初学者向けプログラミング演習における探索的プログラミングの実態調査と支援手法の提案

Support and Analysis for Exploratory Programming by Novice in Programming Exercise

槇原 絵里奈*　井垣 宏†　藤原 賢二‡　上村 恭平§　吉田 則裕¶　飯田 元‖

あらまし　開発者が不慣れな言語や API を利用する際等に，ソースコードの編集，コンパイル，実行を繰り返しながらプログラミングを進めることが知られている．このようなプログラミング手法を探索的プログラミングと呼び，インタラクティブデバッガ等の支援環境が提案されている．一方で，初学者がプログラミング演習において，探索的プログラミングをどのように行うかは必ずしも分析されていない．そこで本研究では，実際のプログラミング演習において初学者が行っている探索的プログラミングの実態を調査した．さらに，調査結果にもとづいて，初学者向け探索的プログラミング支援環境「Pockets」を提案した．

1　はじめに

社会におけるソフトウェアの必要性が高まるにつれ，優秀なソフトウェア開発者の育成が望まれている．多くの情報系の大学において，与えられた課題にもとづいて学生が実際にプログラミングを行う形式の実践的な授業が実施されている [1]．このような形式の授業のことをプログラミング演習と呼ぶ [2]．

プログラミング演習では，通常まず教員が対象となるプログラミング言語の文法やアルゴリズムの基礎知識を例を交えて学生に講義を行う．その後，教員によって与えられる課題を解くために学生がプログラミングを行う．学生は，教員の解説に含まれる例題を少し変更した簡単な課題や，複雑なアルゴリズムを必要とする発展的な課題まで複数の課題を指定された時間内に解くことが求められる．これらの課題は，最初の数問は教員による解説や解答例が示されることもあるが，基本的には独力で解くことが望まれる．

初学者はプログラミング演習において，条件式の判定基準の変更や文法の確認といった多様な試行錯誤を行うことが多い．このような試行錯誤を伴うプログラミングのことを一般に探索的プログラミング [3] と呼ぶ．探索的プログラミングは，対象の言語や利用する API に不慣れな開発者が行うことがよく知られている [4]．一方で，実際に受講生がどのような試行錯誤を行い，探索的にプログラミングを進めているかは明らかになっていない．そこで我々は実際のプログラミング演習における受講生のプログラミング行動を観測し，探索的にプログラミングを行っていると考えられる事例を 20 件収集し，分析を行った．さらに，分析結果にもとづき，初学者による探索的プログラミングの実態に即した支援環境 Pockets の提案を行う．

*Erina Makihara, 奈良先端科学技術大学院大学
†Hiroshi Igaki, 大阪大学
‡Kenji Fujiwara, 奈良先端科学技術大学院大学
§Kyohei Uemura, 奈良先端科学技術大学院大学
¶Norihiro Yoshida, 名古屋大学
‖Hajimu Iida, 奈良先端科学技術大学院大学

2　探索的プログラミング

実装するプログラムの要求や要求の実現方法が曖昧であるときに，開発者がプログラムの一部分を変更し，その実行結果を確認するというステップを繰り返すことで，プログラムを完成に近づけていくことができる．このようなプログラミング手法のことを探索的プログラミング [3] と呼ぶ．開発者が不慣れな API の利用方法を学習する際等に，探索的プログラミングを行うことが知られている [4]．

探索的プログラミングは Sheil ら [5] によって確認されたプログラミング手法であり，様々な分析・支援が行われている．Yoon ら [4] や Mackenzie ら [6] は，探索的プログラミングの中で行われる手戻りに注目し，開発者がどのように手戻りを行っているかを分析している．

一方で，初学者を対象とした探索的プログラミングの分析はあまり行われていないのが現状である．そこで本研究では，実際の初学者を対象としたプログラミング演習における探索的な振る舞いを収集・分析し，分析結果にもとづく初学者のための探索的プログラミング支援環境の提案を行う．

