DevBots can co-design APIs

Vinicius Soares Silva Marques
Department of Computer Science
University of Brasília (UnB)
Brasília, Brazil
marques.vinicius@gmail.com

***Abstract*—DevBots are automated tools that perform various tasks in order to support software development. They are a growing trend and have been used in repositories to automate repetitive tasks, as code generators, and as collaborators in eliciting requirements and defining architectures. In this study, we analyzed 24 articles to investigate the state of the art of using DevBots in software development, trying to understand their characteristics, identify use cases, learn the relationship between DevBots and conversational software development, and discuss how prompt engineering can enable collaboration between human developers and bots. Additionally, we identified a gap to address by applying prompt engineering to collaborative API design between human designers and DevBots and proposed an experiment to assess what approach, between using Retrieval Augmented Generation or not, is more suitable. Our conclusion is that DevBots can collaborate with human API designers, but the two approaches have advantages and disadvantages.**

***Index Terms*—DevBots, Conversational Software Development, API Design, Artificial Intelligence, Large Language Models, Generative Pre-Trained Transformers, Retrieval Augmented Generation**

I. Introduction

The use of bots to support software development activities has been a growing trend in recent years. Once restricted to automating repetitive tasks, such as commits to code repositories and opening and monitoring issues [1], [2], today they are also used for code review [3], [4], creating test scripts [5], [6], and monitoring events [3]. With the advance of Artificial Intelligence (AI), and especially with the popularization of Large Language Models (LLMs), they were also used for data extraction and analysis [3], [4], task recommendations [3], [7] and collaborative development through code generation and requirements elicitation [8].

Its use in repositories such as GitHub is widespread. Whether it's to automatically classify, comment on, monitor and close issues [1], [2], [9], [4], or to automate commits [10] [2], its presence is noted in many projects. Its use has evolved to Continuous Integration / Continuous Delivery (CI/CD) tracks [1], [3], [4], with commits, dependency satisfaction and automated deploys, as well as event detection and monitoring [3].

It is also possible to recognize the presence of DevBots in code-related activities. Activities such as code review [1], [3], [4], identifying and correcting bugs [3], [4] or violations of coding standards [4], and static analysis [3], [4] benefit from the use of bots in terms of productivity [4], [11], [5], [12]. With the popularization of LLMs, more complex activities are now being carried out by or in collaboration with DevBots. Extracting, analyzing and sharing data is one of them. In addition, bots can recommend tasks and improvements [3], [4]. More recently, conversational software development has allowed for the elicitation of requirements, collaborative design of architectures [8] and the generation of code, specifications and documentation [5], [13], bringing numerous opportunities for future work.

This study selected and analyzed 24 articles that made it possible to investigate the state of the art of using DevBots in software development. Their main characteristics and use cases were identified. It was also possible to reveal the relationship between DevBots and conversational software development. In addition, the study sought to discuss how prompt engineering can enable collaboration between human developers and bots. Finally, we created and conducted an experiment in which we created API specifications with the aid of an LLM and assessed whether a Retrieval Augmented Generation approach helps in the process.

II. State of the Art

A. Methodology

A Systematic Literature Review (SLR) was carried out following the guidelines of Kitchenham et al. [14] to investigate the use of bots in software development. To do this, research questions were created, a search string was built, databases were selected, selection criteria were defined, a quality assessment was carried out and finally the data was extracted and analyzed.

B. Research Questions

In this work, the aim is to reveal the state of the art in the use of bots in software development, to enable the identification of possible gaps for future studies. The research questions are as follows (Table I):

RQ	Research Questions
RQ.1	What are the most common features of DevBots?
RQ.2	What are some use cases of DevBots?
RQ.3	What is the relationship between DevBots and conversational software development?
RQ.4	How does prompt engineering enable the use of DevBots in software design and development?

TABLE I
List of Research Questions

C. Search String

As the two main subjects to be investigated in this work cover DevBots and conversational software development, we used these keywords to construct the search string:

```
"conversational software development" OR
    devbots
```

D. Databases

The databases selected were ACM Digital Library, IEEE Xplore, Scopus, Web of Science and, to ensure that the most recent results were included, arXiv. To augment the search, Google Scholar was also included, with the following search string:

```
"AI-assisted software development"
```

E. Selection Criteria

Inclusion and exclusion criteria were defined. The inclusion criteria concern the topics covered by the study, the period and the type of articles accepted in the context of this study. The inclusion criteria are listed below:

1) Papers which describe a use case or the development of a DevBot, or literature reviews on the subjects.
2) Published works (including preprints) from 2020 onwards (year of publication of GTP-3 [15]).

