プロンプトは自然言語プログラミングになりうるか？

伊東 和香 [1] 森部 七海 [2] 中村 優希 [2] 西潟 優羽 [1] 倉光 君郎 [2]
[1] 日本女子大学大学院 理学研究科 [2] 日本女子大学 理学部
m2016013iw@ug.jwu.ac.jp kuramitsuk@fc.jwu.ac.jp

概要

大規模言語モデル (LLM) の発展により，自然言語で書いたプロンプトからコードを生成できるようになった．プロンプトは新しい自然言語プログラミングではないかという問いも生じた．本研究では，この問いに対し，プロンプト品質とプログラム品質の関係を定量的に調査した．プロンプトの評価として，ソフトウェア開発の知見に基づき 5 カテゴリ 15 項目の評価基準を策定し，画像分類モデルの作成，名前性別判定，オセロ AI の 3 種類の課題で実験を行った．学生から収集した合計 212 件のプロンプトを分析した結果，プロンプト評価とプログラム性能の間に有意な相関は見られなかった．この結果は，LLM の非決定性や，従来のプログラミング言語との本質的な違いを示している．本研究は，プロンプト技能の訓練だけでは不十分であることを示唆している．

1 はじめに

大規模言語モデル (LLM) が，プログラミングやソフトウェア開発の世界を大きく変えつつある [1, 2]．LLM は自然言語で書いたプロンプトから，指示に沿ったコードを生成できることはよく知られている．

生成されるコードの品質も年々向上している．当初は大学初級レベルの能力で，明らかな間違い (ハルシネーション) も多かった [3]．しかし今では，中級エンジニアに匹敵する能力を持つようになっている．生成されたコードを人間が理解して修正する必要すら，なくなりつつある．

ここで新しい疑問が浮かんでくる．プロンプトに書く自然言語こそが，人間がソフトウェア開発で与える指示になっているのではないか．つまり，プロンプトは新しいプログラミング言語と言えるのではないか，という点である．

プログラミングの歴史 [4] を振り返ると，マシン語から始まり，より抽象度の高い言語へと進化してきた．最初のプログラミング言語は，数式と簡単な英単語で指示できるよう設計された．信頼できるコンパイラの登場により，マシン語を理解する必要はなくなった．その後，オブジェクト指向などのより抽象的な概念が導入されてきた．昔から，「最終的には自然言語でプログラミングできるようになる」という予測は多く存在していた [5]．

本研究の目的は，「プロンプトは自然言語プログラミングになりうるか？」という問いを，プログラミング教育 [6] の視点から検討することである．

プログラミング教育の視点で考えると，もしプロンプトが自然言語プログラミングになるなら，何を教えるべきか，そして教えた内容が成果につながるのかが重要になる．実際，著者らがプログラミング演習で生成 AI を導入した際 [7]，興味深い現象を観察した．例えば，手順を詳しく書いたプロンプトよりも，簡潔なプロンプトの方が良いプログラムになる場合があったのである．これは，プロンプトの書き方とプログラム品質の関係が単純ではないことを示している．

そこで本研究では，プロンプトの書き方 (プロンプト品質) と生成されるプログラムの品質 (プログラム品質) の関係を調査し，プロンプトが自然言語プログラミングになりうるかを明らかにする．

2 プロンプトとプログラミング

プロンプトが自然言語プログラミングだとすれば，何を書くべきなのか？ あるいは，何を書くように教えるべきなのか？

この問いに答えるため，我々はソフトウェア開発の経験をもとに，コード生成プロンプトに記述すべき内容を分析した．そして 4 カテゴリ，12 の評価項目に整理した．さらに，プロンプト評価の一般的な評価項目を加え，最終的に 5 カテゴリ，15 項目を設定した．

