日本ソーシャルデータサイエンス学会
2025 年春季研究発表会

エディタログのプログラミング教育支援への応用に関する一検討

鈴木 源吾[†1]，曽根 歩生[†1]，熊倉 一哉[†1]，松崎 優樹[†1]，
吉田 貴裕[†1]，堀川 桂太郎[†1]，飯村 結香子[†2]，斎藤 忍[†2]

キーワード：プログラミング教育，大規模言語モデル，エディタログ，データ分析

1. はじめに

ChatGPT などの大規模言語モデル（Large Language Models，以下，LLM）の登場は，プログラム開発の効率性を大きく向上させている．しかし，大学におけるプログラミング教育では，学生が生成 AI で生成したプログラムを説明できなかったり，生成されたプログラムのエラーを修正できなかったり等の負の影響も目立ってきている．この課題を解決するためには，学生の生成 AI の利用状況を詳細に把握し，的確に助言することが重要と考えた．そこで，エディタログを活用し，学生の詳細な状況の把握や，プログラミング状況を再生し観察することの有効性の検証に着手した結果について報告する．

2. 既存研究

ChatGPT の一般利用は 2022 年 11 月に始まったために，生成 AI のプログラミング教育に対する影響に関する研究例は多くないが，村田他[1]に現状がまとめられている．そこでは，生成 AI を利用したプログラミング教育において，プログラミング概念習得に適すること，反転型授業で有効であること，プログラム完成率や成績向上につながったことなどの，ポジティブな結果が多く示されている．一方，一般的な教育では，批判的思考能力の低下可能性や生成される情報の評価の難しさが指摘されているが[2]，プログラミング教育における負の影響に関する研究は未開拓である．

また，エディタログのプログラミング教育への活用という観点では，プログラミング課題の成績予測[3]や，プログラミングパターン（行動遷移モデル）と成績の相関[4]などの研究があり，比較的短期間に大量のログデータを取得し機械学習を活用することで，有用な結論を出すことに成功しており，生成 AI に関する課題解決が期待できる．

3. 研究のアプローチ

3.1 リモート開発環境 MORDEn とエディタログ活用

本研究では，NTT が開発したリモート開発環境 MORDEn (Micro Organized Remote Development Environment) [5][6]を利用してエディタログを取得した．MORDEn は，開発中のソースコードから「秘匿したいコード」を分離し，秘匿コードを読み書きできない状態で，プログラムを作成／テスト／実行できる開発環境である．図 1 に MORDEn の構成を示す．開発者は MORDEn クライアントがインストールされたエディタを使って公開コードの編集やデバッグを行い，サーバ側でコーディング支援やデバッグ支援などの機能を提供する.クライアントとサーバのやりとりにはオープンスタンダードなプロトコルである，Language Server Protocol（LSP）[7]，Debug Adapter Protocol（DAP）[8]が用いられ，秘匿情報を漏洩するようなアクションはプロキシでブロックされる．

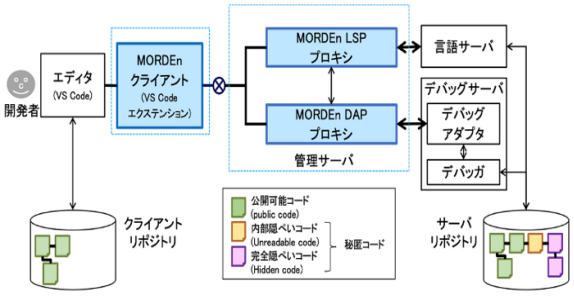

図 1　リモート開発環境 MORDEn の構成

本研究では，機能の確認や評価・デバッグなどの目的で保存された LSP・DAP メッセージのログをデータベース化し利用する．行動履歴の取得方法として，エディタログは動画に比べると解析が容易だが，独自にキーロガーなどを開発する必要があったため取得コストは高かった．現在，ログ取得を MORDEn に完全に依存しているが，LSP・DAP のライブラリは整備され利用しやすく，独自にログ収集を開発することも容易になっている．LSP・DAP ログはキーロガーよりもコピペ文字列・デバッグ結果などを取得しやすいメリットがある．

3.2 エディタのリプレイツールを活用した観察

エディタログのデータベースを用いれば，学生の躓いた箇所としてエラー発生・テスト失敗・長大コピペなどを検索で発見できる．さらに，躓きの詳細を知るためには，その時点でのプログラム内容や，どのような試行錯誤があったかを実際に観察することが有効である．そこで，エディタログデータを用いて，エディタ画面を再生し，特定時点にジャンプできるリプレイツールを試作した．

3.3 学生実験の実施

Python によるプログラニング実習を想定した学生（8 名）を被験者としたエディタログを収集する実験を行った．初心者向けの基本問題からプログラミングコンテストの易し

