N-005

多様な活動の履歴を用いたプログラミング学習支援環境 HALO Tools の提案

HALO Tools to Support Self-Reflexive Program Learning by Using the History of a Variety of Activities

渡部 丈† 中小路 久美代† 山本 恭裕†
Jou Watabe Kumiyo Nakakoji Yasuhiro Yamamoto

1. はじめに

インターネット上では，プログラミングに取り組むためのパッケージやソースコードが数多く公開されている．Python におけるパッケージを例にとると，データ読み込みのための Pandas[1]や機械学習のための scikit-learn[2]などが挙げられる．

プログラマは，目的とする実装のために必要だと思われるパッケージやアルゴリズム，ソースコード等を検索し，検索結果の中から使えそうなものを選び出し吟味する．コーディングに取り組み，場合によっては発生したエラーについてデバック作業をおこなう．このような活動を繰り返し，ソースコードを完成へと近づけていく．完成までの過程において，プログラマは専門用語やパッケージ，文法といった，プログラミングに関する技術的な知識を獲得する．加えて，「ここはこうしたら失敗する/うまくいく」といった，プログラミングにおける経験則的な知識を修得していく．

本論の問題意識は，このようにしてプログラマに蓄積されていく知識と経験は，その後に取り組むプログラミングにおいては必ずしも活かせていないのではないかという点にある．例えば，ソースコードの編集について考える．様々な調査，思考を経て完成に向けてソースコードを編集していくが，編集の度にバージョンを変えるといった工夫をしない限り，完成形のソースコードが残るのみである．うまく実装できた結果のソースコードだけを見ても，そこから編集過程等を読み取ることはできない．我々は，獲得した知識を活用可能な資源として蓄積し，必要な場面で引き出せるようなプログラミング環境が有用であると考えている．

本研究では，プログラミングにおいて自己の経験を十分活かせていないという課題に対して，個人の履歴に着目したプログラミング支援のアプローチを採る．我々は，学習者のプログラミング活動によって生じる，プログラムのエラーやソースコードの履歴，学習者自身によって作成されるメモ等の，必ずしも明示的に履歴として残らないデータを蓄積し提示するプログラミング学習支援環境，HALO (History-based Augmenting Learning Opportunities) Tools の構築に着手してきた[10]．HALO Tools は，エラーの履歴や過去の自分が書き溜めたファイルをテキストデータとして可視化したり，過去のプログラムやメモデータから必要な情報を取り出したりすることのできる環境である[10]．

†公立はこだて未来大学 Future University Hakodate

本稿では，プログラミングにおける多様な活動の履歴の収集と，履歴を活用したプログラミング学習の様相について述べる．2 章では関連研究について説明する．3 章ではプログラミングで蓄積できる履歴データについて説明し，4 章では一人称研究を通して実装した HALO Tools について説明する．5 章では HALO Tools を用いた観察実験について述べ，今後の展望について論じる．

2. 関連研究

本研究では，プログラミング学習は「過去に学んだ自分」と「今学ぶ自分」の対話的関係によって進めていくことができる，という立場を採る．この考え方は，Cockburn (1991) が論じた，Reflexive CSCW (Computer Supported Coordination of Work) の考え方に依るものである[3]．Cockburn は，複数のタスクを持つユーザを，タスクごとに別人格として捉えることができると主張した[3]．CSCW (Computer Supported Cooperative Work) 研究の多くでは，他者との協調的作業を対象とするが，Reflexive CSCW は個人に着目した考え方である．本研究では，この Reflexive CSCW の考え方を取り入れ，「過去に学んだ自分」というのは「知識(情報)を与える存在」であり,「今学ぶ自分」というのは「知識(情報)を求める存在」と見なせるものというアプローチを採る．

