

卒業研究報告書

題目 オープンソースソフトウェア開発における
ChatGPT 提案コードの再利用動向の分析

指導教員 水野 修 教授

崔 恩瀨 助教

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号 20122055

氏名 宮永 翔多

令和6年2月13日提出

オープンソースソフトウェア開発における ChatGPT 提案コードの再利用動向の分析

令和 6 年 2 月 13 日

20122055 宮永 翔多

概 要

ChatGPT のような大規模な言語モデルの登場は、ソフトウェア開発の状況を一変させた。ChatGPT によって生成される応答の品質やコードの性能比較を調査する研究など数多くの研究が行われてきた。しかし、ChatGPT が実際にソフトウェア開発者にどのように使用されているかについては、ほとんどわかっていない。

本研究では、ソフトウェア開発者による ChatGPT を利用したソースコードの再利用タスクに着目し、ChatGPT の提案したソースコードの再利用動向について調査する。具体的には、まず、ChatGPT が使用されたオープンソースプロジェクトにおいて、開発者が ChatGPT から再利用したソースコードに着目し、それがどのように再利用されているかを調査した。次に、開発者が ChatGPT の提案したソースコードを再利用するまでの開発者と ChatGPT のやり取りに着目し、開発者は ChatGPT からソースコードを再利用するまでにどの程度の回数会話しているかを調査した。調査には、ソフトウェア開発者と ChatGPT のやり取りを包括的に分析するために作成されたデータセットを用いる。このデータセットは、GitHub 上のオープンソースプロジェクトにおいて、ChatGPT が使用されたコミットやプルリクエストなどの中間生成物の情報を持っている。そのため、ChatGPT が使用されたコミットにおけるソースコードの再利用動向を調査し、その特徴を明らかにした。

調査の結果、開発者は ChatGPT が提案したソースコードを再利用する際に、修正せずにそのまま再利用することが、再利用されたソースコードのうち 83.5% と多いことがわかった。また、開発者は ChatGPT が提案したソースコードを再利用するまでに、平均で 3.65 回 ChatGPT と会話していることがわかった。さらに、修正せずにそのまま再利用する場合、平均で 3.49 回、修正して再利用する場合、平均で 4.77 回 ChatGPT と会話していることがわかった。

目次

1. 緒言	1
2. 背景	3
2.1 ChatGPT	3
2.2 オープンソースソフトウェア開発	3
2.3 ソースコードの再利用	4
3. 調査概要	6
3.1 調査目的	6
3.2 調査対象のデータセットの説明	6
4. RQ1: ChatGPT が提案したソースコードは開発者によってどのように再利用されているか?	7
4.1 動機	7
4.2 方法	7
4.3 結果	11
5. RQ2: 開発者は ChatGPT が提案したソースコードを再利用するまでにどの程度の長さ ChatGPT と会話しているか?	17
5.1 動機	17
5.2 方法	17
5.3 結果	19
6. 議論	24
6.1 RQ1 ChatGPT が提案したソースコードは開発者によってどのように再利用されているか?	24
6.2 RQ2 開発者は ChatGPT が提案したソースコードを再利用するまでにどの程度の回数 ChatGPT と会話しているか?	25
7. 結言	27
謝辞	27

1. 緒言

ChatGPTのような大規模な言語モデルの登場は、ソフトウェア開発の状況を一変させた。ChatGPTは、人間のような会話をする能力、ユーザのフィードバックから継続的に学習する能力を持ち合わせており、さらに、アクセスのしやすさもあり、絶大な人気を誇る。ChatGPTは、様々な分野で利用されており、ソフトウェア開発の分野も例外ではない。GitHub上に公開されたオープンソースプロジェクトのコミットやプルリクエストには、ChatGPTを利用した形跡を見られることがある。また、ChatGPTに関する研究も数多く行われており、ChatGPTによって生成される応答の品質やコードの性能比較を調査する研究などが行われている。しかし、ChatGPTが実際にソフトウェア開発者にどのように使用されているかについては、ほとんどわかっていない。

本研究では、ソフトウェア開発者によるChatGPTを利用したソースコードの再利用タスクに着目し、ChatGPTの提案したソースコードの再利用動向について調査する。具体的には、まず、ChatGPTが使用されたオープンソースプロジェクトにおいて、開発者がChatGPTから再利用したソースコードに着目し、それがどのように再利用されているかを調査した。次に、開発者がChatGPTの提案したソースコードを再利用するまでの開発者とChatGPTのやり取りに着目し、開発者はChatGPTからソースコードを再利用するまでにどの程度の回数会話しているかを調査した。調査には、ソフトウェア開発者とChatGPTのやり取りを包括的に分析するために作成されたデータセットを用いる。このデータセットは、GitHub上のオープンソースプロジェクトにおいて、ChatGPTが使用されたコミットやプルリクエストなどの中間生成物の情報を持っている。そのため、ChatGPTが使用されたコミットにおけるソースコードの再利用動向を調査し、その特徴を明らかにした。

調査の結果、開発者はChatGPTが提案したソースコードを再利用する際に、修正せずにそのまま再利用することが、再利用されたソースコードのうち83.5%と多いことがわかった。また、開発者はChatGPTが提案したソースコードを再利用するまでに、平均で3.65回ChatGPTと会話していることがわかった。さらに、修正せずにそのまま再利用する場合、平均で3.49回、修正して再利用する場合、平均で4.77回ChatGPTと会話していることがわかった。

最後に本論文の章構成を示す。まず、第2章では、本研究で必要となる背景知識について述べる。第3章では、本研究の調査目的について、調査対象となるデータセットの概要について述べる。第4章では、1つ目の研究設問における調査目的、調査方法や調査結果について述べる。第5章では、2つ目の研究設問における調査目的、調査方法や調査結果について述べる。第6章では、調査結果について議論する。最後に第7章で結言を述べる。

2. 背景

2.1 ChatGPT

ChatGPT は GPT (Generative Pre-trained Transformer) 言語モデルを搭載した AI チャットボットで、2022 年 11 月に OpenAI によってリリースされた。ChatGPT はプロンプトの指示に従い詳細な返答をするように訓練されており [1], ユーザとの対話を通じて、自然言語での質問応答を行うことができる。プロンプトとは、ChatGPT に対して提供される命令のことでカスタマイズしたり、改良したりすることで、ChatGPT の応答を操作できる [2]。また、ChatGPT はインターネット上の膨大なテキストデータで学習されているため、幅広いタスクをこなすことができる。例えば、エッセイを書いたり、楽曲の作詞をしたり、コードを書いたり、さらにデバッグしたりできる [3]。しかし、時にもっともらしく聞こえるが正しくない、または無意味な応答を提供する [1]。そのため、最近の研究では、事実の矛盾を検証する手段を持たないエンドユーザにリスクをもたらすことが懸念されている [4-7]。ユーザを簡単に惑わせる誤った情報が蔓延していることから、Stack Overflow は ChatGPT で生成された回答の投稿を禁止している [8]。ChatGPT については、生成される応答の品質、様々な効果的なプロンプト技術、プログラミングコンテストでの比較性能など、多くの研究が行われている [2,9-11]。しかし、ChatGPT が実際にソフトウェア開発者にどのように使われているかについては、ほとんどわかっていない。

