秒	26
HundredBoxes.java	75	2.6	47	6 分 18 秒	32
MyFavorite.java	375	25.6	289	4 分 49 秒	152
TurtleRace.java	93	4.5	72	10 分 16 秒	26

表 3 インタビュー結果

実験者	インタビュー結果
A	これ（変数ブロック）くっつけてみたら（変数ブロックが）赤くなったんで,「あ，何か間違っているんだ」と思って，なんかこれ（変数の宣言ブロック）を（構造化ブロックから）出したらいいんじゃないかなと思いました.
B	あー違うんだなって．このブロック（構造化ブロック）の中に入んないと使えないから，中に入れれば使えるんじゃないかと思った.
C	これじゃあ当てはまらない，形を変えなきゃダメ.
D	使えない変数.（中略）そこでは使えない変数とかだったら，なんかはまらない
E	何か，普通に入れてみて「入らん」ってなっただけ.
F	あー，ヒントで初めて「あー」って（思った）.
G	どっか違うんだろうなって.
H	違うって思った.
I	絶対入らないなって思った.

についても回答している．学習者 C はブロックの形（変数の型）が間違っていると誤解している．学習者 D は参照できない変数であることは理解している．学習者 E，F，G，I，J はブロックが連結しないことの理解にとどまっており，学習者 F は何も思うことはなかったと回答している．

BIVi を用いて学習者のプログラミングプロセスを提示しながらインタビューを行うことで，学習者自身が行ったプログラミングの過程に添って，その時何を考えていかのかを思い出しながらインタビューに答えている様子が観察できた．また，提示されたプログラミングプロセスを指し示しながらインタビューに回答している様子も観察できた．このことから，提案ツールを用いてインタビューを行うことは，1) 学習者の記憶に頼ること無く回答を得ることが可能になること，2) インタビューの進行をスムーズに進めることができること，の 2 つの利点があると考えられる．

学習者はプログラミングプロセスを振り返りながらインタビューを受けることで，スコープの外では値の参照を行えないことや，意図したように値を参照するにはどうすればいいのかを確認した．このことから，学習者はプログラミングプロセスを振り返ることでスコープの概念に対する理解を深めていると考えられる．

5.2.3 開発者による「分析結果の分析」結果

提案システムによって取得したデータ，インタビュー結果を用いて，BlockEditor 開発者による，現状の VSC の有用性の分析を行った．分析は以下のとおりであった．

学習者 A，B はヒント無しで課題に解答し，インタビュー結果から，スコープの概念理解が十分であったと考えられる．このことから，スコープの概念理解が十分な学習者であれば，VSC の動作によってプログラムの修正が十分に可能であると考えられる．

研究者（第 1 著者）によるプロセス分析結果から，VSC の動作後に，変数の型を変換しようとしたり，すでに組まれているブロックを分解したりと，スコープに関係ない操作をしている，というプロセスが観察された．このことから，スコープ概念理解が不十分な学生にとって，VSC の動作は,「スコープの誤り」だと認識させるには不十分であると考えられる．

以上より，現在のデザインでは，スコープの概念理解が不十分である学習者に対し,「変数ブロックが参照不可能であること」,「変数ブロックがどこで宣言されているか」,という情報は与えている．しかし,「変数のスコープの範囲がどこからどこなのか」，については提示できていないという問題が考えられる．そこで，スコープの範囲を示すデザインとして，以下の 3 つのデザインが提案された．

(1) スコープ外となる範囲に変数ブロックを移動させると変数ブロックが半透明にする
(2) 変数ブロックをクリックしている間は，スコープ外にある他のブロックを半透明にする
(3) 変数ブロックをクリックしている間はスコープ内にあるブロックを強調表示させる

6. 考察

6.1 画像データを用いたハイブリッド方式

実験 I によって，ブロック言語を変換した Java ファイルの LOC 数から画像データのサイズを概算するモデルが得られた．モデルを用いて計算したところ，画像データサイズが 100MB を超えるプログラムの LOC 数は約 500 行である．2012 年度の本学プログラミング入門講義の演習問題における LOC は 58 行であり，自由にプログラムを作成させる課題の一部を除けば，プログラムの LOC 数が 500 行を超えるプログラムは存在しなかった．そのため，提案モデルが妥当であれば，提案方式のデータサイズは，当講義の演習課題に適応可能な大きさであることがいえる．

しかし，提案モデルは第 1 著者によって作成されたプロ

グラムを元に作成されたモデルであるため，プログラミング初学者がプログラミング行うことによって出力される画像データサイズと同程度になるとは限らないという問題はある．プログラミング初学者によるプログラミングでは，第 1 著者によるプログラミングと比べ，試行錯誤のプロセスが多くなる．そのため，LOC 数がそれほど多くないプログラムにおいても，出力される画像データサイズが，提案モデルによる概算より大きくなる可能性がある．また，試行錯誤のプロセスが増加するのに伴い，作業時間も増加することから，出力される画像データサイズと作業時間の相関が強くなる可能性もある．