Ronchetti (2018) は，講義中にとったメモと講義の録画ビデオを，時間関係に基づいて同期して統合するシステムを設計・構築した[4]．メモを取っても，それを見直した時に文脈を思い出せず，メモ自体の意味が失われてしまうこともある．メモとビデオを同期させることで，ビデオに付けられたメモは意味をもったマーカーとなり，メモ自体の意味を失うことなくコンテキストを再構築しやすくなる．過去にプログラミング活動をした自分が，何を，どのような文脈で悩んでいたのかということを想起するのは難しい．本研究でも，過去に残したテキストメモ等を履歴情報として活用することを考えており，Ronchetti と同様の立場を採る．

本研究では，プログラミング中に発生したエラーは，次のプログラミングに向けた重要な フィードバックであると考えている．発生したエラーの履歴を閲覧して今のプログラミングに反映させる，ということを繰り返すような学習の仕組みを提案した知見ら(2005) の研究の考え方[5] を，本研究では取り入れた．知見らは，プログラミング学習において発生するエラーについて，失敗学の考え方により，「事象」「背景」「経過」「原因」「対処」「総括」の 6 項目に分類した．実際にエラーが発生した際に，学習者に

エラーを項目に分けて記述させるプログラミング学習環境を構築した[5].

3. プログラミング学習の様相

一概にプログラミングといってもその過程では多様なスタイルが存在する．本節では，プログラミングに伴う様々なタスクや作業について説明する．

3.1 プログラミングに伴う多様なアクティビティ

プログラミングには様々なアクティビティが伴う．実装のために用いる言語の選定をおこない，実装に必要なアイデアを出し，コーディングを進めていく．コーディングの過程において，必要なアルゴリズムやその言語で提供されているパッケージ等について，Web 検索等をおこないながらコードを記述する．記述したコードを実行しエラーが発生したならば，デバック作業をおこなう．検索して得た知識やエラーの解決方法について何らかの媒体で記録するという活動もあるかもしれない．

これらの活動は，順序や頻度など様々な形でプログラミング中に発現する．

3.2 プログラミング環境 Jupyter Notebook

使用するプログラミング環境によって，プログラミングのスタイルは異なると考えられる．その一例として，既存のプログラミング環境である Jupyter Notebook を用いて説明する．

Jupyter Notebook は，Project Jupyter というプロジェクトにおいて設計，開発されてきた，対話的なプログラミングを行うための Web アプリケーションである[6]．Jupyter Notebook 専用の，ipynb ファイルと呼ばれる実行ファイルには，実行したいソースコード， Markdawn，数式，といったものなどを記述することができる．図 1 に Jupyter Notebook のスクリーンショットを示す．

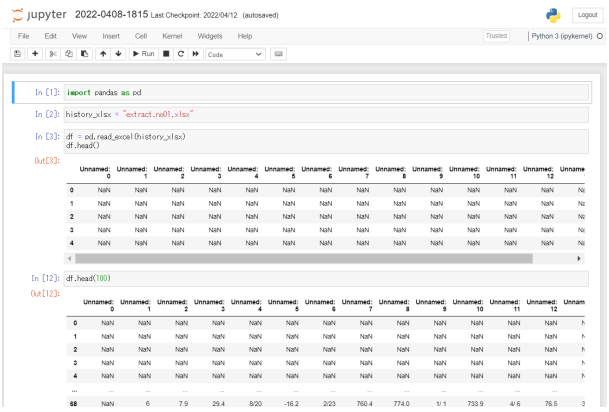

図 1　Jupyter Notebook のスクリーンショット

Jupyter Notebook には，コードセルと呼ばれるソースコードの入力エリアを，単一ファイル内で複数生成できるという特徴がある．通常，一つのソースファイルに記述される内容には意味的な区切りにはなく，一つのまとまりとして処理される．Jupyter Notebook ではいくつものプログラムのまとまりを作ることが可能であり，それぞれ違う単位として実行することができる．