2.2 オープンソースソフトウェア開発

ソフトウェア開発では、Git を用いてソースコードのバージョン管理をすることが一般的である。Git はソースコードの変更履歴を記録し、複数の開発者が同時にソースコードを編集できる。ソースコードの変更を記録するためには、コミットと呼ばれる操作をする。コミットには、変更内容を説明するメッセージを記述できる。また、コミットの履歴を参照することで、ソースコードの変更履歴を確認できる。GitHub は、Git を用いたソースコードのホスティングサービスであり、多くのオープンソースソフトウェアプロジェクトが GitHub 上で開発されている。

本論文では、GitHub 上でのオープンソースソフトウェア開発におけるコミット

メッセージにリンクされた ChatGPT の共有リンクをもとに調査する。

2.3 ソースコードの再利用

ソースコードの再利用は一般的に観測されるタスクであり、開発者はソースコードを再利用することで、開発時間を節約し、コード品質を向上させる目的で行う [12]. しかし、ソースコードを再利用することは容易でない [13]. 再利用タスクは、ソースコードの探索と統合の2つのステップに分けられるが [12], ソースコードの再利用は、(1) 開発者が必要とするソースコードを見つけること、(2) 見つけたソースコードを手元のシステムに統合する、という2点で困難である [12,14]. まず1点目は、開発者は開発しているシステムの背景を意識して必要とするものをキーワードや制約という形で正確に指定することが難しい [14]. この問題に対して、開発者にとってより関連性のあるソースコードを見つけられるようにするコード検索技術など、多くの研究が行われてきた [14]. 2点目は、ニーズを満たすソースコードを見つけられたとしても、そのコードを手元のシステムに統合するには、膨大なコード片に対して変数の参照の変換など些細な変更に注力することが必要となるという点で、この作業は面倒でミスを引き起こしやすく、開発者にとって手作業でコードを統合することは難しい可能性がある [12]. この問題に対処するため、変数名の自動変換など様々な手法が提案されてきた [12].

本論文では、再利用元のシステムと手元のシステム間の類似ソースコード片を検索することで、開発者のソースコード再利用の動向を調査する. 類似ソースコード片とは、ソースコード中に存在する互いに一致、または類似したものを指す. 類似ソースコード片を検索するツールの1つとして、NCDSearchがある [15]. NCDSearchは、ソースコードの類似度を計測するためのツールであり、Normalized Compression Distance (NCD) を含む、いくつかの計測方法が用意されている. 本論文の調査では、NCDSearchの類似性メトリクスとして Lempel-Ziv Jaccard Distance (LZJD) を用いた [16]. NCDは、2つのソースコード片がデータ圧縮アルゴリズムにより高度に圧縮されている場合、それらを類似しているものとみなす. LZJDは、NCDの代替として導入されたメトリクスで、NCDと同じく圧縮から着想を得ているが、実際のデータ圧縮結果ではなく、データ圧縮のための辞書を比較する [16].

また、開発者のソースコード再利用方法について、Stack Overflow からのソースコード再利用を調査した研究では、その利用方法は、「Exact Copy」、「Cosmetic Modification」、「Non-cosmetic Modification」、「Converting Ideas」、「Providing Information」の5つのタイプに分類される [17].

3. 調査概要

3.1 調査目的

本研究の目的は、ソフトウェア開発における ChatGPT の利用動向を明らかにすることである。本研究では、開発者が ChatGPT の提案コードを再利用するタスクに焦点を当てたアプローチを取る。そこで、以下の研究設問を設定し、開発者によって再利用された ChatGPT 提案コードの利用動向について調査した。この調査により、ソフトウェア開発において ChatGPT を活用したコード再利用のアプローチを容易にする方法について洞察が得られると期待する。

RQ1 ChatGPT が提案したソースコードは開発者によってどのように再利用されているか？

RQ2 開発者は ChatGPT が提案したソースコードを再利用するまでにどの程度の長さ ChatGPT と会話しているか？

3.2 調査対象のデータセットの説明

調査対象としてコードスニペットを含む ChatGPT の応答とプロンプトのデータセットである、DevGPT を選択した [18]。このデータセットは、ソフトウェア開発の中間生成物とそれに対応する開発者と ChatGPT の会話ログを含む。中間生成物は、ソースコード、コミット、イシュー、プルリクエストからディスカッションや Hacker News のスレッドである。このデータを収集することで、開発者と ChatGPT のやり取りの包括的な分析を可能にすることを目的に作成された。

このデータセットがどのように作成されたのかを説明する。2023 年 5 月下旬に OpenAI が導入した、ユーザが専用のリンクを通じて ChatGPT とのやり取りを共有できる機能を活用して、2023 年 7 月 27 日から毎週 GitHub と Hacker News での ChatGPT 共有リンクのメンションを追跡した。2023 年 8 月 31 日に追跡したデータには、2237 個の GitHub または Hacker News の参照から 2891 の ChatGPT 共有リンクが含まれる。これらの情報がデータセットとして、提供されている [18]。

4. RQ1： ChatGPT が提案したソースコードは開発者によってどのように再利用されているか？

4.1 動機

ソフトウェア開発における ChatGPT の利用動向として、開発者によって再利用された ChatGPT の提案ソースコードの利用方法に着目する。ソフトウェア開発において再利用タスクは定期的に行われる。開発者によって ChatGPT の提案ソースコードがどのように再利用されたかは、ソフトウェア開発における ChatGPT の利用動向の 1 つの指標となると考えられる。

4.2 方法

RQ1 を答えるために、ChatGPT によって提案されたソースコードとコミットにおいて修正されたソースコードを比較した。調査方法は、以下の手順で進めた。この手順を図 4.1 に示す。

- 手順 1 データセットから、コミットとそれに対応する開発者と ChatGPT の会話ログを取得する。
- 手順 2 開発者と ChatGPT の会話ログから、ChatGPT 提案ソースコードを JavaScript, Python, Java で記述されたコードに絞る。
- 手順 3 NCDSearch を使用して ChatGPT 提案ソースコードに対して、対応するコミットから類似ソースコード片を検索する。
- 手順 4 検索されたソースコード片から、再利用したソースコード片を手作業で識別し、そのコードの利用方法を分類する。

手順 1 では、DevGPT から、コミットとそれに対応する開発者と ChatGPT の会話ログを取得する。DevGPT からは、GitHub リポジトリへの URL、コミットハッシュを取得でき、この情報を用いて、GitHub から該当するソースファイルを取得できる。この時、該当するコミットのソースファイルが取得できないデータについては調査対象から除外した。