3　プログラミング演習における探索的プログラミングの分析

3.1　分析環境

探索的プログラミングの分析のために，41 名の学部 1 年生を対象とした Java の初学者向けプログラミング演習 (2 コマ 180 分) において，学生のプログラミング行動を観測した．プログラミング行動の観測には，C3PV [2] を利用した．C3PV はブラウザ上でプログラミングにおいて必要なソースコードの編集，保存（Save），コンパイル（Compile），実行（Run）といった操作を行うことができるオンラインエディタで，学生は C3PV のみで課題の実装・提出を行うことができる．さらに，C3PV には学生のプログラミング行動を観測する機能がついており，Save，Compile，Run といった操作を学生が行った時刻とそのときのソースコードそのものがログとして保存される．また，ソースコード編集中にも 1 分ごとにソースコードそのものが保存されるようになっている．なお，Run 実行時には Compile も自動的に行われている．

3.2　初学者における探索的プログラミング実態

今回の分析では学生が同一課題の同一行を 4 回以上連続して変更している場合に探索的にプログラミングを行っているものとして抽出した．分析の結果，41 名のログから 16 名が行っていた探索的プログラミング 20 件を特定した．そのうちの 1 件について図 1 に示す．

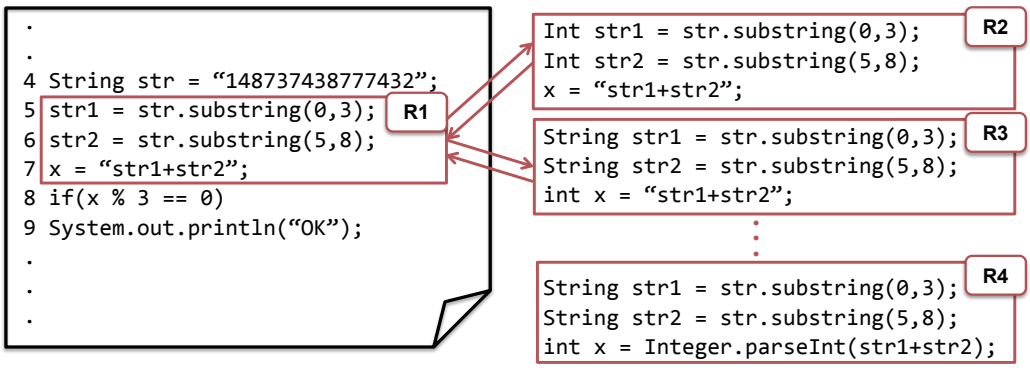

図 1　初学者における探索的プログラミングプロセス

この例では，学生は subString と parseInt という API を利用して，文字列を数値に変換した後に指定された計算を行う課題を解いている．R1 が今回特定した学生の探索的な振る舞いの最初の状態である．この学生は R1，R2，R1，R3，R1，R4 という順でソースコードの編集を行った[1]．この例では，subString の返り値の型が String 型であることや String 型の返り値をどうやって int 型の値に変化すれば良いかを学生が理解しておらず，試行錯誤をしている状況が確認できた．この学生の場合，最終的に R4 にたどり着くまでに 12 分間かけて 16 回 Run コマンドを実行し，また，そのうち 5 回は過去のソースコードへの手戻り [4] を含んでいた．手戻りを行っている事例はこの例を含めて 6 件観測された．

今回の分析では，探索的プログラミングを行っていると判断した複数の連続したソースコードのうち，最初のソースコードが保存された時刻から最後のソースコードが保存されるまでにかかった時間を探索時間と定義する．今回の分析により得られた探索時間のうち，最小は 3 分，最大は 38 分，平均は 12.1 分だった．今回の分析では同一行に対する 4 回以上の連続した変更のみを対象としているため，実際にはより多くの探索的プログラミングが実施されていた可能性が高い．実際に，6052 件のログのうち，既存研究 [5] で述べられている探索的プログラミングの基準の一つである編集，コンパイル（あるいは実行）を複数回繰り返している事例を調べたところ，5 分以内に Compile あるいは run をやり直しているログが 1128 件存在した．すなわち，探索的プログラミングは初学者によって非常に数多く行われている可能性が高い．探索的プログラミングはプログラマの開発技術を向上させる手法の 1 つであると既存研究 [3] でも述べられており，その支援は重要であると考えられる．