この特徴によって，コードセル毎にプログラムの要素を分割して作成したり，コードセル毎にプログラムのバージョンを変えて作成したりするといったことができる．このように，Jupyter Notebook を用いると，少しずつソースコードを実行して，試行していきながら，最終的な目標の状態を目指すというプログラミングのスタイルが可能となる．

4. プログラミング支援環境 HALO Tools

4.1 プログラミング活動データの収集

本研究では，プログラミング活動は人がおこなう創造的な知的行為であるとの立場を採る．個人の閉じた情報環境において行われるプログラミング活動の文脈や，どのような情報を用いて結論としてのソースコードが記述されていくのかを捉えることは容易ではないと考えている．一人称研究の手法を採用して，まずは，一人のプログラミング活動を観察し，データ収集をおこなった．一人称研究は，研究者自身の体験をもとにして研究をおこなうという，研究の方法論である[7][8].

データ収集期間は，2021 年 9 月 6 日から 10 月 6 日の間であった．その期間内で計 11 日分の学習履歴データを収集した．プログラミング学習環境には，Jupyter Lab[6]を用いた．Jupyter Lab は，Jupyter Notebook の基本コンセプトを継承しつつ，アプリケーションとしての機能をより充実させた，Web ブラウザ上で動作するプログラム実行環境である．学習に用いたプログラミング言語は Python であった．

プログラミングにおいて収集・蓄積できる履歴として，以下の六種類の学習データに着目した．

(1) Web ブラウザでの検索履歴
(2) プログラム実行時に発生するエラーのログ
(3) プログラミング学習をする筆者の様子を録画したビデオ(三視点)：
(4) プログラミング学習で作成したソースコード
(5) プログラミング学習を撮影したビデオに対してアノテーションしたテキストメモ：
(6) プログラミング中に記述したテキストメモ

表 1 に，プログラミング学習データの概要を示す．

4.2 収集したデータについての考察

ビデオに対するアノテーションを通して，プログラミング中の学習者の様子を明示的に文字として書き起こした．書き起こした内容には，「あれ，なんのエラーだ」といったエラー文に対する疑問や「三，四行目が円グラフを作るためのデータだ」というプログラムに対する理解，「あ，まずは CSV を読み込まないと」という気づき，等の記述が見られた．

このように，理解した内容やわからなくて調べたこと，知見として得られたことは暗黙的に学習者の中に存在する．これらを明示的に記録し，学習者が振り返ることでプログラミングに対する理解が深まるのではないかと考えられる．

ただし，これらはその時に感じたことをそのまま書き下しているため，かなり口語的で簡略化されて記述されている．後に学習者が振り返りとして使用する時に，文脈や詳細な状況がわかるように表現されている必要があると本研究では考えている．

表 1　プログラミング学習データの概要

	記録形式	合計ファイルサイズ（収集ファイル数）	データ収集のタイミング <学習中>	<学習後>
Webブラウザ (Chrome)の検索履歴	.csv	173 KB (10 個)		✓
Pythonコンパイラが出力するエラーログ	.log	40 KB (10 個)	✓	
ノートPCの画面を記録したビデオ	.mp4	161 MB (6 個)	✓	
学習者の様子を俯瞰視点で記録したビデオ	.MOV	3 GB (6 個)	✓	
キータイピングを記録したビデオ	.MOV	18 GB (6 個)	✓	
Jupyter Labで保存されたソースコード	.ipynb	2 MB (13 個)		✓
学習中の素直な所感を記述したテキストメモ	.txt	12 KB (6 個)		✓
第三者視点での観察を記述したテキストメモ	.txt	18 KB (6 個)		✓
パソコン上で作成したテキストメモ	.txt	12 KB (6 個)	✓	
プログラム作成時の思考過程に関する手書きメモ	.jpg	4 MB (2 個)	✓	

4.3 プログラミング学習支援環境 HALO Tools の設計とプロトタイプ構築

事例観察を踏まえて，プログラミング学習支援環境 HALO (History-based Augmenting Learning Opportunities) Tools の構築に着手している[10].