The exclusion criteria aim to remove articles that do not meet the inclusion criteria and do not contribute to the objectives of this study. The exclusion criteria are shown below:

1) Duplicates between the databases.
2) Use of search terms in contexts other than the context of this work, or unrelated to the main subjects of this study (DevBots and conversational software development).
3) Compendia of works (as opposed to unique works), PhD or Masters thesis, industry papers and abstract only works.
4) Papers not written in English.

F. Quality Assessment

After applying the inclusion criteria, titles and abstracts had to be analyzed in order to apply the exclusion criteria and ensure that the papers selected adhered to the objectives of this study. Papers in which the search terms did not appear in the title or abstract were discarded, as were those in which the terms appeared in contexts other than those proposed in this article. Results that referred to compendia or collections of works were also discarded.

G. Conducting

Initially, the searches returned a total of 79 results based on the search strings and time interval defined. We then applied the criterion of excluding duplicates, which reduced this number to 60, with only 5 not appearing in either Google Scholar or other databases (4 from ACM DL and 1 from IEEE Xplore). 11 articles not written in English and 13 papers in an inappropriate format (collections of papers, doctoral or master's theses, industry articles and abstracts) were excluded. The remaining papers were screened for titles and abstracts and finally 24 articles were selected for data extraction and analysis (Figure 1).

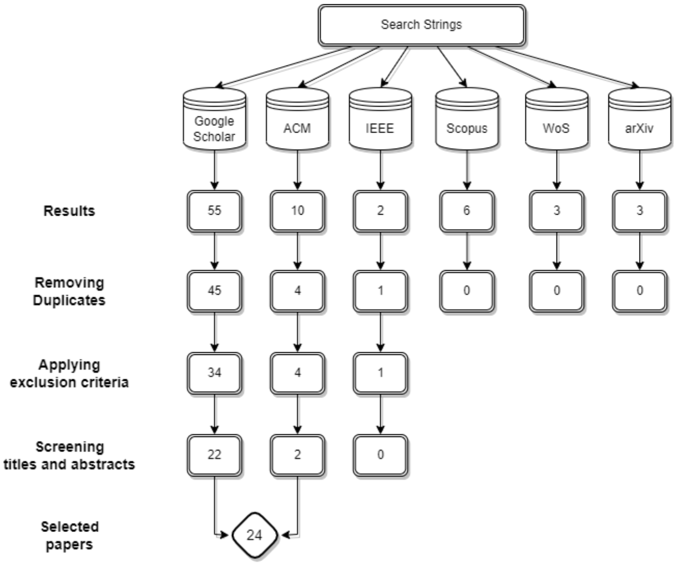

Fig. 1. Selection of papers

H. Data Extraction

The data was extracted considering the objectives of this study and the research questions. In this way, each paper was analyzed to extract the most common characteristics of DevBots (RQ.1); the most common use cases for DevBots (RQ.2); the relationships between DevBots and conversational software development (RQ.3); and insights into how prompt engineering can enable the use of bots in software development (RQ.4).

III. Results of the SLR

A. Related Works

Necula [16] gave an insight into the impact of AI on the software engineering profession, highlighting the new challenges posed by the growing adoption of AI in the field of Software Engineering and the skills that will be demanded of professionals.

Wu et al. [17], Melo [7], Wyrich et al. [18], Wessel et al. [19], Moguel-Sánchez et al. [4], Golzadeh et al. [9] and Bi et al. [1] investigated the use of DevBots on social development platforms such as GitHub.

Tony et al. [20] and Savary-Leblanc et al. [21] sought to understand how developers relate to conversational software development supported by DevBots. Pothukuchi et al. [22], Parashar et al. [13] and Kulkarni et al. [5] discussed how to use AI throughout the software development lifecycle.

Erlenhov et al. [11] worked on defining the DevBot concept, usage scenarios and challenges.