学生がコンパイルや実行を短い時間の間に繰り返して行って結果を確認していることや，何回も手戻りを行っていることから，支援ツールには以下の要件が必要であると考える．

要件 1 過去のソースコードのコンパイルや実行結果が一覧で分かること．

要件 2 過去のソースコードと現在のソースコードの差分，つまり自分がソースコードに加えた編集と，それによる過去と現在の結果の差分が分かること．

要件 3 過去の特定のソースコードへ簡単に戻ることができること．

これらの機能は Git や Subversion と言った版管理システムを用いることで解決できるが，このようなシステムは経験のある開発者向けであるため初学者は操作を行うことが難しい．そこで，我々は以上のような実態にもとづいて初学者による探索的プログラミングを支援するプログラミング環境 Pockets を提案する．

4 探索的プログラミング支援環境 Pockets

初学者が探索的にプログラミングを行う際，ソースコードが完成していない状態でも Compile や Run を何回も実行し，必要に応じて手戻りをしつつその結果を確認しながら，探索を進めていくということが分析によって分かった．そこで我々は Pockets の機能として，ユーザによって Save，Compile，Run が実行されるたびに，出力結果に関係なくソースコードの状態と出力結果両方に対する版管理を自動的に行う仕組みを導入する．以降では Pockets の UI イメージを利用し，Pockets において実現される版管理の仕組みを詳述する．

4.1 Pockets の UI イメージ

図 2 に Pockets の UI イメージを表示する．Pockets の UI は大きく左右に分かれており，左が実際にプログラミングを行うエディタ領域，右が版管理されたソースコード及び出力結果が管理される版管理領域となっている．エディタ領域の上部には Save，Compile，Run，Submit といったソースコードに関する各機能を実施するボタンが配置されており，下部には Compile，Run といったコマンドの実行結果を

[1] 実際には各状態の間に複数回の異なる変更が含まれている

表示するコンソールが備わっている．

版管理領域は画面上部のリビジョン一覧表示領域とユーザが選択したリビジョンのソースコード及び出力結果の差分を表示する差分表示領域から構成されている．ここでリビジョンとは，Save，Compile，Run がユーザによって実行された際に保存されたソースコード及び出力結果を指すものとする．以降では版管理領域の詳細について述べる．

4.1.1 リビジョン一覧表示領域

リビジョン一覧表示領域には，ユーザが Save，Compile，Run のいずれかの操作を行うたびに自動的に新たなリビジョンが追加される．この領域では，ユーザがいつ，どのような操作を行ったのかを示すことができる．特に，Compile あるいは Run が実行されたときには，Compile Error や Runtime Error の有無が確認できるようになっている．図 2 の場合，リビジョン番号と時刻が 1 つの四角の中に表示されている．四角の枠線には実線と点線があり，実線はそのリビジョンが差分表示領域に表示されていることを示している．枠線の色は Save，Compile，Run のいずれが実行されたのかを示しており，それぞれ黄色，青，赤の色と対応している．四角内右上に表示されている ○ 及び × は Compile あるいは Run が実行された際に，Compile error もしくは Runtime error が生じた（異常終了）か生じなかったかを示している．すなわち，○ の表示はプログラムが異常終了しなかったことのみを示しており，必ずしも正しい結果が出力されたとは限らない．

4.1.2 差分表示領域

差分表示領域には，4 つ分のリビジョンを表示する領域があり，左上，右上，左下，右下の順に新しいリビジョンが表示される．ユーザが任意のリビジョンをリビジョン一覧表示領域から選択するか，ユーザの Save，Compile，Run 等の操作によって新たなリビジョンが追加されるたびに，版管理領域内中央の差分表示領域内に対象リビジョンの情報が表示される．なお，リビジョンは 4 つしか表示できないようになっており，既に 4 つ表示されている場合には最も古いリビジョンが削除され，新し