HALO Tools は，多様なプログラミング活動の履歴情報を表示するブラウザである HALO-Browser と，ソースコードエディタとしての役割を担う HALO-CodeEditor から構成されるツール群である[10]．図 2 に HALO Tools の概観を示す.

図 2　HALO Tools の概観

HALO Tools は，履歴情報の閲覧とコーディングを同一環境でおこなうというコンセプトで設計されている．多様な履歴情報を閲覧することで，過去のプログラミングにおける学習内容を想起したり，新たな気づきを得たりできる可能性があると考えている[10].

HALO Tools が提示する履歴情報は，「プログラミング中に検索した，Web ブラウザの検索履歴」，「ソースコードの変更履歴」，「プログラミング中に発生したエラーの履歴」，「学習者が記述するメモ」の四つである.

提示される四つの履歴情報は，それぞれの履歴が生成された際のタイムスタンプをもつ．タイムスタンプをもつことによって，各履歴情報同士は時間連携が可能となり，多様な履歴を関連づけて閲覧することができる.

HALO-CodeEditor は，HALO Tools 起動時に表示される，主に Python のソースコードを記述・編集・実行するためのウィンドウである．図 3 に HALO-CodeEditor を示す.

ソースコードの記述は，文字入力ができるテキストエリア内でおこなう．テキストエリアの直下にある「実行」ボタンを押下することによって記述したソースコードを実行することができる．HALO Tools の上部には，「ファイルを開く」「上書き保存」「名前を付けて保存」というボタン群があり，既存の Python ファイルを開いたり，既存の Python ファイルの編集，新規 Python ファイルの作成をおこなえる.

他の機能として，HALO-CodeEditor 上で編集したソースコードの履歴や実行したソースコードのエラーの履歴について閲覧することができる．さらに，エラー履歴表示スペースの直上にある「情報」ボタンを押下することで，発生したエラーの種別毎の出現回数が示されたポップアップが表示される.

HALO-CodeEditor は，プログラミング中に書き留めたいことをメモしておける機能を有する，プログラミングメモ記述スペースも用意している．この機能は，既存のシステムである TextStep を HALO-CodeEditor 上に埋め込むことによって実現している．TextStep は，中小路ら(2019)が行った研究で構築された，タイムスタンプ付きのテキストメモをとるためのブラウザベースのツールである[9]．Web ブラウザ上で TextStep にアクセスすると，現在時刻と矩形のテキストエリアが表示される．テキストエリア内に文字を入力し Enter キーを押すと，タイムスタンプ付きのデータとして Web ブラウザの LocalStorage に保存される．蓄積されたデータは，「.md 形式」のファイルとしてダウンロードできる.

HALO-Browser は，HALO-CodeEditor の「HALO-Browser」ボタン押下時に遷移する，三種類の履歴情報を閲覧することができるウインドウである．図 4 に HALO-Browser を示す.

HALO-Browser は，検索履歴，ソースコード編集履歴，プログラミング中にメモについて閲覧することができる．検索履歴は，HTML における表として出力される．年月日を指定するテキストボックスに，年月日時を決められたフォーマットで入力し，「検索」ボタンを押下すると，該当する検索履歴が表示される．図 4 に示されている表の一行一行が，一回ごとの検索データである．ソースコードの編集履歴には，これまでにおこなってきたプログラミングの編集活動が全て記録されている．TextStep で記述した内容を，プログラミング中のメモとして表示する．今のところ，予め TextStep からログファイルをダウンロードし，HALO Tools に手動で読み込ませておく必要があるため，自動化は実装課題である.

