Multi-language Software Development in the LLM Era: Insights from Practitioners' Conversations with ChatGPT

Lucas Aguiar
State University of Ceará (UECE)
Fortaleza, Brazil
lucas.aguiar@aluno.uece.br

Matheus Paixao
matheus.paixao@uece.br
State University of Ceará (UECE)
Fortaleza, Brazil

Rafael Carmo
Federal University of Ceará (UFC)
Fortaleza, Brazil
carmorafael@virtual.ufc.br

Matheus Freitas
State University of Ceará (UECE)
Fortaleza, Brazil
fernandes.matheus@aluno.uece.br

Eliakim Gama
State University of Ceará (UECE)
Fortaleza, Brazil
eliakim.gama@aluno.uece.br

Antonio Leal
State University of Ceará (UECE)
Fortaleza, Brazil
leal.sobrinho@uece.br

Edson Soares
State University of Ceará (UECE)
Fortaleza, Brazil
edson.araujo@aluno.uece.br

ABSTRACT

Non-trivial software systems are commonly developed using more than a single programming language. However, multi-language development is not straightforward. Nowadays, tools powered by Large Language Models (LLMs), such as ChatGPT, have been shown to successfully assist practitioners in several aspects of software development. This paper reports a preliminary study aimed to investigate to what extent ChatGPT is being used in multi-language development scenarios. Hence, we leveraged DevGPT, a dataset of conversations between software practitioners and ChatGPT. In total, we studied data from 3,584 conversations, comprising a total of 18,862 code snippets. Our analyses show that only 18.33% of the code snippets suggested by ChatGPT are written in the same programming language as the primary language in the repository where the conversation was shared. In an in-depth analysis, we observed expected scenarios, such as 31.54% of JavaScript snippets being suggested in CSS repositories However, we also unveiled surprising ones, such as Python snippets being largely suggested in C++ repositories. After a qualitative open card sorting of the conversations, we found that in 70% of them developers were asking for coding support while in 57% developers used ChatGPT as a tool to generate code. Our initial results indicate that not only LLMs are being used in multi-language development but also showcase the contexts in which such tools are assisting developers.

KEYWORDS

Multi-language Software Development, ChatGPT, Empirical Studies

ESEM '24, October 24–25, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-1047-6/24/10…$15.00
https://doi.org/10.1145/3674805.3690755

ACM Reference Format:
Lucas Aguiar, Matheus Paixao, Rafael Carmo, Matheus Freitas, Eliakim Gama, Antonio Leal, and Edson Soares. 2024. Multi-language Software Development in the LLM Era: Insights from Practitioners' Conversations with ChatGPT. In *Proceedings of the 18th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24), October 24–25, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3674805.3690755

1 INTRODUCTION

Software programs are symbolic expressions depicted in a programming language, describing the tasks practitioners want the computer to perform [1]. Managing complexity is essential in programming. Developers build abstractions to hide details and establish conventional interfaces, enabling software system construction. A programmer's ability to create abstractions depends on the features and design choices of the programming language [1]. Thus, different programming languages vary in their strategies for handling complexity, with some providing features that better fit certain categories of software problems.

Multi-language software development (MLSD) consists of employing more than one programming language to develop a software system [5, 19]. MLSD is a choice effortlessly justifiable by the increasing flexibility of combining the complementary strengths of different programming languages [13, 14, 19]. MLSD has been a predominant practice for years [12, 15, 21]. In fact, a study showed that considering both industry and open-source, over 80% of software projects employ two or more languages [9, 23]. However, MLSD is not straightforward. Research points to several challenges faced by practitioners, such as the use of multiple programming languages significantly increasing bug proneness [6], the occurrence of design smells involving communication between different technologies [27], and specific languages that are statistically more defect-prone when used in a multi-language setting [10, 13].

With the advent of powerful Large Language Models (LLMs), AI-based assistant tools to aid practitioners in coding are becoming widely adopted. Recent research shows that LLMs have been

assisting practitioners in a variety of challenges, from dealing with bugs [11], recommending meaningful names [16], automated documentation [17], code comprehension [20], and code summaries [24].

In this context, our intuition suggested that software practitioners may have been using ChatGPT to assist in MLSD tasks. Hence, this paper sets out to systematically investigate the extent to which this perception is accurate. By performing this empirical study, we expect to validate whether and how practitioners are using ChatGPT in MLSD, which, in turn, will shed light on and motivate further studies regarding MLSD and LLMs.