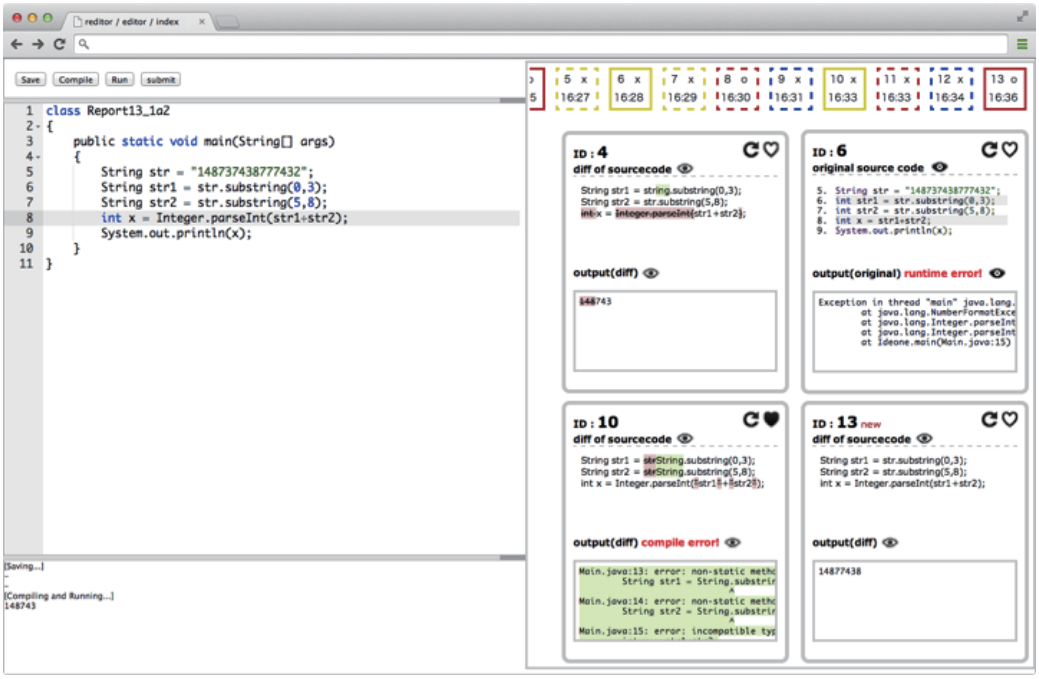

図 2　Pockets の UI イメージ

表 1　Compile，Run を伴う探索的プログラミングの例

ID	操作時刻	操作種別	ソースコード	出力結果
1	18:01	Compile	60:int i = this.Gas; 61:while(i < 0){	cannot find symbol
2	18:03	Run	60:double i = getGas(); 61:while(i < 0){	0km 走行しました。
3	18:05	Run	60:double i = getGas(); 61:while(i = 0){	incompatible types
4	18:09	Run	60:double i = getGas(); 61:while(i > 0.0){	（出力なし）
5	18:12	Compile	60:while(car.getGas() > 0.0){	300km 走行しました。

いものが先頭に追加される．差分表示領域における各リビジョンの領域では，ソースコード・出力結果の差分あるいはオリジナルのソースコード・出力結果の表示切り替え，エディタ領域に表示されているソースコードの対象リビジョンへの差し替え（手戻り），特定のリビジョンの固定，を行うことが可能となっている．

5　ケーススタディ

本ケーススタディでは，Pockets を利用する際の一般的な流れについて説明した後，3.2 節で述べたような既存の探索的プログラミング事例において Pockets がどのような点で有用かを検証する．

Pockets は 3.1 節で述べた C3PV がベースになっている．そのため，ユーザは図 2 に示す Pockets のページを開くだけで，対象課題のプログラミングを開始することが可能となる．つまり，ユーザは既存のエディタに対する操作と同様に開発を続けるだけで，Pockets による版管理機能を利用することが可能となる．

表 1 に探索的プログラミングと判断した 1 件のログの一部を示す．この表では，Compile，Run といった操作種別とその操作を実施した時刻，変更されたソースコードの一部，その操作によって得られた出力結果が示されている．

この事例では，コンパイルエラーや異常値の出力を繰り返しながら探索を行い，最終的に求められていた結果と同じ出力を行うソースコードを完成させている．このような探索的プログラミングでは，Pockets を利用することで過去 4 リビジョンにわたってどのような変更を行い，その結果がどうなったかを把握することが可能となる．特にこの事例のように条件判定の内容を順に探索しているようなケースでは，自分が過去に実施した変更とその結果を容易に確認できるようになることで，探索をより効率よく進められるようになる可能性がある．