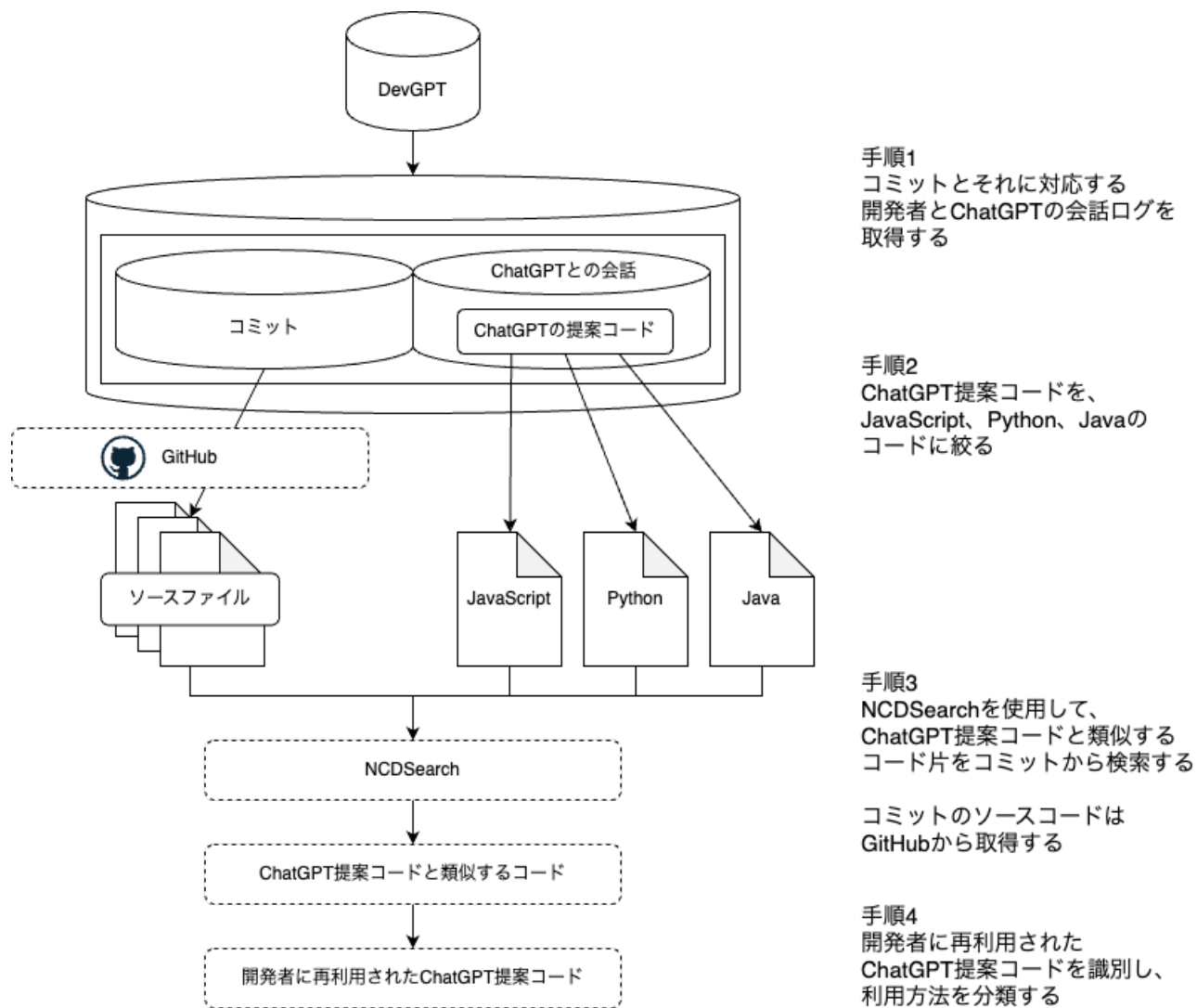


図 4.1 RQ1 における調査方法の概要図

手順2では、コミットに対応する開発者と ChatGPT の会話ログの中で、ChatGPT が提案したソースコードを JavaScript, Python, Java の3つのプログラミング言語で記述されたコードに絞る。全てのプログラミング言語を対象にデータを分析するには時間がかかってしまうため、プログラミング言語を限定したデータに対して分析することにした。ここで、JavaScript, Python, Java を選択した理由は、コミットに対応する開発者と ChatGPT の会話ログ内の、ChatGPT 提案ソースコードの中で、シェルスクリプト言語 (sh, bash) を除いた上位3つのプログラミング言語だからである。データセットにおいて、JavaScript, Python, Java のソースコードはそれぞれ、13.77%, 4.78%, 4.45%の割合で含まれている。

手順3では、NCDSearch を用いて ChatGPT が提案したソースコードに対して、対応するコミットのソースコードから類似ソースコード片を検索する。NCDSearch が類似ソースコード片の検索に使用する計測方法には LZJD を利用する。またこの時、検索された類似ソースコード片から、対応するコミットで修正されたコード片と関連性のないコード片は検索結果から除外した。

手順4では、検索された類似ソースコード片から、開発者が ChatGPT の提案ソースコードから再利用したコード片を手作業で識別する。そして、開発者が ChatGPT の提案ソースコードを再利用したコード片について、その利用方法を分類する。ここでは、検索された類似ソースコード片を全て開発者が ChatGPT の提案ソースコードから再利用したコード片であると決定しなかった。その理由は、開発者が再利用したソースコードの候補となる ChatGPT 提案ソースコードが複数存在するためである。ChatGPT は、開発者のニーズに応えられる提案を1回で生成しないこともあるため、会話中に開発者の再利用候補となるソースコードが複数存在する。そのため、本論文では、その中から再利用されたソースコードを識別するために手作業で識別することにした。開発者が、ChatGPT の提案ソースコードからどのようにコードを再利用しているか、表4.1に示す4タイプへ分類した。このタイプ分類は、Stack Overflow からのソースコードの再利用に関する研究に基づいて、作成した [17].

表 4.1 ソースコードの再利用タイプ分類

ID	名称	定義
C1	完全コピー	開発者が全く修正せず, ChatGPT の提案コードをコピー&ペースト
C2	表面的な修正	開発者がソースコードの機能を変更せず, ChatGPT の提案コードをコピー&ペースト
C3	内部的な修正	開発者がソースコードの機能を変更し, ChatGPT の提案コードをコピー&ペースト
C4	アイデアの活用	開発者が ChatGPT の提案コードをコピー&ペーストしない代わりに, 提案を受け入れゼロからソースコードを実装

4.3 結果

Python, Java, JavaScript で記述された ChatGPT の提案ソースコードについて、開発者によって再利用されたソースコードの利用方法を分類した結果を表 4.2 に示す。表 4.2 に示すように、再利用されたソースコードの 83.5% が修正されずにそのまま再利用されている。これは、開発者は ChatGPT の提案したソースコードを再利用する際、修正せずにそのまま再利用するのが多いことを示唆している。また、表 4.2 より、開発者によって再利用された ChatGPT の提案ソースコードのうち、15.4% が何らかの形で修正されている。これは、開発者は ChatGPT の提案したソースコードを再利用する際に、何らかの修正を必要とする場合があることを示唆している。

開発者によって再利用されたソースコードの 83.5% において、開発者は ChatGPT の提案からソースコードを修正せずにコピー&ペーストしている。Listing 4.1 と図 4.2 にその例を示す。この Listing は、コミットに含まれる 1 ファイルの変更差分^(注 1)を示したものであり、緑色が追加の行を示している。図 4.2 は、再利用された ChatGPT の提案ソースコード^(注 2)である。これらを比較すると、開発者は ChatGPT の提案したソースコードをそのまま再利用していることがわかる。