B. DevBots Features (RQ.1)

To answer RQ.1, the papers were analyzed for characteristics commonly associated with DevBots, whether perceived or desired. The most cited feature was **the ability to interact with the developer through natural language processing**, noted by Ahmad et al. [8], Erlenhov et al. [11], Kulkarni et al. [5], Nembhard and Carvalho [12], Parashar et al. [13], Pothukuchi et al. [22], Savary-Leblanc et al. [21] and Tony et al. [20]. This was followed by features related to this capability, such as **AI-driven collaboration**, cited by Ahmad et al. [8], Kulkarni et al. [5], Nembhard and Carvalho [12] and Parashar et al. [13]; **intelligent behavior**, noted by Ahmad et al. [8], Erlenhov et al. [11], Kulkarni et al. [5] and Parashar et al. [13]; **the ability to answer questions**, mentioned by Ahmad et al. [8], Nembhard and Carvalho [12], Tony et al. [20] and Melo [7]; and **interaction with the user**, informed by Erlenhov et al. [23], Nembhard and Carvalho [12], Savary-Leblanc et al. [21] and Tony et al. [20]. **Presenting itself as a conversational agent** was noted by Ahmad et al. [8] and Erlenhov et al. [11]. **Creating synergy between human collaborators and bots** was a feature cited by Ahmad et al. [8] and Nembhard and Carvalho [12].

Characteristics linked to **the ability to provide automation and autonomous behavior** were also cited, like **the ability to generate code, specifications and documentation** was cited by Ahmad et al. [8], Kulkarni et al. [5] and Nembhard and Carvalho [12]. **The ability to build knowledge base systems** was a feature cited by Kulkarni et al. [5], Parashar et al. [13] and Tony et al. [20]. **The ability to make recommendations** was noted by Ahmad et al. [8] and Tony et al. [20].

Erlenhov et al. [11] cites the following characteristics of DevBots: **being triggered agents, presenting identity, being scalable and presenting adaptability**. This last characteristic was also cited by Tony et al. [20], as well as **clarity in interactions, reliability, accessibility, readiness and support for integration with the existing development environment**. Uzair and Naz [24] cites **modularity, reusability and maintainability**.

C. Use Cases (RQ.2)

The findings that answer RQ.2 are summarized in Table II. It is important to note that although the automation of repetitive tasks is among the most cited, more complex use cases are presented by several authors.

D. Relationships between DevBots and conversational software development (RQ.3)

The answers to RQ.3 are not simple. What can be inferred from the literature is that the use of DevBots for conversational software development is still in its infancy, although we can already see initiatives in this regard. According to Ahmad et al. [8], Parashar et al. [13], Pothukuchi et al. [22] and Uzair and Naz [24], **the translation of high-level requirements into architectural models and evaluation scenarios is a relationship that can emerge from the use of AI in software development**, taking advantage of conversational strategies. To do this, according to Erlenhov et al. [11], Nembhard and Carvalho [12] and Tony et al. [20], **these technologies need to be able to parse intents**, to infer what the developer wants and present a relevant solution.

E. DevBot Usage Enablement by Prompt Engineering (RQ. 4)

According to Ahmad et al. [8], Pothukuchi et al. [22] and Uzair and Naz [24], **to enable the use of DevBots through Prompt Engineering the bots need to be able to create the initial specification and iterative enhancements from carefully prepared prompts, requiring human supervision for the generation of code, specifications and documentation**. Ahmad et al. [8], Moguel-Sánchez et al. [4], Pothukuchi et al. [22] and Uzair and Naz [24] point out that these scenarios require the **extraction of requirements based on conversations**. Another possible relationship, according to Ahmad et al. [8] and Tony et al. [20], could be in **obtaining testing and debugging scenarios from conversational interactions between humans and bots**.

IV. Discussion of the SLR

Although many authors have applied AI to software development, there is still room for future research. In addition to applications in repetitive task automation [1], [2], [10], [9], [6], some studies have employed DevBots to actually collaborate in more complex work. Nembhard and Carvalho [12] developed a framework to integrate virtual assistants with human developers through verbal conversations to discuss security aspects and produce more secure software.

Ahmad et al. [8] described a case study involving collaboration between a novice software architect and ChatGPT for software architectural analysis, synthesis, and evaluation using architecture-centric software engineering (ACSE) concepts, and concludes that ChatGPT can simulate the role of an architect to support or even lead ACSE under human oversight and support for collaborative architecture.

