G-10

生成 AI プログラミングにおけるプロンプト分析ツールの開発

Development of a Prompt Analysis Tool for Generative AI Programming

○品田名留[1]，依田みなみ[2]，松野裕[2]，
Naru Shinada[1], Minami Yoda[2], Yutaka Matsuno[2]

Abstract: The rapid spread of large language models (LLMs) such as ChatGPT has transformed software development practices. A notable trend is vibe coding, where AI generates code and developers concentrate on design and refinement. While this approach promises efficiency, it raises concerns regarding code quality, maintainability, and the decline of programming skills. To investigate these issues, we developed a tool that collects prompts and outputs during AI-assisted development for systematic analysis. This paper introduces the tool and discusses its potential to support research and education on the practical implications of vibe coding.

１．はじめに

近年，ChatGPT や Claude に代表される大規模言語モデル（LLM）の登場により，ソフトウェア開発において生成 AI を活用する事例が急速に増加している．その中でも，コード生成を AI に委ね，人間が設計や修正に注力する開発手法はバイブコーディングと呼ばれ，新しいソフトウェア開発スタイルとして注目を集めている．[1]しかし，バイブコーディングはまだ発展途上の段階にあり，生成 AI が出力するコードの品質や保守性の低下，開発者自身のプログラミング能力低下など，いくつかの課題が指摘されている．[1]

これらの課題を明らかにするためには，生成 AI と人間との対話過程におけるプロンプトの内容や修正過程を体系的に調査・分析することが重要である．プロンプトは生成 AI に与える要求の表現であり，その質が出力結果の精度や妥当性に直結するためである．

本研究では，バイブコーディングにおける開発過程を調査可能とする基盤として，プロンプト収集・分析ツールを開発した．本論文では，提案ツールの設計と実装を述べ，その利用による分析方法を示す．最後に，教育や研究への応用可能性，および今後の課題について考察を行う．

２．原理

２.１．バイブコーディング

バイブコーディングとは，生成 AI にコード生成を委ね，人間が設計や修正に注力する新しい開発手法である．対話的に開発を進められる点が特徴であり，近年注目を集めている．

利点として，コード記述作業の効率化，迅速なプロトタイピング，学習支援といった効果が期待される．一方，生成コード品質や保守性低下，設計原則軽視，開発者のスキル低下などの課題が指摘されている．[1]

２.２．プロンプトエンジニアリング

生成 AI の出力は，入力されるプロンプトの質に大きく依存する．同じ内容を依頼する場合でも，表現の仕方や具体性の有無によって結果が大きく変化する．そのため，適切な指示の構造化や文脈の明示といった工夫が必要になっている．

バイブコーディングにおいては，開発者が生成 AI に与えるプロンプトが，コードの品質や開発効率を左右する．したがって，プロンプト設計技術であるプロンプトエンジニアリングは，バイブコーディングを有効に活用するための基盤的なスキルと位置づけられる．

３．提案ツールの概要

３.１．概要

本研究では，LLM Study Helper（LSH）を開発した．生成 AI 利用時に入力されたプロンプトと生成結果を収集し，分析可能な形で保存することを目的としている．本ツールを用いることで，AI に対する指示内容や履歴を捉え，開発プロセスの試行錯誤を可視化することが可能となる．

収集されたデータは，単なるログとして保存されるだけでなく，分析指標に基づいて整理される．これにより，プロンプトの長さや具体性，修正回数などを定量的に把握でき，生成 AI を活用した開発における特徴を明らかにできる．

３.２．設計

LSH では，プロンプトの収集から保存，さらに分析までを一貫して実現することを目的として設計した．全体の流れは Fig．１に示す．本設計はユーザの操作を最小限に抑えるため，GoogleChrome 拡張機能を利用し，データの保存，送信を一つの機能に統一している．

１：日大理工・学部・応用情報　２：日大理工・教員・応用情報

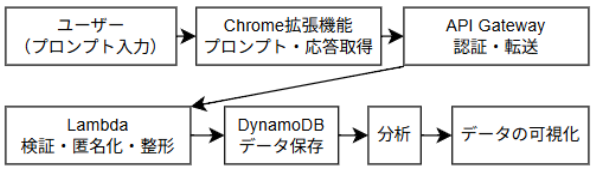

Figure 1.System flowchart

４．提案ツールの開発

LSH は，フロントエンドとして，Chrome 拡張機能と React を利用した UI システムと，バックエンドとしての AWS サーバレス基盤から構成した（Fig．2）．