To this end, we leveraged DevGPT [26], a dataset of conversations between software practitioners and ChatGPT, which includes links to the GitHub repositories from which the conversations were originally shared. We studied data from 3,584 conversations, comprising a total of 18,862 code snippets suggested by ChatGPT.

Based on our findings, only 18.33% of the code snippets ChatGPT suggests to practitioners are written in the repository's primary programming language. In addition, an in-depth analysis of the languages in which the snippets suggested by ChatGPT were written showcases interesting remarks. While we observed expected scenarios, such as 31.54% of JavaScript snippets being suggested in CSS repositories, we also unveiled surprising ones, such as Python snippets being largely suggested in C++ repositories. In addition, we performed an open card sorting procedure on a sample of the conversations to find out the most common contexts ChatGPT suggests snippets in a different language. As a result, we found that in 70% of the conversations developers were seeking coding support regarding a different language while in 57% developers used ChatGPT as a tool to generate code in different languages.

2 BACKGROUND

2.1 Multi-language Software Development

Multi-language software development (MLSD) consists of employing more than one programming language to develop a software system [5, 19]. Utilising multiple programming languages in large-scale software development has been a predominant practice for years [12, 15, 21]. This approach allows, in principle, employing a language-as-a-tool: using distinct languages to solve specific problems and combining the complementary strengths of different languages to solve different software problems. A survey suggested a rather universal usage of MLSD among open-source projects, with a mean number of 5 programming languages used per project [18].

However, MLSD is not straightforward. Its practice introduces a considerable number of additional challenges related to increased complexity and the need for proper interfaces and interactions between the different languages and environments [2]. The software engineering research community has provided common patterns and guidelines to support the development, maintenance, and evolution of such systems [3–6, 15], but it has hardly covered the demand. Hence, modern technologies and tools, such as LLMs and ChatGPT, may be sought by practitioners to assist in these challenges.

2.2 The DevGPT Dataset

DevGPT is a recently published dataset comprised of conversations between software practitioners and ChatGPT [26]. The conversations were shared by the practitioners themselves through OpenAI's shared links[1]. A shared link is a feature that allows users to generate a unique URL for a ChatGPT conversation, which can then be shared with friends, colleagues, and collaborators.

DevGPT collects conversations shared on different mediums. At the time of writing this paper, DevGPT is composed of conversations shared on GitHub and HackerNews. For GitHub, the dataset collects conversations shared on the body of textual Files (source code, README, etc), Commit messages, Issues, Pull Requests, and Discussions. For this paper, we consider each location in which a conversation can be shared on GitHub as a different **source**.

Within a certain conversation, we have access to all the prompts written by the practitioner and all the responses given by ChatGPT, including eventual code snippets it may have suggested. For each suggested snippet, ChatGPT also identifies the programming language in which the snippet is written. We refer to this as the **snippet language**. DevGPT also provides metadata regarding the repository in which the conversation was shared, such as URL, name, etc. Most importantly for this study, it also provides the repository's primary programming language, as identified by GitHub itself. We refer to this as **repository language**.

For this paper, we define **native snippet** as a code snippet suggested by ChatGPT in which the snippet's and repository's programming language is the same. Accordingly, we define **foreign snippet** as a code snippet that is written in a programming language other than the repository's language. By characterizing native and foreign snippets, we can investigate the extent in which ChatGPT is being used in MLSD scenarios.

For example, consider the conversation in the following shared link[2]. This conversation was originally shared in a pull request[3] of the Darklang[4] project, where developers are building a new programming language and infrastructure "to make it easy to build backends and CLIs". Even though the Darklang project is primarily written in F#, this conversation resulted in three C# snippets being suggested by ChatGPT. Upon close inspection of this pull request (see Section 3.3), a reviewer mentioned the F# solution proposed by the developer was 'overkill', suggesting one of the C# snippets provided by ChatGPT as more adequate. For this conversation, F# is the repository's native language. Hence, the three C# snippets suggested by ChatGPT are considered foreign snippets. One could argue that the developer didn't specify the usage of F# in the prompt, but not only ChatGPT replied with a snippet in the language it saw most fit, but the developer used the C# solution instead of asking ChatGPT to create a new solution in F#.

3 EMPIRICAL METHODOLOGY

This study aims to answer the following research questions:

- **RQ1**: To what extent does ChatGPT suggest native and foreign snippets to practitioners?
- **RQ2**: What is the distribution of programming languages in foreign snippets suggested by ChatGPT?
- **RQ3**: In which contexts do foreign snippets emerge in conversations between ChatGPT and practitioners?