We are aware of other studies that were not included in our SLR. Blocklove et al. [26] performed a case study on the use of ChatGPT for hardware design and programming. Through a ChatGPT prompt conversation, a design for an 8-bit shift register is generated (along with a test bench) and evaluated and tested for errors, first by means of a simulation tool, then with the help of human feedback at three levels (simple, moderate, and advanced). The findings show that the approach is useful as long as it is used for co-design in collaboration with a human designer, since ChatGPT cannot generate a complete design with only the initial human interaction.

Gupta et al. [27] provided an AI-augmented framework for persona pool creation, software requirements specification, and usability evaluation that can be used in the requirements analysis phase to improve the usability of software under design. This framework proposes ontologies for representing elemental and abstracted knowledge that can be employed to maximize the effectiveness of AI techniques applied to usability evaluation.

Use case	Paper	# of papers
Issue and pull request lifecycle management	Bi et al. [1], Golzadeh et al. [2], Golzadeh et al. [9], Moguel-Sánchez et al. [4], Erlenhov et al. [11], Liao et al. [6], Wyrich et al. [18], Wu et al. [17], Melo [7]	9
Code review	Bi et al. [1], Copche et al. [3], Moguel-Sánchez et al. [4], Erlenhov et al. [11], Liao et al. [6], Nembhard and Carvalho [12], Wessel et al. [19], Uzair and Naz [24], Wu et al. [17]	9
Testing	Copche et al. [3], Erlenhov et al. [23], Kulkarni et al. [5], Liao et al. [6], Parashar et al. [13], Pothukuchi et al. [22], Wessel et al. [19], Necula [16], Wu et al. [17]	9
Bug fixing and code debugging	Copche et al. [3], Moguel-Sánchez et al. [4], Erlenhov et al. [11], Nembhard and Carvalho [12], Pothukuchi et al. [22], Savary-Leblanc et al. [21**?**] Tony et al. [20], Wessel et al. [19], Wyrich et al. [18]	9
CI/CD	Bi et al. [1], Copche et al. [3], Moguel-Sánchez et al. [4], Erlenhov et al. [11], Pothukuchi et al. [22], Wu et al. [17]	6
Code analysis and verification	Copche et al. [3], Moguel-Sánchez et al. [4], Erlenhov et al. [11], Nembhard and Carvalho [12], Savary-Leblanc et al. [21], Tony et al. [20]	6
Extracting and analysing data	Copche et al. [3], Moguel-Sánchez et al. [4], Erlenhov et al. [11], Parashar et al. [13], Wessel et al. [19]	5
Dependency management	Erlenhov et al. [25], Liao et al. [6], Uzair and Naz [24], Wyrich et al. [18], Wu et al. [17]	5
Coding and implementing software	Kulkarni et al. [5], Parashar et al. [13], Pothukuchi et al. [22], Savary-Leblanc et al. [21], Uzair and Naz [24]	5
Software requirements specification	Ahmad et al. [8], Kulkarni et al. [5], Parashar et al. [13], Pothukuchi et al. [22], Uzair and Naz [24]	5
Automatic issue commenting	Bi et al. [1], Golzadeh et al. [2], Golzadeh et al. [9], Moguel-Sánchez et al. [4]	4
Task recommendation	Copche et al. [3], Moguel-Sánchez et al. [4], Liao et al. [6], Melo [7]	4
Software design	Ahmad et al. [8], Kulkarni et al. [5], Parashar et al. [13], Pothukuchi et al. [22]	4
Bug detection	Moguel-Sánchez et al. [4], Pothukuchi et al. [22], Wessel et al. [19], Uzair and Naz [24]	4
Code refactoring	Liao et al. [6], Savary-Leblanc et al. [21], Wessel et al. [19], Wyrich et al. [18]	4
Detecting and mitigating vulnerabilities	Nembhard and Carvalho [12], Parashar et al. [13], Pothukuchi et al. [22], Tony et al. [20]	4
Detecting and monitoring events	Copche et al. [3], Liao et al. [6], Wu et al. [17]	3
Connecting with stakeholders	Copche et al. [3], Parashar et al. [13], Pothukuchi et al. [22]	3
Providing feedback	Copche et al. [3], Moguel-Sánchez et al. [4], Wessel et al. [19]	3
Software engineering teaching	Kulkarni et al. [5], Wessel et al. [19], Necula [16]	3
Prototyping	Parashar et al. [13], Pothukuchi et al. [22], Savary-Leblanc et al. [21]	3
Sharing information	Copche et al. [3], Moguel-Sánchez et al. [4]	2
Commiting to repositories	Golzadeh et al. [2], Dey et al. [10]	2
Violation of coding standards detection	Moguel-Sánchez et al. [4], Wu et al. [17]	2
24/7 task handling	Erlenhov et al. [11], Wessel et al. [19]	2
Integrating newcomers into the team	Liao et al. [6], Wessel et al. [19]	2
Commenting in social media	Copche et al. [3]	1
Build error identification	Moguel-Sánchez et al. [4]	1