また、再利用されたソースコードの 1.1% において、開発者は ChatGPT の提案からソースコードをコピー&ペーストして、開発者のシステムを動作させるために必要ではないと考えられる修正を加えている。Listing 4.2 にその例を示す。この Listing は、ChatGPT が提案したコードと、開発者が実際にコミットしたコード^(注 3)の差分を示したものである。このうち、緑色の行は開発者が新たに追加したコードを、赤色の行は ChatGPT の提案したコードから開発者が削除したコードを示している。開発者は ChatGPT の提案からソースコードをコピーし、Listing 4.2 の 1 行目にある、開発者にとって不必要な「path」モジュールの import 文を削除している。

さらに、再利用されたソースコードの 14.3% において、開発者は ChatGPT の提案からソースコードをコピー&ペーストして、開発者のシステムを動作させるために

(注 1): <https://github.com/tisztamo/Junior/commit/0f993913a84f5f62efe9277ad430b4f0d1fbbb8a#diff-79f4c7cdc0b026f6e556196f5911a449157807cfffdb3f9d33f0b5404b4ffc2f3>

(注 2): <https://chat.openai.com/share/bbf03be2-dc7b-4451-a614-1a542af33712>

(注 3): <https://github.com/tisztamo/Junior/commit/59b38e750652080c26d04274b097b4e9e7ef5d91#diff-cd467d7fe7963408f8456e69ca4684c6d3927fc2cb4e69d881f31b768daf0dcf>

必要な修正を加えている。Listing4.3にその例を示す。このListingも、Listing4.2と同様に、ChatGPTが提案したコードと、開発者が実際にコミットしたコード^(注4)の差分を示したものである。開発者はChatGPTの提案からソースコードをコピーし、Listing4.3の17行目のように、「"Translate"」というテキストの「Button」に、ボタンが押下されたのを見えるように機能させる「bind_return_key=True」を追加している。同様に、39行目では、イベントのトリガとして機能する「'\r」を削除している。

そして、再利用されたソースコードの1.1%において、開発者はChatGPTの提案したソースコードから、そのアイデアを取り入れて新しいソースコードを作成している。Listing4.4にその例を示す。このListingも、Listing4.2と同様に、ChatGPTが提案したコードと、開発者が実際にコミットしたコード^(注5)の差分を示したものである。開発者はChatGPTの提案したソースコードから、「marked」モジュールはデフォルトのエクスポートがないことを理解し、より正しいimport文に修正している。

(注4): [https://github.com/PyBites-Open-Source/italki-class-assistant/commit/](https://github.com/PyBites-Open-Source/italki-class-assistant/commit/8dc9e3fa38adff7af997bd5bf425ed242e5986fb#diff-b10564ab7d2c520cdd0243874879fb0a782862c3c902ab535faabe)

8dc9e3fa38adff7af997bd5bf425ed242e5986fb#diff-b10564ab7d2c520cdd0243874879fb0a782862c3c902ab535faabe

(注5): [https://github.com/tisztamo/Junior/commit/91ff33634444341b77697193f1cc1cb7bf628210#](https://github.com/tisztamo/Junior/commit/91ff33634444341b77697193f1cc1cb7bf628210#diff-79f4c7cdc0b026f6e556196f5911a449157807cffdb3f9d33f0b5404b4ffc2f3)

diff-79f4c7cdc0b026f6e556196f5911a449157807cffdb3f9d33f0b5404b4ffc2f3

表 4.2 ソースコードの再利用タイプ分類結果

ID	名称	合計	割合
C1	完全コピー	76	83.5%
C2	表面的な修正	1	1.10%
C3	内部的な修正	13	14.3%
C4	情報の活用	1	1.10%

```

1 + import express from 'express';
2 + import cors from 'cors';
3 + import { createPrompt } from './prompt/createPrompt.js';
4 + import { saveAndSendPrompt } from './interactiveSession/saveAndSendPrompt.js';
5 + import { api } from './config.js'; // Import your api object
6 +
7 + const app = express();
8 +
9 + // Enable CORS for all routes
10 + app.use(cors());
11 +
12 + app.use(express.json());
13 +
14 + app.post('/generate', async (req, res) => {
15 +   const { notes } = req.body;
16 +   const { prompt, saveto } = await createPrompt(notes);
17 +   const result = await saveAndSendPrompt(prompt, saveto, null, api); // pass the
    ↪   api object here
18 +   res.json({ prompt: result });
19 + });
20 +
21 + app.listen(3000, () => {
22 +   console.log('Server is running on port 3000');
23 + });

```

Listing 4.1: 開発者による ChatGPT 提案ソースコードの再利用例（再利用タイプ C1）



ChatGPT

Thank you for providing the `config.js`.

From the provided file, I see that you export `api` object which you can import in `server.js` and pass it as an argument to the `saveAndSendPrompt` function.

Here is the corrected `server.js`:

```
javascript Copy code

import express from 'express';
import cors from 'cors';
import { createPrompt } from './prompt/createPrompt.js';
import { saveAndSendPrompt } from './interactiveSession/saveAndSendPrompt.js';
import { api } from './config.js'; // Import your api object

const app = express();

// Enable CORS for all routes
app.use(cors());

app.use(express.json());

app.post('/generate', async (req, res) => {
  const { notes } = req.body;
  const { prompt, saveto } = await createPrompt(notes);
  const result = await saveAndSendPrompt(prompt, saveto, null, api); // pass
  res.json({ prompt: result });
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Please replace `import { api } from './config.js';` with the actual import statement that corresponds to your project structure.

If you are still getting the error, you might need to double-check how the `api` object is initialized in the `config.js` file. Make sure that the `OPENAI_API_KEY` environment variable is correctly set and the `ChatGPTAPI` is correctly initialized.

図 4.2 開発者が修正せず再利用した ChatGPT の提案ソースコード

```
1 - import path from 'path';
1  import { processPath } from './filesystem.js';
2  import { processInterfaceSection } from './processInterfaceSection.js';
3  import { printFolderStructure } from './printFolderStructure.js';
4
5  export const readAttention = async (attentionArray = [], attentionRootDir = '.')
  ↪ => {
6    try {
  @@ -9,6 +9,8 @@
9      if (trimmedLine.endsWith(' iface')) {
10         const filePath = trimmedLine.slice(0, -6).trim();
11         return processInterfaceSection(attentionRootDir, filePath);
12     } else if (trimmedLine.endsWith('/')) {
13         return printFolderStructure(attentionRootDir, trimmedLine.slice(0,
  ↪ -1).trim());
14     } else {
15         return processPath(attentionRootDir, trimmedLine);
16     }
```

Listing 4.2: 開発者による ChatGPT 提案ソースコードの再利用時の変更例（再利用タイプ C2）