[1] https://help.openai.com/en/articles/7925741-chatgpt-shared-links-faq
[2] https://chat.openai.com/share/2a6f10f0-d45d-4e71-ac57-584570baeda8
[3] https://github.com/darklang/dark/pull/5063
[4] https://github.com/darklang/dark

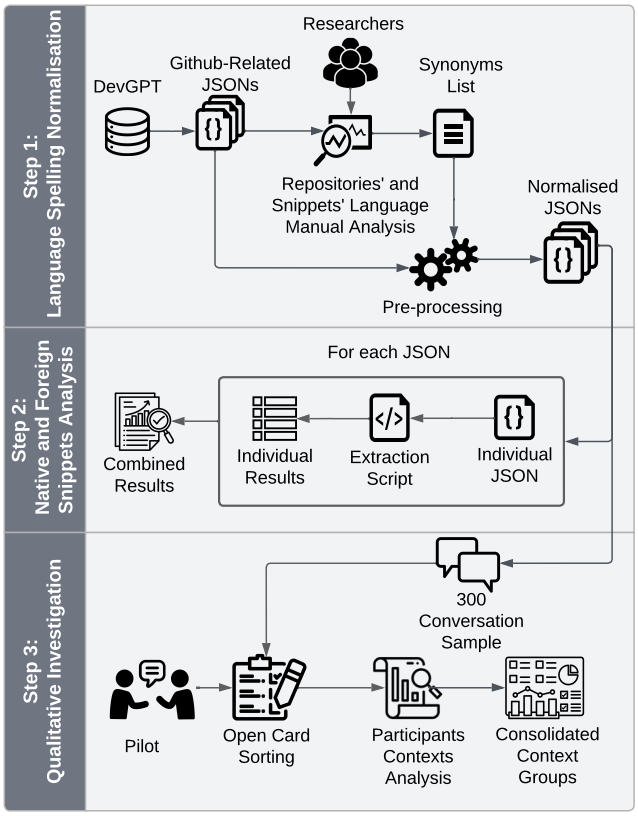

Figure 1: Empirical methodology employed in this paper

To answer our research questions, we followed the empirical methodology illustrated in Figure 1. We made the data, code and artifacts generated in this study available in a replication package [7].

3.1 Step 1: Language Spelling Normalization

To identify native and foreign snippets suggested by ChatGPT, we need to compare the primary programming language in a project's repository and the language in which a suggested snippet is written. Hence, we took the DevGPT dataset (snapshot 20231012) and considered only the conversations shared within GitHub. Since the HackerNews conversations are not linked to a specific repository, its data does not fit our study.

In the dataset, the data regarding each different GitHub source is presented in a separate JSON file. First, we extracted each repository's primary language using the `RepoLanguage` attribute. Second, we extracted the programming language of each snippet using the `Type` attribute within the `ListOfCode` object. Finally, we created a unique list of all programming languages listed as a repository or snippet language. In total, there were 75 and 125 programming languages listed as repository and snippet languages, respectively.

We noticed the languages' spelling was not uniform. For instance, JavaScript appeared as 'javascript' and 'js'. We also noticed language extensions listed as standalone languages, such as 'jsx', which is a JavaScript extension. To address this, we created a list of synonyms to group different spellings of each programming language. This list was manually created by the authors in an iterative manner. Using this list, we normalized the language spelling in the JSON files throughout repositories and snippets. After normalization, our dataset contains 72 repository languages and 100 snippet languages.

3.2 Step 2: Native and Foreign Snippets Analysis

Using the normalized JSON files as input, we executed a Python script to identify the native and foreign snippets suggested by ChatGPT. For each JSON, we grouped conversations by repository language. Consider the conversations shared in commit messages within repositories whose primary language is JavaScript, for example. After processing the commits JSON, we identified 7 conversations, in which 78 code snippets were suggested by ChatGPT. For each snippet, we checked the language the snippet was written. Snippets written in JavaScript were considered native and snippets written in other languages were considered foreign. Using this procedure, we identified 18 and 60 native and foreign snippets, respectively. These results indicate that, for conversations shared within commit messages of JavaScript repositories, ChatGPT suggestions consisted of 23.08% and 76.92% of native and foreign snippets, respectively.

We performed this analysis for all JSON files and all programming languages in our dataset. Finally, we combined the results of all JSONs to get an understanding of the data as a whole. The results of this step were used to answer RQ1 and RQ2.