†1 開志専門職大学
†2 NTT コンピュータ＆データサイエンス研究所

めの問題 10 問を，生成 AI 使用を許した状況で実施し（最大 4.5 時間），約 38 万行のログデータを取得した．

4. 結果と考察

4.1 学生実験から得られたデータの可視化

図 2 は，ある学生のプログラミング課題毎に取り組んだ時間の幅を示している．作業状況を正確に把握できる．

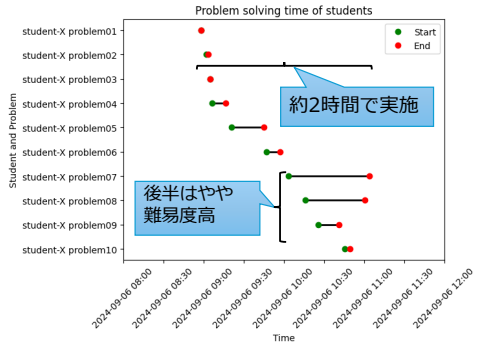

図 2　学生の作業状況

図 3 は，2 人の学生のログにおける修正文字列の長さの時系列変化を示している．通常のエディタ入力ではログに修正文字列は 1 文字ずつ記録される．この図に現れる 600 文字以上の修正文字列は，生成 AI で生成された文字列をコピペしている可能性が高い．最後に長大コピペが多く出現しているのは，期限ぎりぎりで生成 AI を利用して駆け込みで課題に取り組んだことが観察される．

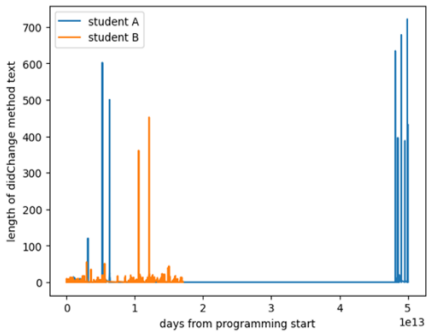

図 3　2 人の学生の修正文字列長の比較

4.2 エディタリプレイツール

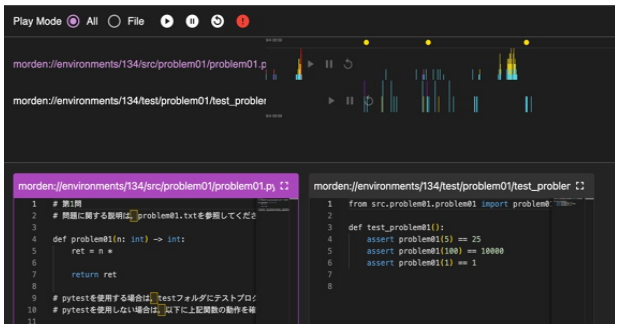

図 4　リプレイツールの画面

図 4 に試作したリプレイツールの画面を示す．上部が再生・ジャンプなどを実施する制御部，中間がファイルごとの編集操作量などを時系列で表示するタイムライン部，下部がファイルの内容を表示するエディタ部である．ログファイルを読み込んで，再生ボタンを押すことでエディタ部に編集操作が再現される．躓き発見の機能として，エラー箇所へのジャンプ機能を実装した（赤の！ボタン）．ボタンを押すと，まず，タイムライン部にエラー発生箇所がファイル毎に黄色の丸で表示される（図 5）．この黄色の点をクリックすると，エラーの発生した時点のエディタの状態（図 6）が再現される．プログラムやエラーに対する理解度や失敗した過程を，教師や研究者が確認でき．学生への助言や状況の把握に活用することが期待できる．

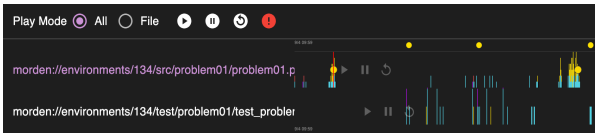

図 5　エラー箇所の表示

図 6　エラー箇所の再生

5. おわりに

現状としては，目標とする生成 AI に関する課題解決にはギャップがあるが，それを埋めていくために．生成 AI 利用・非利用で取得したログの比較分析，プロンプト作成のログ取得と分析，生成 AI 利用・テスト失敗，SQL like な任意のクエリ結果，などの時点に飛べる柔軟なジャンプ機能の実現などに取り組んでいく．

謝辞

本研究は日本電信電話株式会社より支援を受けている．

参考文献

[1] 村田他, “生成 AI によるプログラミング教育のパラダイム転換と教育支援ツールに関する研究構想”, 情報処理学会 情報教育シンポジウム, 2024.
[2] Rahman et al., “ChatGPT for Education and Research: Opportunities, Threats, and Strategies”, Appl.Sci.vol.13, 2023.
[3] Gao et al., “Early Performance Prediction using Interpretable Patterns in Programming Process Data”, ACM SIGCSE’21, 2021.
[4] Carter et al., “Using Programming Process Data to Detect Differences in Students' Patterns of Programming”, ACM SIGCSE’17, 2017.
[5] S. Saito, “Coding and Debugging by Separating Secret Code Toward Secure Remote Development,” 38th IEEE/ACM ASE, 2023.
[6] 吉田他, “ソフトウェア開発における秘匿コードを保護するセキュアな開発環境のユーザビリティ評価”, 第 86 回情報処理学会全国大会講演論文, 2024.
[7] Language Server Protocol (LSP), https://microsoft. github.io/language-server-protocol/ , (参照 2025-02-01) .
[8] Debug Adapter Protocol (DAP), https://microsoft. github.io/debug-adapter-protocol/ , (参照 2025-02-01) .