```

1  import PySimpleGUI as sg
2  from translate import translate_text, LANGUAGES
3
4
5  def create_window():
    @@ -15,4 +15,4 @@
15      sg.InputText(key="-TEXT-", size=(50, 5))
16      ],
17 -      [sg.Button("Translate"), sg.Button("Exit")],
17 +      [sg.Button("Translate", bind_return_key=True), sg.Button("Exit")],
18      [sg.Text("Translation output:", size=(40, 1))],
    @@ -35,7 +35,7 @@
35      if event == sg.WINDOW_CLOSED or event == "Exit":
36          break
37
38      # Check if "Translate" button is clicked or Enter key is pressed
39 -      if event == "Translate" or event == '\r':
39 +      if event == "Translate":
40          target_language_key = {v: k for k, v in
↳ LANGUAGES.items()}[values["-LANG-"]]
41          translated_text = translate_text(values["-TEXT-"],
↳ target=target_language_key)

```

Listing 4.3: 開発者による ChatGPT 提案ソースコードの再利用時の変更例（再利用タイプ C3）

```

4 - import * as marked from 'marked';
4 + import { marked } from 'marked';

```

Listing 4.4: 開発者による ChatGPT 提案ソースコードの再利用時の変更例（再利用タイプ C4）

5. RQ2：開発者は ChatGPT が提案したソースコードを再利用するまでにどの程度の長さ ChatGPT と会話しているか？

5.1 動機

ソフトウェア開発における ChatGPT 利用動向の指標として、開発者が ChatGPT の提案を受け入れるまでのプロセスに着目する。開発者は、ChatGPT が提案したソースコードを受け入れられるまでにどの程度の回数会話しているかを調査する。会話の長さが長いことは、開発者は ChatGPT の提案ソースコードが必要とするソースコードであるかの判断を複数回しなければならないことを示している。そのため、ソースコードの再利用にかかる時間が増え、開発者の負担が大きくなることになる。ChatGPT 提案ソースコードの受理までにかかる会話の長さを調査することで、ソフトウェア開発における ChatGPT の利用動向を理解できる。

5.2 方法

RQ2 を答えるために、開発者が再利用した ChatGPT の提案ソースコードについて、開発者は、そのソースコードを受け入れるまでにどれくらい会話を繰り返しているかを調査した。調査方法は、以下の手順で進めた。この手順を図 5.1 に示す。

- 手順 i データセットから、コミットとそれに対応する開発者と ChatGPT の会話ログを取得する。
- 手順 ii 開発者と ChatGPT の会話ログから、ChatGPT 提案ソースコードを Python, Java, JavaScript で記述されたコードに絞る。
- 手順 iii NCDSearch を使用して ChatGPT 提案ソースコードに対して、対応するコミットから類似ソースコード片を検索する。
- 手順 iv 検索されたソースコード片から、再利用したコード片を手作業で識別する。
- 手順 v 再利用されたソースコードが、ChatGPT によって提案されるまでの会話回数を求める。

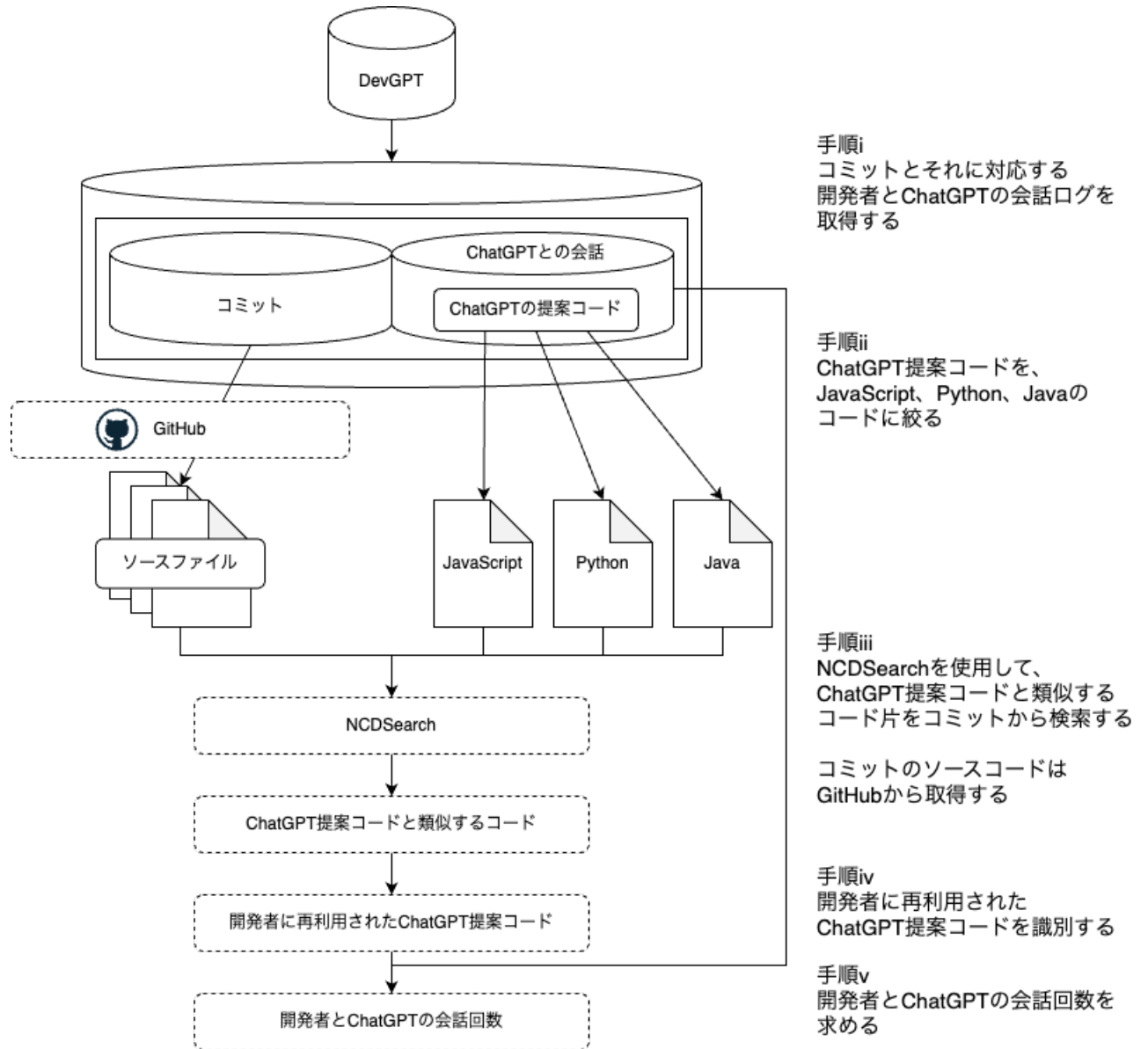


図 5.1 RQ1 における調査方法の概要図

手順 i~手順 iv は、RQ1 の手順 1~手順 4 と同様であるため、本章では省略する。

手順 v では、開発者が ChatGPT の提案ソースコードを再利用するまでの開発者と ChatGPT の会話回数を求める。会話回数は、開発者の与えたプロンプトに対して ChatGPT が応答するのを 1 回の会話として数えることにする。手順 i で取得した会話ログから、開発者によって再利用されたソースコードが、ChatGPT によって開発者に提案されるまでに行われた会話の回数を求める。この時、1 つの会話ログの中で、複数のソースコードの再利用がある場合、2 回目以降のソースコードの再利用については、その 1 つ前の再利用以降の会話を対象にプロンプトの回数を求めた。

5.3 結果

開発者が再利用した ChatGPT の提案ソースコードについて、開発者がそのソースコードを受理するまでの ChatGPT との会話回数を求めた結果を表 5.1 に示す。再利用の分類がタイプ C2 と C4 のソースコードは、それぞれ 1 つずつであったため、表には示していない。2 つのタイプのソースコードが開発者によって再利用されるまでに、タイプ C2 に分類されたソースコードは 1 回、タイプ C4 に分類されたソースコードは 4 回会話されている。

表 5.1 の情報に基づいて作成した箱ひげ図を図 5.2, 図 5.3, 図 5.4 に示す。図 5.2 に示すように、開発者が ChatGPT の提案ソースコードを受け入れるまでに、平均が 3.65 回で受け入れており、全体の半分を占めるソースコードが 1 回から 4 回の会話で受け入れられている。これは、開発者が ChatGPT と複数回会話してから、ソースコードを再利用していることが多いことを示唆している。