3.3 Step 3: Qualitative Investigation

To fully explore developers' usage of ChatGPT in MLSD scenarios, only employing quantitative analyses (RQ1 and RQ2) is not sufficient. In this manner, a qualitative investigation may assist in obtaining an in-depth understanding of the contexts in which ChatGPT suggests foreign snippets. To achieve this goal, we employed an open card sorting procedure on a sample of the conversations where foreign snippets were suggested.

Card sorting is a thematic analysis approach where participants categorize similar items into different groups [22]. An item can be in more than one group. In this study, we categorize conversations where ChatGPT suggested a foreign snippet, aiming to identify the conversation **context** the snippet emerged. The context indicates the overall theme, such as 'coding support' and 'documentation writing'. Closed card sorting, where participants categorize into predefined contexts, was not feasible due to the exploratory nature of this study. Thus, we employed open card sorting, allowing participants to freely create groups to categorize the conversations.

To avoid personal biases in qualitative investigations, the data must by analyzed independently by more than one person [25]. Given the subjective nature of open card sorting, we selected three participants to independently perform the procedure. To ensure different background and levels of expertise, we selected one undergraduate, one master's student and one PhD candidate in computer science. All participants had previous experience using ChatGPT.

To ensure all participants fully understood open card sorting, we prepared a presentation and ran a pilot study. For the pilot, we randomly sampled 10 conversations in which ChatGPT suggested a foreign snippet. Next, we asked the participants to perform open card sorting as a group, i.e., to categorize the 10 conversations into thematic groups and provide a description for each created

group. In total, the pilot study took 40 minutes. The participants were asked to provide comments and feedbacks. These were later incorporated into the study's design.

Since the goal of our open card sorting is to identify the contexts in which foreign snippets emerge, we only considered conversations in which foreign snippets were suggested by ChatGPT. To enhance the confidence that the conversations actually impacted the projects's code bases, we limited open card sorting to conversations originating from GitHub sources that are closely related to source code, namely Files, Commits and Pull Requests. Hence, in total, the population for our qualitative investigation consisted in 1340 conversations, where 737 (55%), 483 (36%) and 121 (9%) originated from Files, Commits and Pull Requests, respectively. Next, we took a stratified statistically significant sample with 95% confidence. This resulted in a sample of 300 conversations (165 from Files, 108 from Commits and 27 from Pull Requests).

Our open card sorting consisted in each of the three participants independently categorizing the 300 conversations in our sample into different thematic contexts. Participants were free to create as many context as they wanted, but could only categorize a conversation in up to three contexts to ensure consistency. Finally, each participant wrote a description of each context. After the participants concluded their sorting, one of the authors with knowledge in both software development and LLMs acted as a mediator, tasked with analyzing the contexts, identifying similarities and merging the categorizations. This resulted in a unique categorization of conversations into contexts based on the three independent sortings.

4 EMPIRICAL RESULTS

4.1 RQ1:To what extent does ChatGPT suggest native and foreign snippets to practitioners?

Due to space constraints, we could not include the results for all programming languages in the paper. Hence, to select which languages' results to present, we first counted the total number of snippets for each language in the dataset. Next, we ranked the languages by the number of snippets and selected the languages above the third quartile in the distribution. This resulted in 18 languages whose total number of suggested snippets sums up to 16,965, consisting of 89.94% of the 18,862 total snippets in the dataset.

Table 1, presents the number of snippets and the percentage of native snippets for each language and each GitHub source. Languages marked with an asterisk (*) were only identified in repositories, not in snippets. This is likely due to the language identification process being different in the repositories and snippets. We discuss this matter in more detail in the Threats to Validity section.

Consider the CSS language, for instance. We identified 4,017 snippets suggested in conversations shared in Files. Out of these snippets, only 8.84% are native, i.e., snippets written in CSS. When considering the suggested snippets in CSS repositories in all datasets, only 7.26% of snippets are native. This indicates that 92.74% of the snippets suggested by ChatGPT for CSS repositories are written in other languages. When looking at other languages, we can observe a similar pattern. For C++ and JavaScript, 79.90% and 76.16% of suggested snippets are foreign, respectively. However, there are exceptions, such as the C# and CodeQL languages, in which one can observe 60.18% and 58.08% of native snippets.