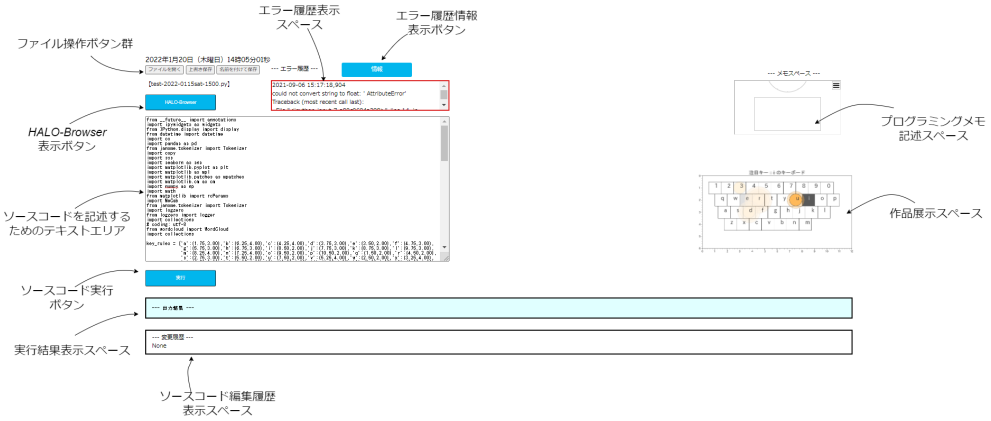

図 3　HALO-CodeEditor

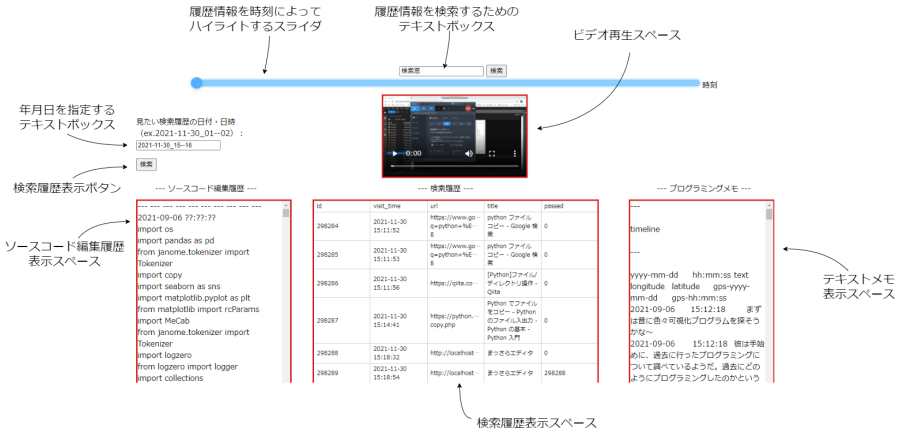

図 4　HALO-Browser

加えて，HALO-Browser における履歴閲覧のための機能として，語句検索機能を実装した．HALO-Browser の最上部に位置するテキストボックスに任意の文字列を検索ワードとして入力することにより，各履歴情報に存在する該当文字列をハイライトする機能である．表の場合，該当文字列があるセル自体が青く塗りつぶされる．テキストデータの場合，該当文字列の背景部が青く塗りつぶされる．

その他の機能として，0 時から 24 時の 1 時間刻みのスライダを左右に動かすことによって，対応するタイムスタンプをもつ履歴をハイライトする機能や，プログラミング活動を記録したビデオ再生する機能の実装も試みたが，未完成となっている．

5. HALO Tools を用いた事例観察

本章では，実際に収集したプログラミングに関する活動の履歴を HALO Tools に取り込んで実施した事例観察と観察を通した考察について述べる．

5.1 ケーススタディ 1：一人称による事例観察

観察対象を著者自身として，一人称事例観察を実施した．本観察では，プログラミング履歴ブラウザ HALO-Browser とソースコードエディタ HALO-CodeEditor を用いたプログラミング学習における課題を抽出することを目的とした．