また、図 5.3 に示すように、再利用されたソースコードのうち、再利用の分類がタイプ C1 のソースコードは、平均 3.49 回で受け入れられており、全体の半分を占めるソースコードが 1 回から 3 回の会話で受け入れられている。これは、開発者がソースコードを ChatGPT から修正せずそのまま再利用する場合、ChatGPT との会話回数が少なくなる傾向にあることを示唆している。

一方で、図 5.4 に示すように、再利用の分類がタイプ C3 のソースコードは、受け入れられるまでに平均で 4.77 回会話されており、全体の半分が 2 回から 6.5 回の会話で受け入れられている。これは、開発者がソースコードを ChatGPT から内部的

に修正して再利用する場合，修正せずに再利用する場合と比較して，ChatGPT との会話回数が多くなる傾向にあることを示唆している。

表 5.1 開発者と ChatGPT の会話回数の統計量

	C1	C2	C3	C4	全体
平均	3.49	-	4.77	-	3.65
第 1 四分位数	1.00	-	2.00	-	1.00
中央値	2.00	-	4.00	-	2.00
第 3 四分位数	3.00	-	6.50	-	4.00

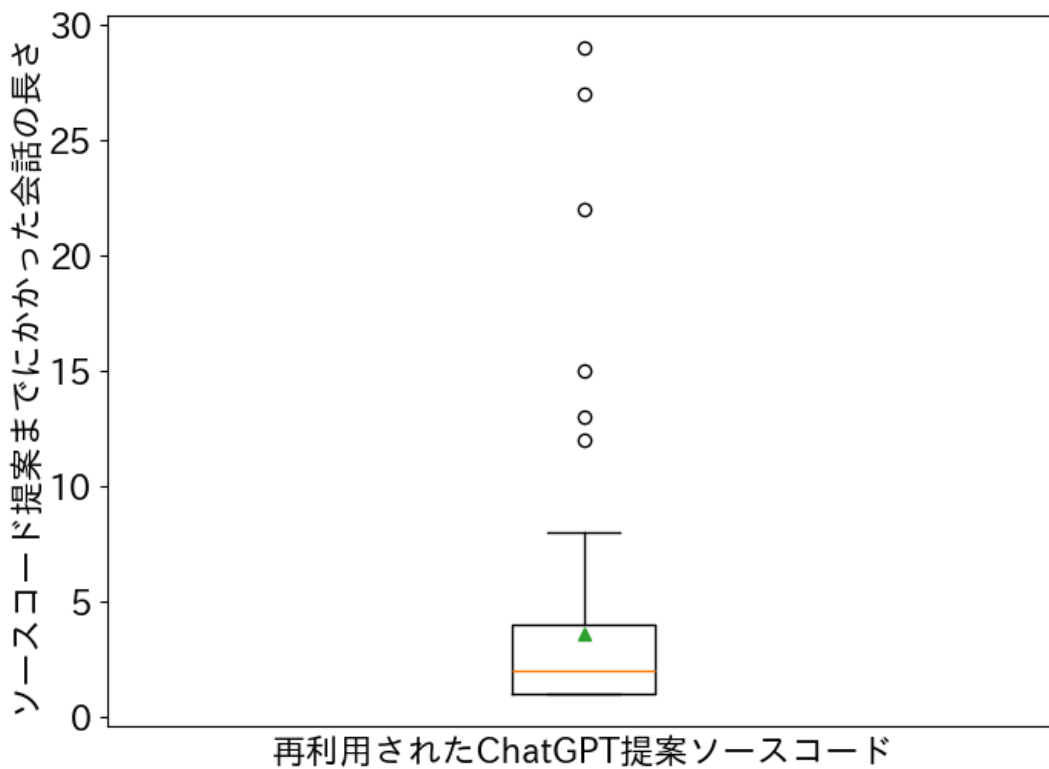


図 5.2 開発者が再利用したソースコード（全体）を ChatGPT が提案するまでの会話長さの分布

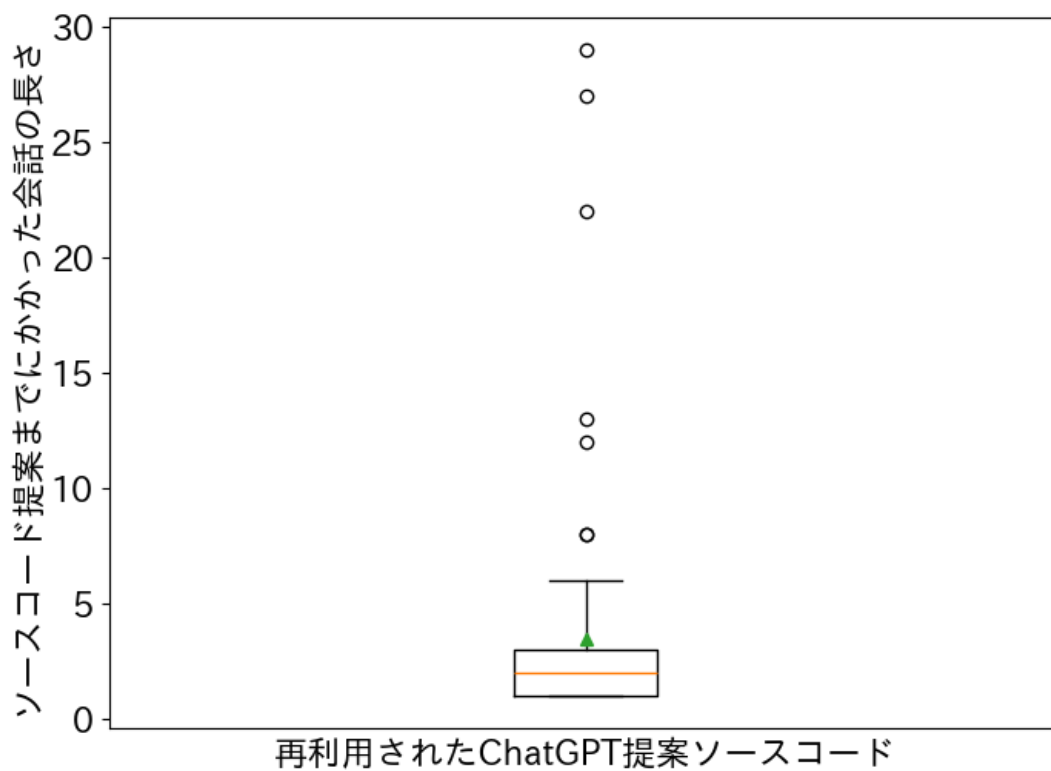


図 5.3 開発者が再利用したソースコード（タイプ C1）を ChatGPT が提案するまでの会話長さの分布

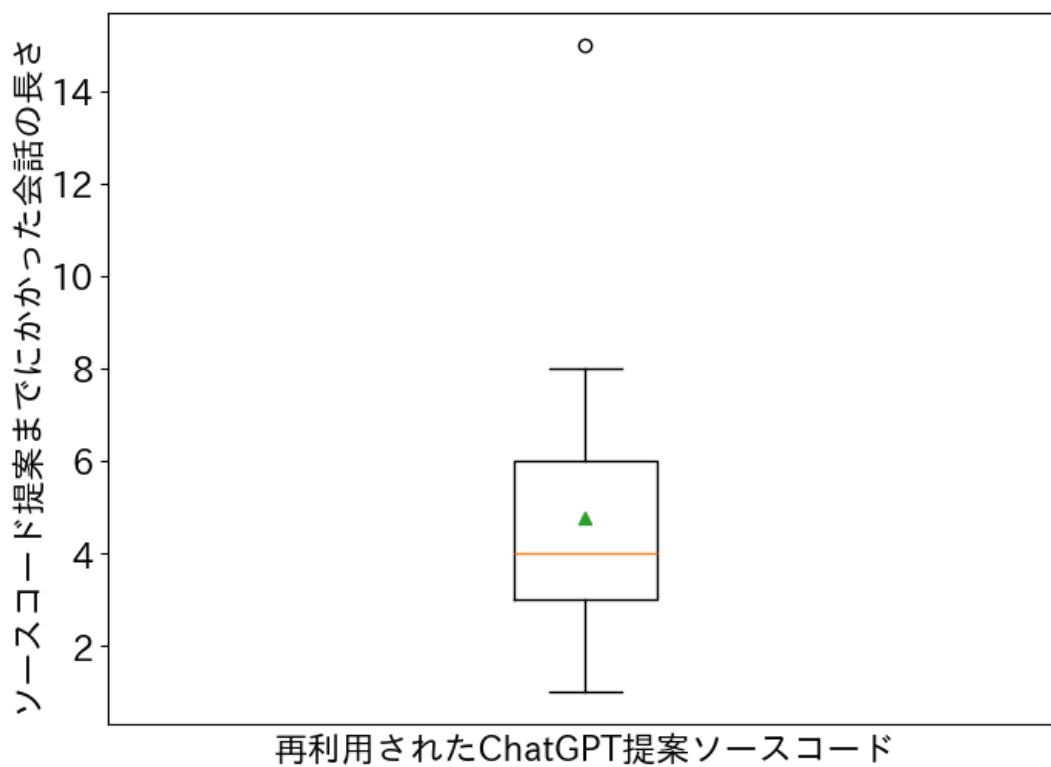


図 5.4 開発者が再利用したソースコード（タイプ C3）を ChatGPT が提案するまでの会話長さの分布

6. 議論

6.1 RQ1 ChatGPT が提案したソースコードは開発者によってどのように再利用されているか？

ChatGPT が提案したソースコードは開発者によってどのように再利用されているかを調査した。その結果、第4章に示したように、表4.2より、ChatGPT が提案したソースコードを再利用する際、修正することが少ないとわかった。これは、開発者が必要とするソースコードを ChatGPT が提案できていることが多いと考えられ、ChatGPT を開発者のプロンプトに関連するソースコードを返すことができるコード検索ツールとして活用できる可能性があると考えられる。しかし、表4.2より、ChatGPT が提案したソースコードを再利用する際、何らかの修正を必要とする場合があることも分かった。すなわち、ChatGPT は開発者の必要とするソースコードを完全に提案できていないこともあることを示唆している。そのため、ChatGPT をコード検索ツールとして利用する上で、提案されたソースコードを開発者のシステムに統合する際、自動化機能があると、ChatGPT を開発者の障壁をより取り除いたコード検索ツールとして活用できると考えられる。

本論文では、ChatGPT を活用したコミットについて、ChatGPT の提案ソースコードが開発者によってどのように再利用されたかを調査したため、その再利用した提案ソースコードがバグなど問題を引き起こすことがなかったかどうかはわかっていない。そのため、今後は再利用された ChatGPT の提案ソースコードの動向を調査し、その影響について明らかにする必要があると考えられる。また、本論文では、ChatGPT との会話をリンクしたコミットについて調査したため、ChatGPT から何かしら有用な提案を得られ、それを利用したケースに、調査対象が限られていると考えられる。ChatGPT を利用しようとしたが、ChatGPT から開発者にとって有用な提案を得られなかったケースも存在すると考えられ、そのようなケースについても調査する必要があると考えられる。

さらに、Stack Overflow からのソースコード再利用について調査した研究 [5] では、調査されたファイルのうち 20.5%が、Stack Overflow からソースコードを修正せずそのまま再利用しており、31.5%が、何らかの修正をして再利用していることが報告さ

れている。これを本調査の結果と比較すると、開発者が ChatGPT の提案ソースコードを再利用する際、修正することは少なく、より完全なコピーが多く、ChatGPT の提案したソースコードは、Stack Overflow のソースコードよりも再利用される際の修正が少ない傾向にあるとわかる。すなわち、ChatGPT からのソースコードの再利用は、Stack Overflow を利用した場合と比べて、ソースコードを開発者のシステムに統合する障壁が低いと考えられる。

6.2 RQ2 開発者は ChatGPT が提案したソースコードを再利用するまでにどの程度の回数 ChatGPT と会話しているか？

開発者は ChatGPT の提案したソースコードを再利用するまでにどの程度の回数 ChatGPT と会話しているかを調査した。その結果、第 5 章に示すように、開発者は ChatGPT が提案したソースコードを再利用するまでに、ChatGPT と 1 回から 4 回程度の会話をしていることが多いとわかった。開発者が ChatGPT の提案ソースコードを再利用するまでどれくらい会話をしているかは、開発者が ChatGPT の提案をどの程度検討してから受け入れいているかを示す指標となる。平均 3.65 回という会話数は、開発者が ChatGPT に複数回提案を提示させてから、受理する提案を決定していることを示唆している。すなわち、ChatGPT の提案を受け入れるまでに開発者が ChatGPT との会話を重ね、自身の必要とするソースコードを得ようとしていると考えられる。