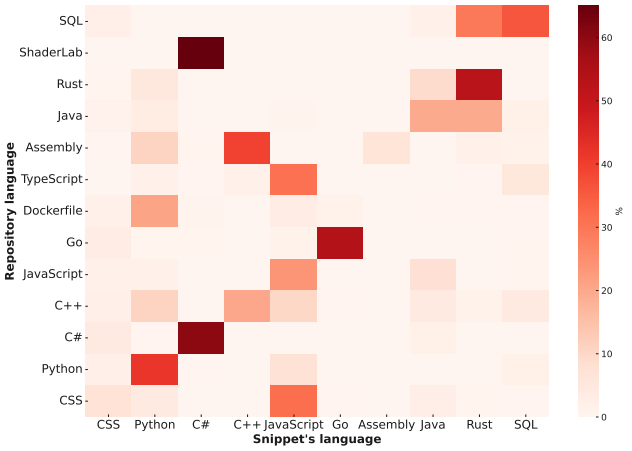

Figure 2: Distribution of foreign snippets suggested within repositories of different primary languages

As an answer to RQ1, we have shown that more than 75% of snippets suggested by ChatGPT are foreign, i.e., snippets in languages other than the repository's primary language. This indicates that, most of the time, conversations between practitioners and ChatGPT are related to multiple languages. This evidences how ChatGPT is being employed in MLSD scenarios. In RQ3, we qualitatively investigate the contexts foreign snippets are suggested by ChatGPT.

4.2 RQ2: What is the distribution of programming languages in foreign snippets suggested by ChatGPT?

Figure 2 showcases the distribution of foreign snippets suggested by ChatGPT in conversations with practitioners. The Y axis indicates the primary languages of the repositories while the X axis represents the languages in which snippets were suggested. A cell's darkness is proportional to the number of snippets suggested in a certain language. In CSS repositories, for example, there were more snippets suggested in JavaScript than in CSS. Differently, almost all snippets suggested in C# repositories were written in C#. To improve the visualization, we only included in the figure the languages with most repositories and snippets.

By looking at the figure, some of the observations were expected, such as the strong relationship between CSS and JavaScript. Considering all foreign snippets suggested for CSS repositories, 31.54% are written in JavaScript. Other results were unexpected. For instance, C++ repositories were found to have a large occurrence of Python snippets. However, the opposite is not true. In Python repositories, the most popular foreign snippets are written in JavaScript.

The recurrent pattern within the data showcases close to uniform distribution of foreign snippets into multiple languages. Considering the entire dataset, for all languages (not only the ones in the figure), the average number of unique foreign languages is 10. This is a strong indicator of how MLSD is indeed prevalent in the software industry with ChatGPT being an important tool to assist developers in their MLSD tasks.

As an answer to RQ2, our results show that in general, foreign snippets are written in several different languages (average of 10+),

Table 1: Number of snippets and percentage of native snippets for different languages and different GitHub sources in which the snippets were shared. Languages marked with an asterisk (*) were only identified in repositories, not in snippets.

Repository Language	Files		Commits		Issues		Pull Requests		Discussions		Total	
	Snippets	Native (%)	Snippets	Native (%)	Snippets	Native (%)	Snippets	Native (%)	Snippets	Native (%)	Snippets	Native (%)
CSS	4,017	8.84	1,284	2.18	13	23.08	0	-	0	-	5,314	7.26
Python	1,028	48.25	153	31.37	653	43.49	286	19.23	45	46.67	2,165	41.76
HTML	1,590	4.65	99	8.08	105	15.24	4	25.00	27	0.00	1,825	5.43
C#	1,231	60.03	0	-	13	76.92	2	0.00	2	100.00	1,248	60.18
C++	802	28.30	0	-	28	42.86	391	1.79	3	0.00	1,224	20.10
JavaScript	604	21.69	78	23.08	208	32.21	44	13.64	10	30.00	944	23.84
Jupyter Notebook*	742	0.00	0	-	21	0.00	0	-	2	0.00	765	0.00
C	694	5.04	8	75.00	56	28.57	0	-	1	100.00	759	7.64
Go	487	57.29	6	0.00	23	21.74	17	47.06	8	0.00	541	53.97
Batch	453	0.22	7	0.00	0	-	0	-	0	-	460	0.22
Dockerfile	237	2.53	76	0.00	0	-	0	-	0	-	313	1.92
TypeScript	49	46.94	0	-	183	14.21	36	36.11	30	13.33	298	22.15
Assembly	297	6.40	0	-	0	-	0	-	0	-	297	6.40
Java	101	19.80	0	-	67	35.82	76	5.26	0	-	244	19.67
CodeQL	167	58.08	0	-	0	-	0	-	0	-	167	58.08
Rust	118	55.09	0	-	36	47.22	0	-	2	0.00	156	52.56
ShaderLab*	123	0.00	0	-	0	-	0	-	0	-	123	0.00
SQL	119	34.45	0	-	3	100.00	0	-	0	-	122	36.07
Total	12,859	25.42	1,711	17.46	1,409	37.03	856	18.51	130	29.00	16,965	23.18

which indicates the prevalence of MLSD and the important role ChatGPT currently has within practitioners.