以下にプロンプト品質の評価項目を示す．

1. **基本構造**
   - (**ゴール**)「ゴールが具体的かつ明確か，つまり常にコードが同じ動作になるか」
   - (**入力**)「プログラムへの入力仕様が明確か，エラーを避けられるか」
   - (**出力**)「プログラムからの出力仕様が明確か，常に同じ出力形式になるか」
2. **品質向上**
   - (**テスト**)「テストケースや実行例が十分か，コーナーケースの抜けはないか」
   - (**例外**)「例外時の処理内容が具体的か」
   - (**前提**)「実行前などに満たすべき前提が条件化されているか」
3. **実装処理**
   - (**手順**)「実装すべき手順について詳細があるか」
   - (**アルゴリズム**)「適切なアルゴリズムが指定されているか」
   - (**データ構造**)「データ構造に関する指示が適切か」
4. **エンジニア**
   - (**NFR**)「与えられるべき非機能要求 (パフォーマンス，セキュリティなど) が明確に指示されているか」
   - (**スタック**)「使用すべき技術スタック (ライブラリやバージョン) が明確か」
   - (**コード**)「コードの品質 (規約，コメント，名前慣習) が明確か」
5. **プロンプト一般**
   - (**プロンプト**)「プロンプトエンジニアリングの観点で効果的か」
   - (**矛盾**)「記載内容が相互に矛盾していないか」
   - (**トークン効率**)「トークン効率の観点から冗長過ぎる指示がないか」

以上の項目に基づき，LLM-as-a-judge 手法を用いてプロンプトを評価する．

3 実験

3.1 実験デザイン

プロンプトの多様性を確保するため，著者らの担当する講義で学生に協力を依頼し，プロンプトを収集した．収集方法は，(1)LMS 経由での提出，(2) 専用ツール Hutch[8] による自動収集の 2 種類である．

プロンプト評価には，Google Gemini 3.0 を用いた LLM-as-a-judge を採用した．各評価項目について 10 点満点で採点した．LLM-as-a-judge による評価は，人間の採点に比べて安定した評価が得られた．

3.2 プログラミング課題

オープンエンドな 3 種類の課題を用意した：

(1) 画像分類モデル：実写真と AI 生成画像の二値分類モデルを作成する．学生は，Gemini 2.5 を使用してプログラムを生成した．最終的に，17 件のプロンプトを収集した．プログラム品質の評価はテストデータの正解率を用いた．

(2) 名前性別判定：日本人の名前から性別を判定する関数を定義する．102 件のプロンプトを収集し，Claude Haiku 4.5 でプログラムを再生成し，評価を行った．評価は 3 万件のテストケースの正解率を用いた．

(3) オセロ AI：提供された UI と統合可能な強いオセロ AI を作成する．学生は，Claude Haiku 4.5 を使用してプログラムを生成した．最終的に 93 件のプロンプトを収集した．プログラム品質の評価には，戦略の異なる 3 種類のオセロ AI との対戦成績 (総獲得枚数) とした．

3.3 実験結果

図 1,2,3 に，各課題におけるそれぞれのカテゴリごとのプロンプト品質とプログラム品質の散布図，全てのカテゴリの平均 (合計) とプログラム品質の散布図を示す．それぞれ縦軸がプロンプト品質の結果，横軸がプログラム品質の結果となっている．相関は，ピアソン相関係数を用いている．

結果より，いずれの課題においても，両者の間に有意な相関は見られなかった．

名前性別判定の結果 (図 2) では，プロンプト品質評価結果に大きなばらつきが見られた．例えば，基本，品質向上，詳細実装のカテゴリにおける結果では，プログラム品質 (正解率) がほぼ同等であっても，プロンプトの評価結果が 7〜8 点の差がつくケースが見られた．