本観察は，2022 年 1 月 14 日，1 月 15 日の 2 日間に渡っておこなった．本観察のプログラミングに用いるタスクとして，2021 年 9 月から 10 月にかけて実施した一人称事例観察時のタスクを採用した．

HALO Tools は，データブラウジング環境とプログラミング学習環境の両方の側面をもつ．事例観察は，

- HALO-Browser にプログラミング履歴の実データを挿入し，表示されたデータの閲覧
- HALO-Browser による履歴閲覧を伴う，HALO-CodeEditor 上でプログラミングの実践

の 2 部構成で行うこととした．

5.1.1 観察概要

HALO-Browser 上の履歴データの閲覧は，閲覧時間を 10 分間 1 セットとし，2 セット実施した．閲覧中の HALO-Browser 画面と，背後から筆者の様子を録画した．音声も一緒に記録するために，なるべく思ったことを声に出すようにした．気づいたこと，感じたことがあれば，パソコン上のテキストファイルにメモとして記述した．

HALO-CodeEditor 上でプログラミングは，履歴閲覧と同様に 10 分間 1 セットとし，計 3 セット実施した．こちら

も同様に気付き等をメモし，プログラミングタスク中に考えたことは発話するようにした．

5.1.2 HALO-Browser でのデータ閲覧

現状の HALO-Browser は，ただ三種類のデータが横並びに並べられているだけであり，時系列の近い履歴データの比較が困難であることがわかった．具体的には，

- 近しい時系列で何が起きているのかあまりよくわからない
- 三種類の履歴データが表示されているが，同時系列でデータを並べての比較がしづらい．対応関係をもっとみたい

といった課題が挙げられた．

HALO-Browser の語句機能を用いて，タイムスタンプで履歴検索をおこなえば，各履歴データの時間関係での連携は可能である．その一方で，より適切な，時系列による履歴データ比較の方法を模索していかなければならないと考えている．例えば，粒度の細かい時間軸のスライダを用いたインタラクティブな履歴表示を実現する機能の実装などが考えられた．

HALO-Browser 上の学習履歴を閲覧することによって，過去に取り組んだプログラミングの内容を推測するというアクティビティが見られた．そのアクティビティの中でメモとして記述された内容の一部を紹介する．

- 検索履歴からプログラミング内容を推測しようとした
- 日付を検索ワードに用いて該当するソースコードの箇所やメモを閲覧し，そこからさらに絞り込んで検索ワードを考えた
- 検索履歴やメモの，「ループ処理」「スライダー」「グラデーション」といったワードから，動的に色を変えることをやりたかったのだろう，と推測した

HALO-Browser の検索機能を用いることによって，過去のプログラミング内容について想起出来ていたが，その時の文脈の話は語られていなかった．これは，特定の時間軸に着目して閲覧したデータに，プログラミングメモの履歴が存在しなかったことが原因として考えられる．

HALO-Browser の UI 面の検討課題として，表示される履歴の文字化けや履歴表示のための入力トリガの指定方法，文字情報の多さが挙げられた．欲しい機能として，検索履歴の URL リンクにアクセスできること，ソースコード編集履歴にあるコードの実行結果も見られること等が挙げられた．

5.1.3 HALO-CodeEditor 上でのプログラミング

HALO-Browser 上で表示された履歴を用いて，HALO-CodeEditor でプログラミングをおこなった際の気づきについて以下に一部を示す．

- HALO-CodeEditor 上でのプログラミング学習には，プログラミングメモとソースコード編集履歴を主に用いた
- HALO-Browser 上で，語句検索のワードを「円→circle」と変えながら調べている
- ソースコード編集履歴から，該当するプログラムをコピーして現プログラムにペーストした

HALO-Browser のみを用いたデータブラウジング時は主に検索履歴とソースコード編集履歴に着目していた．一方，HALO-Browser と HALO-CodeEditor を用いたプログラミングでは主にプログラミング中のテキストメモとソースコード編集履歴を閲覧していた．ソースコード編集履歴から過去のソースコードをコピーして，HALO-CodeEditor 上で実行する，といった活動が見られた．