4.3 RQ3: In which contexts do foreign snippets emerge in conversations between ChatGPT and practitioners?

Table 2 presents the contexts in which foreign snippets are suggested by ChatGPT according to our qualitative investigation. For each context, the table provides a short description, the number of conversations categorized into the context and an example of a conversation. In total, 9 distinct contexts were identified by the open card sorting participants. As previously mentioned, each conversation was categorized in up to three contexts. Based on our results, the average number of contexts in which a conversation was categorized is 1.76. This indicates that most of the MLSD-related conversations practitioners have with ChatGPT are non-trivial.

Within the sample of 300 conversations, the **Coding Support** context was identified in 212 (70%), configuring the most identified context. This result indicates that most of the suggested foreign snippets were triggered by coding issues in a language the developer may not be fully proficient (since it is not the primary language in the repository). The selected example for this context showcases a conversation where a developer of a project primarily written in F# looked for support regarding a C# snippet. According to the developer in the original pull request, the C# solution was more adequate for the situation when compared to the F# 'overkill'.

Another popular context is **Configuration Support**, with 98 total conversations. This context represents situations where developers may be proficient in a certain language but are not quite as knowledgeable regarding the coding environment. For instance, as one can see in the table, the developer of a web app needed assistance regarding bash commands to update the project's Node dependencies. Due to space constraints, we are not able to discuss all contexts into details. Nevertheless, we believe the descriptions and examples provided in the table (alongside the full results in our replication package [7]) showcase the wide range of contexts in which foreign snippets are suggested by ChatGPT.

As an answer to RQ3, we identified a total of 9 contexts in which ChatGPT suggests foreign snippets to developers. Out of these, **Coding Support**, **Code Generation** and **Configuration Support** were the most popular. This indicates that foreign snippets suggestions are more often triggered by conversations where developers need assistance to fix and/or generate code in a foreign language.

5 DISCUSSION

Our findings on RQ1 indicate a substantial appearance of foreign snippets among developers' conversations, with over 75% of the code suggestions provided by ChatGPT being in languages different than the repositories' primary language. This trend suggests that developers are not only seeking assistance with many languages but are also exploring and/or integrating solutions across these multiple languages. Such observation is corroborated by our qualitative investigation of the contexts foreign snippets are suggested (RQ3). These findings not only reinforce MLSD as a predominant practice in software development [12, 15, 21] but also indicate how LLMs will likely be assisting this practice in the future.

The high appearance of foreign snippets in languages like Python and JavaScript, evidenced in RQ2, can also relate to the high popularity among developers and the interoperability of these languages [8]. Also, the interaction patterns between languages reveal developers' strategic choices in selecting complementary languages that maximize their productivity and the quality of their solutions.

The large occurrence of contexts such as **Coding Support** and **Code Generation** suggests that developers are actively seeking out LLM's assistance for complex development tasks as well as for routine code generation. This indicates a behavior where developers are not just passively receiving ChatGPT suggestions but are actively engaging with the tool to support their work across various stages of the software development life cycle.

The results of our RQs point out the fact the developers may not seek as much assistance in languages they are already proficient.

Table 2: Contexts in which foreign snippets were suggested, according to the open card sorting procedure. For each context, we provide a short description, the number of conversations and an example of a conversation categorized into the context.