また，オセロ AI の結果 (図 3) では，獲得枚数がほぼ 0 のプログラムにおいて，プロンプト評価が 4〜6 点の差がついているケースも見られる．

画像分類モデル (図 1) は，プロンプトの件数が他の課題よりも限られていたものの，正解率が大きく

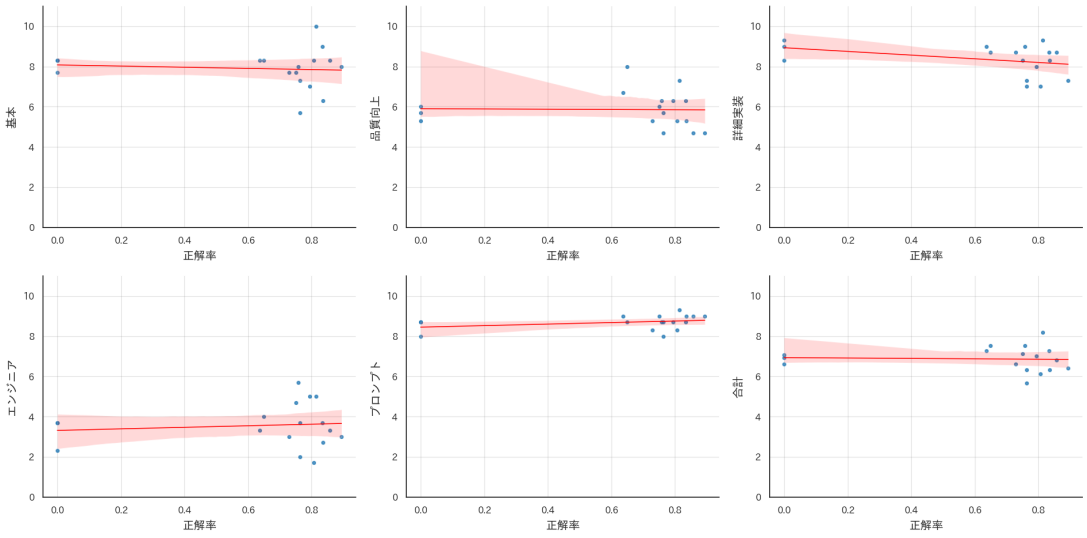

図 1　画像分類モデルにおけるプロンプト品質とプログラム品質 (17 件)

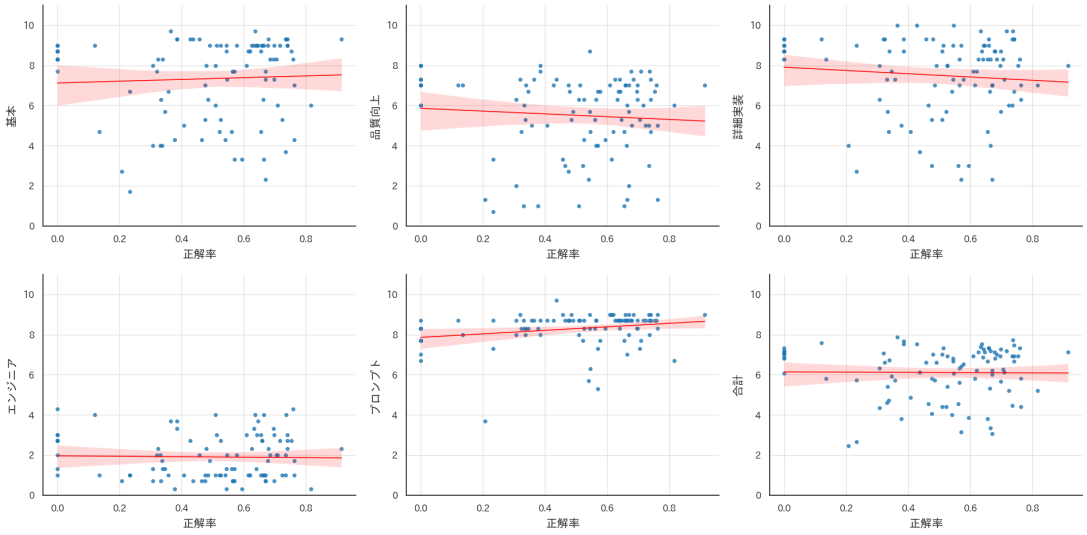

図 2　名前性別判定におけるプロンプト品質とプログラム品質の関係 (102 件)

離れている場合でも，プロンプト評価の結果には大きな差が見られなかった．

4　考察

なぜ相関がないのか — プロンプト品質とプログラム品質に相関が見られなかった主な要因として，以下が考えられる：

(1) LLM の非決定性：同じプロンプトやわずかな違いでも実行のたびに異なるコードが生成される．非決定性の影響が大きい．

(2) 評価基準のミスマッチ：我々が策定した評価基準は，ソフトウェア工学の観点から「良いプロンプト」を定義したものだが，LLM が実際に重視する要素とは異なる可能性がある．