特に着目すべき点は，語句検索に用いたワードの変遷にある．初めはプログラミング中のテキストメモから情報を得るために「円」と検索しているが，参考になる情報は得られなかった．次に，ソースコード編集履歴から情報を得ることにして，ソースコードの文脈に適したキーワード「circle」に変更した．このような思考の過程の変遷があることがわかり，今後の HALO Tools の開発に役立つ可能性がある．例えば，HALO-Browser における語句検索に用いたワードの変遷を，HALO Tools 上のアクティビティの履歴として蓄積していく仕組み等が考えられる．

5.2 ケーススタディ 2：実験参加者による事例観察

本学大学生 1 名を対象にプログラミング観察実験を実施した．プログラミング活動を観察し，プログラミングに伴う活動履歴のデータを収集・蓄積した．

観察実験から 33 日後の 2022 年 6 月 14 日に，収集した履歴データを HALO-Browser に取り込んで，実験参加者に閲覧してもらった．HALO-Browser 上で履歴をどのように閲覧しているのか，履歴閲覧を通してプログラミングの内容についてどの程度思い出せているか，といったことの調査を目的とした．

5.2.1 履歴閲覧の概要

事前に，実験参加者に対して，一か月前のプログラミングの内容について簡単なインタビューを実施した．実験参加者には，「履歴を見て 1 か月前の状況を思い出せるか」「自分の履歴を見て気づくところ，思うところはあるか」という観点で，HALO-Browser で履歴を 10 分間閲覧してもらった．履歴閲覧中は，思ったこと・気づいたことをなるべく声に出してみるように教示した．履歴閲覧終了後には，「5 月に取り組んだプログラミングの内容を思い出せたか」「どの履歴データを見て何を考えたか」といった内容で半構造化インタビューを実施した．

5.2.2 履歴閲覧の結果と考察

履歴閲覧前のインタビューで，「一か月前に取り組んだプログラミングの内容を覚えているか」という質問をした．実験参加者は，ある程度自分の取り組んだ内容について覚えているようだった．プログラミングの出来についても，「惜しいところまではいったような気がする」と述べていた．一方，実験後インタビューで「一か月前に取り組んだプログラミングの内容を思い出せたか」という質問に対して，

- 実装がうまくいった部分については覚えていた．
- うまく実装できていなかったことについてはあまり覚えていなかったが，ソースコード等の履歴を見たら思い出すことができた．

と回答した．

履歴閲覧後の「どの履歴データを一番見たか」との質問では，

・ ソースコード編集履歴を見て，思い出すのがメインだった．
・ ソースコードを見て，ちょっと思い出せなかったら，ソースコードのタイムスタンプと近い時間の発話の書き起こしデータを見て確認した．それで状況は掴むことができた．

と回答した．5 月のプログラミング観察実験では実験参加者の発話を音声データとして収集し，書き起こしデータを作成した．今回 HALO-Browser では，プログラミングメモの代わりに発話の書き起こしデータを表示した．ソースコードと発話の書き起こしを，HALO-Browser で時間連携によって閲覧していた．ソースコードを中心として，他の履歴と連携させていくことが，過去のプログラミングの状況を理解するのに有効であるように考えられる．

「HALO-Browser の改善点」について質問したところ，

・ 検索履歴はあまり閲覧しなかった．
・ ソースコードと，実行結果を一緒に見たかった．
・ 語句検索で「色」と入力した際に．色に関連する処理をするソースコードがハイライトされたら便利かもしれない

といった回答が得られた．検索履歴をあまり閲覧しなかったのは，検索履歴として表示する情報に不足があり，実験参加者にとって有用ではなかったためだと推測された．検索履歴をただ見るのではなく，Web サイトの URL から実際のページに遷移できる等の仕組みがあればもう少し活用されていたように考えられる．