Context	Description	Conversations	Example
Coding Support	Coding-related problems, usually debugging and questions related to the usage of a programming language concepts and libraries.	212	https://chat.openai.com/share/2a6f10f0-d45d-4e71-ac57-584570baeda8
Code Generation	ChatGPT is asked to generate code or scripts. Usually, the developer asks for a code snippet or an entire feature from scratch.	172	https://chat.openai.com/share/5221714a-7251-4a84-b304-5fd4f72d5fb9
Configuration Support	The developer commonly asks ChatGPT for help with the project's setup and configuration.	98	https://chat.openai.com/share/ba3847d4-8a7a-4f5e-b798-8a2584a6ccf7
Documentation Writing	The developer commonly asks ChatGPT to enhance and/or write documentation regarding a code snippet or feature.	32	https://chat.openai.com/share/8526b242-9920-43f2-b199-0df1700ffc3a
Code Refactoring	Scenarios where ChatGPT was asked to refactor a prompted code to improve its quality.	31	https://chat.openai.com/share/9af123e7-5522-4d34-aa5f-dd0b891998a2
Conceptual Questions	For these conversations, developers usually ask about programming concepts, system engineering concepts and software design.	25	https://chat.openai.com/share/2770bf7b-9927-4fd5-a675-e34dce7cefc7
Text Manipulation	Use of ChatGPT to generate, gather, or format text. This is usually related to data engineering.	19	https://chat.openai.com/share/d3e45c4c-af65-4bc1-b8b9-d29e907539fe
Testing Support	For these conversations, the developer's main concern was the implementation, evaluation, and general problems related to testing.	18	https://chat.openai.com/share/99d2ebdc-613c-4a84-8281-bfc2fe82bc14
Database Support	Conversations where ChatGPT was asked to provide support in database queries, data modelling, and administration of a DBMS.	9	https://chat.openai.com/share/2af7029e-20e1-46e2-9d98-f9072ede7c63

In contrast, developers seem to often seek support for languages needed in their projects but which they may not be fully capable yet. To some extent, ChatGPT may serve as a 'normalizer' of programming knowledge, assisting developers to reach a similar level of proficiency for languages they are not knowledgeable when compared to languages they are more familiar. Nervertheless, such an observation needs to be properly investigated in future studies.

6 THREATS TO VALIDITY

We discuss the Threats to Validity according to Wohlin et al. [25] guidelines on experimentation.

Concerning **Conclusion validity**, the contexts identified in the open card sorting procedure and their proportions might not faithfully reflect the dataset. To minimize this threat, we drew a stratified and statistically significant sample with 95% confidence. Regarding **Internal validity**, the contexts observed in the conversations are a product of each participant's bias and worldview. We mitigated this threat by selecting participants with different levels of expertise and different backgrounds.

When considering **Construct validity**, changes in the repository's after this study was performed, such as the primary language changing, may affect how these results hold in the future. Additionally, the identification of repositories' languages and snippets' languages are different between GitHub and ChatGPT. Since the methods used to identify the language are not public, we have no means of verifying its accuracy. To mitigate this threat, we performed a manual analysis to identify spelling differences and normalize the data. Since DevGPT dataset lists the primary language simply as the most used language in each repository, we could not verify the impact of other languages that could be equally important, from the developers' perspective, but less used.

Finally, regarding **External validity**, it is possible that our results could not be generalized to all software development processes since our study restricts to analyzing the DevGPT dataset, which only contains conversations that developers were willing to share. Nevertheless, our results align with previous studies on the subject of MLSD, encouraging a broader analysis of the impact of LLM-based tools on software development. To foster this research direction, we provide a full replication package of our study [7].

7 CONCLUSIONS AND FUTURE WORK

In this paper, we set out to investigate data from conversations between software practitioners and ChatGPT to provide insights regarding Multi-Language Software Development (MLSD). By leveraging the DevGPT dataset, we have shown that most (75%) of snippets suggested by ChatGPT are written in a foreign language, i.e., a programming language different than the primary language in the repository where the conversation originated. Additionally, by executing an open card sorting procedure, we have identified 9 distinct contexts in which foreign snippets were suggested. In general, the usage of ChatGPT in MLSD scenarios are mostly related to seeking programming support for other languages and generating code in other languages.

As the next steps of our research, we envision the extension of our methodology, creating a new dataset based on DevGPT but without its limitations, an so, augmenting our capacity to study both Multi Language and Single Language Software Development and compare them side-by-side. In addition, we plan to expand our qualitative analyses to encompass other nuances regarding developers' usage of ChatGPT within MLSD, analysing in depth the used prompts and prompt engineering techniques.

ACKNOWLEDGEMENTS

This work received partial funding from CNPq-Brazil, Universal grant 404406/2023-8, and support from CAPES - Funding Code 001.

REFERENCES

[1] Harold Abelson and Gerald Jay Sussman. 1996. *Structure and interpretation of computer programs*. The MIT Press.

[2] Mouna Abidi, Manel Grichi, and Foutse Khomh. 2019. Behind the scenes: developers' perception of multi-language practices. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*. 72–81.

[3] Mouna Abidi, Manel Grichi, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2019. Code smells for multi-language systems. In *Proceedings of the 24th European conference on pattern languages of programs*. 1–13.