プログラミング言語との比較 — 従来のプログラミング言語は，決定的 (deterministic) である．同じソースコードは，同じ環境で常に同じ動作をする．これに対し，プロンプトによるコード生成は，本質的に非決定的である．

また，プログラミング言語には明確な構文規則があり，エラーは即座に検出される．プロンプトには形式的な正しさの基準がなく，何を書いたらよいのか曖昧である．

さらに，プログラミング言語では，小さな変更が予測可能な影響をもたらす．プロンプトでは，わずかな表現の違いが予測不可能な結果を招くことがある．

生成 AI 教育への示唆 — 本研究の結果は，「プロ

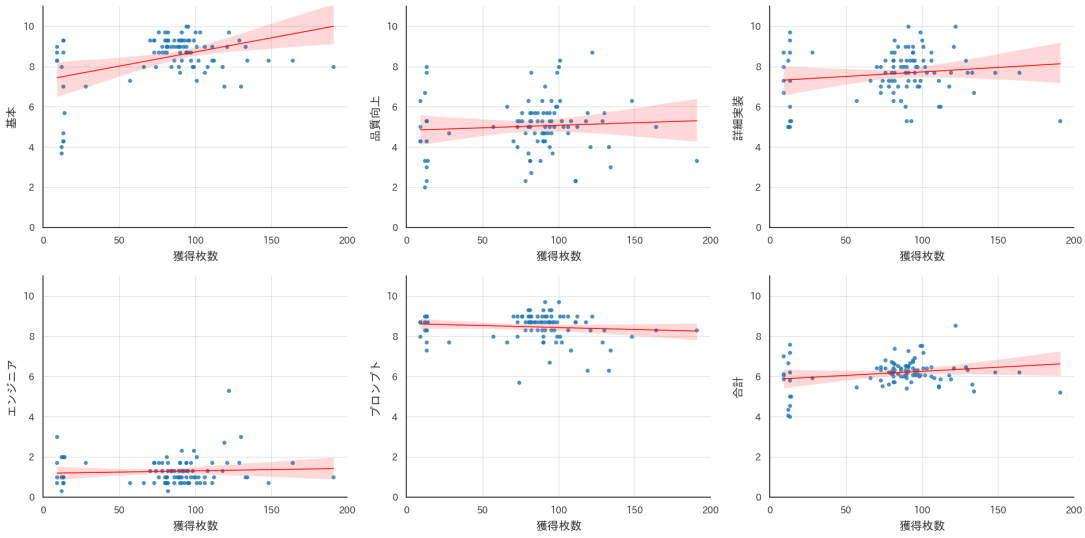

図 3　オセロ AI におけるプロンプト品質とプログラム品質の関係 (93 件)

ンプト技能」を重視する教育アプローチ [9] に疑問を投げかける．単に，プロンプトの書き方を訓練しても，それがプログラムなどの生成物の品質向上につながらない可能性が示された．

ただし，著者らはプログラミング教育に対する生成 AI の導入を否定するものではない．生成物の品質向上につながる再現性のあるメソッドを開発することが AI 教育には必要と考える．

5　関連研究

プロンプトエンジニアリングは，LLM から望ましい出力を得るための技法である．Few-shot プロンプティング，Chain of Thought，In-context Learning など，様々な手法が提案されている [10]．プログラミング教育においても，エラー解説 [11] やコード訂正 [12] への活用が報告されている．しかし，これらの技法は主に生成 AI の特性に基づくものであり，ソフトウェア開発の本質的な知識とは異なる．また，LLM の性能向上により，高度なプロンプティング技法を使わなくても十分な出力が得られるケースが増えている．

プログラミング言語は，機械語から高水準言語へと抽象度を上げながら発展してきた [4]．LLM の登場により，自然言語による記述が実現し，「プロンプトはプログラムである」[13] とする主張もある．

Prompt Problems[14] や Promplty[15] など，プロンプトを教育の中心に据えた教材も登場している．しかし，プロンプトの書き方が実際のプログラム品質にどう影響するかを定量的に評価した研究は，我々の知る限り存在しない．