ソースコードと実行結果を一緒に見たいという要望は，一人称観察実験の際にもみられた．これからの HALO Tools における情報提示の一つとして考えていきたい．

実験参加者からは，HALO-Browser における語句検索機能の改善のアイデアについての発言もあった．単なる文字列であるソースコードに，人が理解できる形で意味付けをおこなうというアイデアである．予めソースコードの処理内容についての辞書を定義しておくことで，日本語等のキーワードからソースコードの検索が可能となると考えている．

6. おわりに

本研究では，まず一人称研究をベースにした事例観察を実施して得た知見や収集した実データを基に，学習者のプログラミングによって生じる学習履歴を蓄積し提示する学習支援環境である HALO Tools の構築に着手し，HALO Tools を用いた事例観察を行った．事例観察の結果から，HALO Tools 上でのプログラミングと履歴の在り方について考察した．

一度目の事例観察では，ソースコードや検索履歴，学習者が作成したテキストメモといった，Jupyter Lab におけるプログラミングに伴う学習の履歴データを収集した．構築した HALO Tools を用いたプログラミング学習を実施した二度目の事例観察では，履歴を用いたプログラミングが自己の経験を引き出す上で役立つことが示唆された．知識も含めて経験を想起出来ていたとまでは言えないが，履歴から情報を検索して過去のプログラミング内容を推測するまでには至っていた．一人称ではない他者についての事例観察からも，HALO-Browser における履歴の時間連携の有用性が示唆された．

今後の展望として，本研究での事例観察によって明らかになった HALO Tools が持つ機能の問題を改善することと，HALO-Browser 上での履歴表示の在り方を模索することが挙げられる．事例観察では，履歴表示の在り方として日付データに着目するのか，プログラミングの中身に着目するのか等の観点があった．履歴表示の在り方は，自己の経験を引き出すことにも直結するため今後の課題である．

今後は，様々な状況や課題設定によるプログラミングの事例観察を HALO Tools を用いて実施していく予定である．

謝辞

HALO Tools の実装にあたっては松原伸人氏にご協力いただいています．感謝します．観察実験の実施にあたって本学大学生 1 名に参加協力いただきました．

参考文献

[1] pandas - Python Data Analysis Library, https://pandas.pydata.org/, 2022/06/20 visited.
[2] scikit-learn: machine learning in Python - scikit-learn 1.1.1 documentation, https://scikit-learn.org/stable/, 2022/06/20 visited.
[3] Andrew J.G. Cockburn, Reflexive CSCW and Managing Commitments, IEE Colloquium on CSCW: Some Fundamental Issues, pp.1-5 , 1991.
[4] Marco Ronchetti, Work in Progress: Real-Time Annotations of Video-Lectures, Teaching and Learning in a Digital World, pp.299-304, 2018.
[5] 知見邦彦, 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, 電子情報通信学会論文誌, J88-D-I 巻 1 号, pp.66-75, 2005.
[6] Project Jupyter - Home, https://jupyter.org/, 2022/06/20 visited.
[7] 藤井晴行, 創造という行為の研究について, 人工知能学会誌, 28 巻 5 号, pp.720-725, 2013.
[8] 堀浩一, 人工知能研究の方法, 人工知能学会誌, 28 巻 5 号, pp.689-694, 2013.
[9] 中小路久美代, 山本恭裕, 松原伸人, 北雄介, 白石晃一, 小林潤平, デザイン過程が蓄積された複数データ群をブラウジングする情報環境, Design シンポジウム 2019, pp.24-30, 2019.
[10] 渡部丈, 中小路久美代, 山本恭裕, 自己のプログラミング履歴を用いたインタラクティブなプログラミング学習支援環境, 情報処理学会第 84 回全国大会, 5ZH-06, pp.4-629-4-630, Matsuyama, Ehime, March, 2022.