[4] Mouna Abidi and Foutse Khomh. 2020. Towards the definition of patterns and code smells for multi-language systems. In *Proceedings of the European Conference on Pattern Languages of Programs 2020*. 1–13.

[5] Mouna Abidi, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2019. Anti-patterns for multi-language systems. In *Proceedings of the 24th European conference on pattern languages of programs*. 1–14.

[6] Mouna Abidi, Md Saidur Rahman, Moses Openja, and Foutse Khomh. 2021. Are multi-language design smells fault-prone? An empirical study. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 3 (2021), 1–56.

[7] Lucas Aguiar, Matheus Paixao, Rafael Carmo, Matheus Freitas, Eliakim Gama, Antonio Leal, and Edson Soares. 2024. *Replication package for the paper "Multi-language Software Development in the LLM Era: Insights from Practitioners' Conversations with ChatGPT"*. https://doi.org/10.5281/zenodo.13710992

[8] Tegawendé F Bissyandé, Ferdian Thung, David Lo, Lingxiao Jiang, and Laurent Réveillere. 2013. Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *2013 IEEE 37th annual computer software and applications conference*. IEEE, 303–312.

[9] Daniel P Delorey, Charles D Knutson, and Christophe Giraud-Carrier. 2007. Programming language trends in open source development: An evaluation using data from all production phase sourceforge projects. In *Second International Workshop on Public Data about Software Development (WoPDaSD'07)*.

[10] Manel Grichi, Mouna Abidi, Fehmi Jaafar, Ellis E Eghan, and Bram Adams. 2020. On the impact of interlanguage dependencies in multilanguage systems empirical case study on java native interface applications (JNI). *IEEE Transactions on Reliability* 70, 1 (2020), 428–440.

[11] Kevin Jesse, Toufique Ahmed, Premkumar T Devanbu, and Emily Morgan. 2023. Large Language Models and Simple, Stupid Bugs. *arXiv preprint arXiv:2303.11455* (2023).

[12] T Capers Jones. 2007. *Estimating software costs*. McGraw-Hill, Inc.

[13] Pavneet Singh Kochhar, Dinusha Wijedasa, and David Lo. 2016. A large scale study of multiple programming languages and code quality. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. IEEE, 563–573.

[14] Wen Li, Na Meng, Li Li, and Haipeng Cai. 2021. Understanding language selection in multi-language software projects on github. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 256–257.

[15] Panos Linos, Whitney Lucas, Sig Myers, and Ezekiel Maier. 2007. A metrics tool for multi-language software. In *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications*. Citeseer, 324–329.

[16] Fang Liu, Ge Li, Zhiyi Fu, Shuai Lu, Yiyang Hao, and Zhi Jin. 2022. Learning to recommend method names with global context. In *Proceedings of the 44th International Conference on Software Engineering*. 1294–1306.

[17] Qinyu Luo, Yining Ye, Shihao Liang, Zhong Zhang, Yujia Qin, Yaxi Lu, Yesai Wu, Xin Cong, Yankai Lin, Yingli Zhang, et al. 2024. RepoAgent: An LLM-Powered Open-Source Framework for Repository-level Code Documentation Generation. *arXiv preprint arXiv:2402.16667* (2024).

[18] Philip Mayer and Alexander Bauer. 2015. An empirical analysis of the utilization of multiple programming languages in open source projects. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. 1–10.

[19] Philip Mayer, Michael Kirsch, and Minh Anh Le. 2017. On multi-language software development, cross-language links and accompanying tools: a survey of professional software developers. *Journal of Software Engineering Research and Development* 5 (2017), 1–33.

[20] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help With Code Understanding. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 881–881.

[21] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. 2014. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*. 155–165.

[22] Donna Spencer. 2009. *Card sorting: Designing usable categories*. Rosenfeld Media.

[23] Federico Tomassetti and Marco Torchiano. 2014. An empirical assessment of polyglot-ism in github. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. 1–4.

[24] Michel Wermelinger. 2023. Using GitHub Copilot to solve simple programming problems. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 172–178.

[25] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[26] Tao Xiao, Christoph Treude, Hideaki Hata, and Kenichi Matsumoto. 2024. DevGPT: Studying Developer-ChatGPT Conversations. In *Proceedings of the International Conference on Mining Software Repositories (MSR 2024)*.

[27] Haoran Yang, Weile Lian, Shaowei Wang, and Haipeng Cai. 2023. Demystifying Issues, Challenges, and Solutions for Multilingual Software Development. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1840–1852.