TABLE II

DevBot Use Cases

However, there are still gaps in the use of AI in collaboration with humans. Our study aimed to use conversational design to perform API design. To achieve this, we proposed an experiment in which GPT-3.5 and a human API designer create the collaborative design of an API and its OpenAPI Specification (OAS) through a single prompt using Prompt Engineering (PE) [28] and Prompt Optimization (PO) [29] techniques, and evaluate whether Retrieval Augmented Generation (RAG) [30] leads to better results.

V. Experiment design

The experiment consisted of a series of steps executed in Microsoft Azure Machine Learning Studio. First, we created a prompt to request for the creation of an OAS. This prompt was executed in two pipelines: one containing an RAG structure using the text-embedding-ada-002 model together with a query to the gpt-35-turbo-0613 model and the other only with a direct query to the gpt-35-turbo-0613 model. The prompts for the two pipelines were crafted slightly differently, with the only difference being that the one dedicated to the pipeline with RAG contained two questions to reinforce the discovery, with the help of the embedded documents, of the server URL for the three environments (development, homologation, and production) and the OAuth scopes within the security schema. The pipeline with RAG was fed (through embedding and vector indexing) with PDF files produced from the OAS publicly available on Banco do Brasil's developer portal [1], so that it would be possible to assess the impact of the presence of OAS models considered correct as an input for the generation of OAS by LLM.

This was performed by alternately triggering the two pipelines 10 times each. The response went through a cleaning process that consisted of replacing the characters `\n` and `\"`

[1] https://www.bb.com.br/site/developers/solucoes-api-bb/

with a line break and a double quotation mark, respectively, and removing any text generated by the LLM in addition to the JSON containing the OAS. The JSON was then copied and pasted into Smartbear's Swagger Editor [2] to check the parameters for correctness. The replication package is available at GitHub [3].

A. Correctness assessment

The following correctness parameters were selected:

1) **No syntax errors:** OAS free of errors, in which case the attempt received a score of 0.2 in this regard, otherwise it scored 0.00;
2) **Renders correctly:** OAS presented errors that prevented it from being rendered, in which case the attempt received a score of -0.1. If the error was just a missing character that, when inserted, allowed rendering, or if the errors did not prevent rendering, it received a score of 0.00 in this regard;
3) **Paths according to the REST standard [31]:** in which case the attempt received a score of 0.2 in this regard, otherwise it scored 0.00;
4) **Methods according to the REST standard**: in which case the attempt received a score of 0.15 in this regard, otherwise it scored 0.00;
5) **Functional security scheme:** OAS presented a correct security scheme with two OAuth scopes (one for queries and one for other requests) and the scopes were correctly linked, in which case the attempt received a score of 0.2 in this regard, otherwise it scored 0.00;
6) **Examples of requests included:** in which case the attempt received a score of 0.05 in this regard, otherwise it scored 0.00;
7) **Examples of successful responses included:** if both endpoints had successful response examples (with status code 200 for GET and 201 for POST), the attempt received a score of 0.05, but if only one of them had a response example, the attempt received a score of 0.03, otherwise it scored 0.00;
8) **Error status codes included:** at least one status code besides the success code was included, in which case the attempt received a score of 0.05 in this regard, otherwise it scored 0.00;
9) **One server for each environment:** a server URL was designated for each of the three environments (development, homologation and production), in which case the attempt received a score of 0.1 in this regard, otherwise it scored 0.00.

These parameters were scored such that the attempt received a final score of 1 if all the requirements were met, and 0 if none were met. All OAS provided as benchmarks to the RAG pipeline scored 1 according to these parameters.

VI. Results and discussion of the experiment

The scores obtained for each attempt are presented in Table III. Greater consistency can be observed in the OASs generated by the RAG pipeline, although this does not necessarily translate into a lack of syntax errors. None of the RAG pipeline attempts scored above 0.80, although the pipeline without RAG obtained a score of 0.88 and surpassed the average of the other pipeline in three attempts.