6.2 提案ツールを用いた機能分析の有用性

学習者のプログラミングプロセスの分析を行ったところ，VSC が動作した後のプログラミングプロセスのパターンを明らかにすることができた．このことから，提案ツールには，機能が使われている状況や，機能が動作した後の学習者の行動パターンを明らかにすることが可能であると考えられる．

プログラミングプロセスの分析結果から，スコープの概念理解が不十分な学習者に対しては VSC の構文エラーの提示の仕方は不十分であるという結論が得られた．改善案として，「スコープの範囲を示すデザイン案」が議論された．このことから，提案ツールを用いた機能分析によって，機能の問題点を明らかにし，改善案を導き出すことが可能であると考えられる．しかし，今回は一機能に対する機能分析を実施したに過ぎず，他の機能においても同様に問題点を明らかにできることを示せていないため，他の機能についても分析を行う必要がある．

提案ツールを用いて，プログラミングプロセスを提示しながら学習者に対するインタビューを行った．結果から，学習者がプログラミングプロセスを振り返ることでスコープに対する概念の理解を深めている様子が観察された．このことから，理解度が向上したという直接的なデータは得られていないものの，提案ツールを用いた振り返り学習に，学習効果があるという可能性が導かれた．

7. おわりに

本研究では，画像データとイベントログデータを用いたハイブリッド方式のプログラミングプロセス復元方式を提案し，BlockEditor 開発者による機能分析を行った．実験より，提案方式により生成される画像サイズを概算するモデルを示した．プログラミングプロセスの復元のために提案したツールによって，BlockEditor の動的なスコープチェック機能の有用性の調査を行ったところ，現在の当機能のデザインでは，学習支援能力は限定的であることが明らかになった．そのため BlockEditor 開発者で議論を行ったところ，当機能の改善案を示すことができた．

本研究によって，新たに，プログラミングプロセスを用いたビジュアルプログラミング言語を用いた学習支援環境の分析を行う方法の枠組みが示された．提案方式を利用することによって，学習支援環境におけるプログラミング学習の更なる実態解明がなされることが期待される．

謝辞 本研究は JSPS 科研費 25730203，26280129 の助成を受けたものです．

参考文献

[1] 松澤芳昭，保井元，杉浦学，酒井三四郎：ビジュアル-Java 相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価, 情報処理学会論文誌 55(1),, pp.57-71, 2014.

[2] 松澤芳昭，岡田 健，酒井三四郎：Programming Pro-cess Visualizer：プログラミングプロセス学習を可能にするプロセス観察ツールの提案，情報処理学会情報処理シンポジウム SSS2012，pp. 267-264（2012).

[3] Ingalls, D., Kaehler, T., Maloney, J., Wallace, S. and Kay, A.: Back to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself, Proc. ACM OOPSLA '97, pp.318-326 (1997).

[4] Scratch Team Lifelong Kingdergarten Group MIT Media Lab/: Scratch -imegine.program.share-http://scratch.mit.edu/, referenced at 2015.05.21

[5] Lewis, C.: How programming environment shapes perception, learning and goals: logo vs. scratch, *Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10)*, pp. 346–350 (2010).

[6] Dann, W., Cosgrove, D., Slater, D., Culyba, D. and Cooper, S.: Mediated Transfer: Alice 3 to Java, *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pp. 141–146 (2012).

[7] Caspersen, Michael E., Kolling, Michael: STREAM: A First Programming Process: ACM Transactions on Computing Education, v9 n1 Article 4 Mar 2009.

[8] Charles Boisvert：A Visualisation Tool for the Programming Process，ITiCSE '09 Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education，pp.328-332(2009).

[9] M. Fatih Kksal , R. Emre Basar , Suzan skdarl: Screen-Replay: A Session Recording and Analysis Tool for DrScheme，Proceedings of the Scheme and Functional Programming Workshop, Technical Report, California Polytechnic State University, CPSLO-CSC-09-03, pp.103-110 (2009).

[10] 井垣 宏, 齊藤 俊, 井上亮文, 中村亮太, 楠本真二: 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可視化システム C3PV の提案, 情報処理学会論文誌 54(1), pp.330-339, (2013).

[11] 伏田享平, 玉田春昭, 井垣宏, 藤原賢二, 吉田則裕: プログラミング演習における初学者を対象としたコーディング傾向の分析, 電子情報通信学会技術研究報告 111(481), pp.67-72(2012).