また、表 5.1 と図 5.3 より、再利用されたソースコードのうち、再利用の分類がタイプ C1 のソースコードは、平均 3.49 回で受け入れられており、全体の半分を占めるソースコードが 1 回から 3 回の会話で受け入れられていることがわかった。これは、ChatGPT からソースコードを修正なしで再利用する場合、会話が短くなる傾向にあることを示唆している。すなわち、開発者が ChatGPT に提供するプロンプトによっては、より少ない回数のプロンプトで必要とするソースコードを ChatGPT に提案させることができると考えられる。さらに、表 5.1 と図 5.4 より、再利用の分類がタイプ C3 のソースコードは、受け入れられるまでに平均で 4.77 回会話されており、全体の半分を占めるソースコードが 2 回から 6.5 回の会話で受け入れられていることがわかった。これは、図 5.3 の結果と比較すると、開発者が ChatGPT の提

案ソースコードを何らかの内部的な修正をして再利用する場合、修正なしで再利用する場合より会話が長くなる傾向にあることを示唆している。つまり、開発者が ChatGPT に提供するプロンプトによっては、必要とするものにより近いソースコードを ChatGPT に提案させるために、より多くのプロンプトが必要になると考えられる。さらに、開発者の必要とするものに近いソースコードを得られたとしても、それを再利用するためには、何らかの内部的な修正が必要になることが多いと考えられる。このように、再利用するソースコードの探索と統合に手間がかかると、開発者にとって大きな負担となると考えられる。そのため、開発者が再利用タスクのためのコード検索ツールとして ChatGPT を使用する際、必要とするソースコードを安定して得るために、開発者が ChatGPT に提供するプロンプトの最適化が必要であると考えられる。

7. 結言

本研究では、開発者が再利用した ChatGPT の提案ソースコードに着目し、開発者が ChatGPT の提案ソースコードをどのように再利用しているか調査した。再利用の方法、再利用までの会話回数の2点に焦点を当てて調査した。

調査の結果、開発者は ChatGPT の提案ソースコードを何らかの形で修正して再利用することもしばしばあるが、修正せずそのまま再利用することが多いとわかった。すなわち、開発者の提供した情報に基づいて、開発者の必要とするソースコードを提案する、コード検索ツールとして ChatGPT を活用できる可能性があると考えられる。また、開発者が ChatGPT の提案ソースコードを再利用するまでの会話について、修正せずに再利用する場合、回数は少ないことが多く、修正して再利用する場合、修正せずに再利用する場合と比べて回数はより増加することが多いとわかった。これは、ChatGPT からソースコードを再利用する際に、修正なしで再利用する時は、会話が短くなる傾向にあることを示唆している。つまり、ChatGPT に適切にプロンプトを提供できれば、開発者はより少ない回数でより必要とするソースコードに近い提案を ChatGPT に促せられる可能性があると考えられる。そのため、ソフトウェア開発において ChatGPT を安定して活用するためには、ChatGPT に提供するプロンプトを最適化することで、より効率的な再利用を可能にする必要があると考えられる。

謝辞

本研究を行うにあたり、研究課題の設定や研究に対する姿勢、本報告書の作成に至るまで、全ての面で丁寧なご指導を頂きました。本学情報工学・人間科学系水野修教授ならびに崔恩瀨助教に厚く御礼申し上げます。本報告書執筆にあたり貴重な助言を多数頂きました。ソフトウェア工学研究室の皆さん、いつも研究室で温かく迎えてくれたあくあたん、学生生活を通じて著者の支えとなった家族や友人に深く感謝致します。

参考文献

- [1] OpenAI, Introducing ChatGPT, (オンライン), 入手先 <<https://openai.com/blog/chatgpt>> (参照 2024-02-07).
- [2] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D.C. Schmidt, “A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT,” Feb. 2023. arXiv:2302.11382 [cs].
- [3] L. Tung ChatGPT can write code. Now researchers say it’s good at fixing bugs, too, (オンライン), 入手先 <<https://www.zdnet.com/article/chatgpt-can-write-code-now-researchers-say-its-good-at-fixing-bugs-too/>> (参照 2024-02-07).
- [4] A. Borji, “A Categorical Archive of ChatGPT Failures,” April 2023. arXiv:2302.03494 [cs].
- [5] B. Guo, X. Zhang, Z. Wang, M. Jiang, J. Nie, Y. Ding, J. Yue, and Y. Wu, “How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection,” Jan. 2023. arXiv:2301.07597 [cs].