また，探索的プログラミング中は 3.2 節で述べたように手戻りが発生する場合や，変更とコンパイルや実行を繰り返すことによりソースコードが複雑になり学生が混乱する可能性が存在する．しかし，Pockets では過去のソースコードの結果を一覧することができ，また特定のソースコードへの手戻りも簡単に行うことができるため，学生は混乱することなくコーディングを行うことができると考える．

さらに，教員や TA は学生がそれまでどのような試行錯誤を行っていたか知ることで，安易に解答を学生に教えるだけでなく，出力されたエラーメッセージと変更内容の対応を説明したり，学生の変更内容が解答に近づいているかどうかを評価したりといったよりきめ細かいサポートを行うことが可能となると考えられる．

6　考察

初学者による探索的プログラミングの分析により，版管理の機能を組み込んだエディタが有用であると判断し，主要機能として以下の 3 つを組み込んだオンラインエディタ Pockets を提案した．

F1: Save，Compile，Run といった操作に連動したソースコード及び出力の自動保存
F2: 自動保存された各リビジョンと現在プログラミング中のソースコード及び出力との差分表示
F3: 任意のリビジョンに戻ることができる Revert 機能

これらの機能の一部は既存の版管理システムにも導入されているが，初学者が利用することは容易ではない．特別に意図せずに普段通りにプログラミングを行うだけで，リビジョンの保存や表示が行われる機能が初学者にとって重要であると考えられる．また分析によって，初学者はプログラムが必ずしも完成していなくても Compile や Run を実行し，Compile Error 等の出力を確認することが観測された．そのため通常の版管理機能だけでなく，プログラムの出力も含めてリビジョンとして保存し，振り返ることができるようにすることで，わざわざ手戻りを手動で行った後に再度実行して出力を確認するといった対応をとる必要がなくなると考えられる．

F1∼F3 の機能は学生だけでなく，教員にとっても有用である．通常，学生が教員にサポートを求めた場合，教員は学生がサポートを求めた時点でのソースコードを確認し，必要に応じて Compile，Run を行い，状況を把握する．その後，学生のその時点におけるソースコード中の間違いを指摘し，解答あるいは解答に繋がるヒントを提示することでサポートを行う．一方，Pockets を用いた場合，教員は学生による試行錯誤の変遷を容易に確認することが可能となる．具体的には，基礎的な文法上の間違いなのか，アルゴリズムに関する間違い（条件文の間違い等）を繰り返しているのか，といった間違いの傾向や，過去のリビジョンでどのようなエラーメッセージを出力しているのかといった情報をサポートに用いることが可能となる．結果として，過去のリビジョンにおけるエラーメッセージの説明や解答に近づくような変更ができているかといった，学生個々人の理解度や開発の経緯に応じたサポートを教員が行えるようになると考えられる．

7 おわりに

本稿では，初学者による探索的プログラミング事例を分析し，初学者でも容易に版管理機能を利用可能なオンラインエディタ「Pockets」を提案した．今後は実際のプログラミング演習に Pockets を適用し，学生の探索的プログラミングをどの程度支援できるかを定量的・定性的に評価していきたい．

謝辞 本研究は日本学術振興会 科学研究費補助金 24700030，26540182，26730036 の助成を受けている．

参考文献

[1] 独立行政法人情報処理推進機構. IT 人材白書 2012. 独立行政法人情報処理推進機構, 2012.
[2] 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過程可視化システム C3PV の提案. 情報処理学会論文誌, Vol. 54, No. 1, pp. 330–339, January 2013.
[3] David W. Sandberg. Smalltalk and exploratory programming. *ACM SIGPLAN Notices*, Vol. 23, pp. 85–92, October 1988.
[4] YoungSeok Yoon and Brad A. Myers. An exploratory study of backtracking strategies used by developers. In *Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 138–144, June 2012.
[5] Beau. Sheil. Environments for exploratory programming. *Datamation*, Vol. 29, No. 7, pp. 131–144, July 1983.
[6] I. Scott MacKenzie and R. William Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. *Human computer interaction*, Vol. 17, pp. 147–198, 2002.