6　むすびに

本研究では，プロンプト品質とプログラム品質の関係を定量的に調査した．3 種類のプログラミング課題を用いて分析した結果，プロンプト評価と実際のプログラム性能の間に有意な相関は見られなかった．この結果は，LLM の非決定性が想定よりも大きな影響を与えるなど，複数の要因によるものと考えられる．

この調査結果から，プロンプトを「自然言語プログラミング」として扱うには課題があることが分かった．決定性，良いプロンプトの定義，予測可能性などの点で，従来のプログラミング言語とは大きく異なっている．しかし，これは生成 AI の導入を否定する理由にはならない．むしろ，生成 AI という新しいツールに適したプログラミング教育が必要であることを示している．

今後の課題として，どのようなプロンプトの特徴が高品質なコード生成につながるかを詳しく分析することが挙げられる．また，対話的なプロンプト (vibe coding) における品質評価も重要である．

謝辞

本研究は科研費 23K11374 の支援を受けた．実験は日本女子大学倫理審査委員会の承認のもと実施された．

参考文献

[1] Albert Ziegler, Eirini Kalliamvakou, X. Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. Measuring github copilot's impact on productivity. **Commun. ACM**, Vol. 67, No. 3, p. 54–63, February 2024.

[2] Daniel Russo. Navigating the complexity of generative ai adoption in software engineering. **ACM Trans. Softw. Eng. Methodol.**, Vol. 33, No. 5, June 2024.

[3] James Finnie-Ansley, Paul Denny, Brett A. Becker, Andrew Luxton-Reilly, and James Prather. The robots are coming: Exploring the implications of openai codex on introductory programming. In **Proceedings of the 24th Australasian Computing Education Conference**, ACE '22, p. 10–19, New York, NY, USA, 2022. Association for Computing Machinery.

[4] Brent N. Reeves, James Prather, Paul Denny, Juho Leinonen, Stephen Macneil, Brett A. Becker, and Andrew Luxton-Reilly. Prompts first, finally. **ArXiv**, Vol. abs/2407.09231, , 2024.

[5] M. Halpern. Foundations of the case for natural-language programming. **IEEE Spectrum**, 1967.

[6] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. Computing education in the era of generative ai. **Commun. ACM**, Vol. 67, No. 2, p. 56–67, January 2024.

[7] Kimio Kuramitsu, Waka Itoh, Kaoru Kagami, and Ayako Masamune. Prompts first で始めたプログラミング教育の再構築. 第 2025 巻, pp. 64–71. 情報教育シンポジウム論文集, Aug 2025.

[8] 西潟優羽, 伊東和香, 中村優希, 倉光君郎. Hutch: プログラミング教育向けプロンプト品質の可視化ツールの開発. 情報処理学会第 67 回プログラミング・シンポジウム, January 2026.

[9] Nils Knoth, Antonia Tolzin, Andreas Janson, and Jan Marco Leimeister. Ai literacy and its implications for prompt engineering strategies. **Computers and Education: Artificial Intelligence**, Vol. 6, p. 100225, 2024.

[10] Daniel Lee and Edward Palmer. Prompt engineering in higher education: a systematic review to help inform curricula. **International Journal of Educational Technology in Higher Education**, Vol. 22, , 2025.

[11] Kimio Kuramitsu, Yui Obara, Miyu Sato, and Momoka Obara. Kogi: A seamless integration of chatgpt into jupyter environments for programming education. **Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E**, 2023.

[12] Bruno Pereira Cipriano and Pedro Alves. "chatgpt is here to help, not to replace anybody" - an evaluation of students' opinions on integrating chatgpt in cs courses. In **International Conference on Computer Supported Education**, 2024.

[13] Jenny T. Liang, Melissa Lin, Nikitha Rao, and Brad A. Myers. Prompts are programs too! understanding how developers build software containing prompts. **ArXiv**, Vol. abs/2409.12447, , 2024.

[14] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. Prompt problems: A new programming exercise for the generative ai era. In **Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1**, SIGCSE 2024, p. 296–302, New York, NY, USA, 2024. Association for Computing Machinery.

[15] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. Promptly: Using prompt problems to teach learners how to effectively utilize ai code generators. **ArXiv**, Vol. abs/2307.16364, , 2023.