- [6] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza, A. Janz, K. Kanclerz, A. Kocoń, B. Koptyra, W. Mieszczewicz, P. Miłkowski, M. Oleksy, M. Piasecki, c. Radliński, K. Wojtasik, S. Woźniak, and P. Kazienko, “ChatGPT: Jack of all trades, master of none,” Information Fusion, vol.99, pp.1–37, Nov. 2023.
- [7] S. Mitrović, D. Andreoletti, and O. Ayoub, “ChatGPT or Human? Detect and Explain. Explaining Decisions of Machine Learning Model for Detecting Short ChatGPT-generated Text,” Jan. 2023. arXiv:2301.13852 [cs].
- [8] StackOverflow Temporary policy: Generative AI (e.g., ChatGPT) is banned, (オンライン), 入手先 <<https://meta.stackoverflow.com/questions/421831/temporary-policy-generative-ai-e-g-chatgpt-is-banned>> (参照 2024-02-07).
- [9] S. Kabir, D.N. Udo-Imeh, B. Kou, and T. Zhang, “Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions,” Aug. 2023. arXiv:2308.02312 [cs].

- [10] B. Yetiştirgen, I. Özsoy, M. Ayerdem, and E. Tüzün, “Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT,” Oct. 2023. arXiv:2304.10778 [cs].
- [11] H. Tian, W. Lu, T.O. Li, X. Tang, S.-C. Cheung, J. Klein, and T.F. Bissyandé, “Is ChatGPT the Ultimate Programming Assistant – How far is it?,” Aug. 2023. arXiv:2304.11938 [cs].
- [12] R. Cottrell, R.J. Walker, and J. Denzinger, “Semi-automating small-scale source code reuse via structural correspondence,” In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, pp.214–225, ACM, Atlanta Georgia, Nov. 2008.
- [13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software, Addison-Wesley Longman Publishing Co., Inc., Boston, 1995.
- [14] S. Wang, D. Lo, and L. Jiang, “Active code search: incorporating user feedback to improve code search relevance,” In Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, pp.677–682, ASE ’14, ACM, New York, NY, USA, Sept. 2014.
- [15] T. Ishio, N. Maeda, K. Shibuya, and K. Inoue, “Cloned Buggy Code Detection in Practice Using Normalized Compression Distance,” In Proceeding of the 34th IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.591–594, IEEE, Madrid, Spain, Sept. 2018.
- [16] T. Ishio, N. Maeda, K. Shibuya, K. Iwamoto, and K. Inoue, “NCDSearch: Sliding Window-Based Code Clone Search Using Lempel-Ziv Jaccard Distance,” IEICE Transactions on Information and Systems, vol.E105-D, no.5, pp.973–981, May 2022.
- [17] Y. Wu, S. Wang, C.-P. Bezemer, and K. Inoue, “How do developers utilize source code from stack overflow?,” Empirical Software Engineering, vol.24, no.2, pp.637–673, April 2019.
- [18] T. Xiao, C. Treude, H. Hata, and K. Matsumoto, “Devgpt: Studying developer-

chatgpt conversations,” In Proceedings of the 21st International Conference on Mining Software Repositories (MSR), pp.●●–●●, Lisbon, Portugal, 2024.