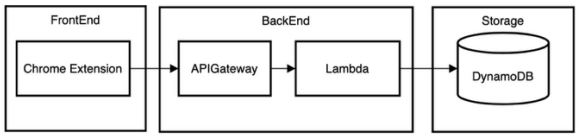

Figure 2.System configuration

・Chrome 拡張機能

生成 AI の入力・応答を監視し，プロンプトと生成結果を収集する．収集データは JSON 形式に整形され，バックエンドに送信される．データの収集にはユーザ自身の許可が必要となるため，自動的にデータの収集を行わないように設計されている．

・API Gateway＋Lambda

API Gateway は拡張機能からのリクエストを受け付け，認証・アクセス制御を行い，Lambda 関数はデータの検証・匿名化処理を行う，これらにより，データの送信，受信が可能となる．

・DynamoDB

ユーザ情報，プロンプト・応答の内容，タイムスタンプといったメタデータを格納するデータベースである．

５．分析方法

LSH では，プロンプト本文，生成 AI の生成結果，メタ情報のデータを DynamoDB 上に収集する．

収集したデータは分析する際に，Table．1 のようにデータを分析する．具体性スコアとは，Table．2 のように，プロンプトのデータを分析し，それぞれの要素があるかないかを目視で確認し分析する．それぞれの合計点数を具体性スコアとする．

Table 1. Analysis indicators and summary

分析指標	概要
プロンプト長	プロンプトの文字数
プロンプト数	同一会話におけるプロンプト回数
エラー文提出回数	同一会話におけるプログラムエラー回数
具体性スコア	プロンプト内の具体的要素の有無の点数化

Table 2. Score metrics and summary

スコア指標	概要
目的の明示	成果物や期待出力を明確に述べているか
入出力指標	型・範囲・例を含めて記述しているか
制約条件	行数、API回数などの制約条件を示しているか
実行環境・依存関係	言語やフレームワーク、OSなどの情報を示しているか
メッセージ引用	実際に発生したメッセージを示しているか
受け入れ基準	合格の条件を示しているか
参照対象の特定	ファイルなどの参照対象を示しているか
非目標の明示	やらないことや範囲外を示しているか
手順の明示	手順や順序を明記しているか

６．動作検証

実際に作成したアプリの実行図を示す．Fig．3 には実際の生成 AI サービスを利用する際の画面，Fig．4 にデータ送信用画面を示す．

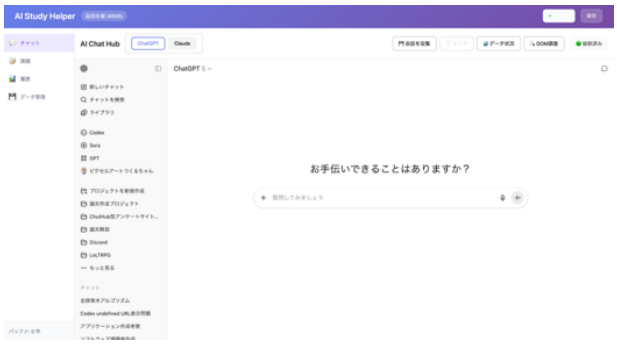

Figure 3. Generative AI Service Interface

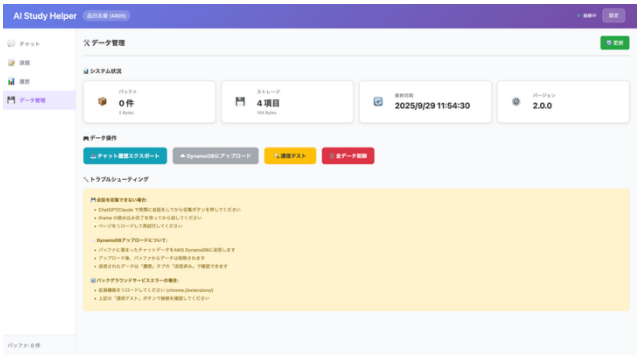

Figure 4. Data Transmission Interface

７．今後の展望

本研究では，生成 AI を活用したソフトウェア開発において，プロンプト収集と分析を可能にするツールを開発した．今後は，教育的応用と機能拡張を進める．

教育面では，プロンプト構築過程を可視化することで，生成 AI を活用した教育効果の検証，プロンプト設計力向上への寄与を目指す．技術面では，プロンプトの自動分析機能や可視化機能を拡張し，研究と教育の双方に役立つ実用的な基盤へ発展させることを目指す．

８．参考文献

[1] Christian Meske et al.:“VibeCoding as a Reconfiguration of Intent Mediation in Software Development: Definition, Implications, and Research Agenda”, arXiv preprint, 2507.21928, 2025.