The most common errors in OAS generation were syntax errors (60% of the attempts without RAG and 100% of the attempts with RAG), the absence of a functional security scheme (50% of the attempts without RAG, while the pipeline with RAG was correct in all attempts), the creation of paths outside the REST standard (70% of attempts by the pipeline without RAG and 30% of the attempts with RAG), and the lack of one or more servers (40% of attempts by the pipeline without RAG, whereas the pipeline with RAG generated this error only once). The absence of response examples for at least the success status code (200 or 201) was observed in both pipelines (80% of the attempts by the pipeline without RAG and 100% of the attempts by the RAG pipeline). Both pipelines correctly chose the methods for each endpoint in every attempt, although this may be due to the fact that only one query (GET) and one resource creation (POST) were requested, without the possibility of using the other available methods (PUT, PATCH, DELETE, OPTIONS, HEAD), which may have made the choice easier. In only one case (in the pipeline without RAG) was an OAS generated with an error that prevented rendering. This error consisted of the omission of a ':' character between the name of a key and its value. By including this character, the problem was solved, and the OAS was rendered correctly. In two attempts, the pipeline without RAG correctly generated the OAuth scopes (attempts 5 and 9), although it made mistakes in linking the scopes to the endpoints and thus the attempt was considered wrong. In one case, the pipeline without RAG, although it generated more than one status code, was limited to including only 200 and 400 (attempt 5, and in the others, it generated at least three different status codes). In three attempts (2, 7 and 9), the pipeline without RAG introduced errors beyond those related to the security scheme, and in one (attempt 7), these were the only errors made (in the others, the security scheme was not generated correctly either). As for the RAG pipeline, all attempts had syntax errors, none of them related to the security scheme.

It can be seen that the results of the RAG pipeline were much more consistent with an overall variance of only 0.008 compared to 0.035 for the pipeline without RAG, but this did not translate into a better overall quality for OAS, as there were plenty of syntax errors in all attempts with RAG. However, the average score for all attempts was significantly better than that for the pipeline without RAG. In contrast, three attempts in the pipeline without RAG outperformed the average of the attempts in the pipeline with RAG, suggesting that consistency is the only real advantage of using RAG to

[2] https://editor.swagger.io/

[3] https://github.com/marques-vinicius/OAGen

	Without RAG										With RAG									
#	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
1	0.00	0.00	0.20	0.00	0.20	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.20	0.00	0.20	0.00	0.00	0.00	0.20	0.00	0.20	0.00	0.20	0.20	0.20	0.00	0.00	0.20	0.20	0.20
4	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
5	0.20	0.00	0.20	0.00	0.00	0.20	0.20	0.20	0.00	0.00	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
6	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
7	0.05	0.03	0.03	0.03	0.03	0.03	0.05	0.00	0.00	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
8	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
9	0.10	0.10	0.00	0.10	0.00	0.10	0.10	0.10	0.00	0.00	0.10	0.10	0.10	0.00	0.10	0.10	0.10	0.10	0.10	0.10
Σ	0.60	0.38	0.88	0.38	0.68	0.78	0.60	0.75	0.45	0.28	0.78	0.58	0.78	0.68	0.78	0.58	0.58	0.78	0.78	0.78
$\overline{\Sigma}$	0.578										0.71									

TABLE III

This table shows the correctness parameters presented in Subsection V-A, represented by the numbers 1 to 9 below the # symbol; the attempts without and with RAG, represented by the numbers 1 to 10 to the right of the # symbol; the scores for each attempt, represented by the scores below each atttempt; and the average of the sum of the scores for all attempts ($\overline{\Sigma}$), present in the last two cells of the last row of the table.

generate OAS, but further studies may be performed with a much larger number of attempts for validation. On the other hand, in the third attempt, the pipeline without RAG obtained a score very close to the maximum (0.88) and above all attempts with RAG, as well as another two attempts with a score higher than the average of the other pipeline. In short, to determine which approach produces the best results with a greater degree of assertiveness, it would be necessary to carry out a much larger number of trials. Therefore, we consider the results to be inconclusive regarding the impact of RAG on LLM performance in generating OAS.

VII. Conclusions

DevBots are widely used in various activities related to software development; however, to the best of our knowledge, there are no records of their use in the collaborative design of APIs with humans. To address this gap, we created and conducted an experiment using an LLM, GPT-3.5, to create the OAS of an API with and without the help of an RAG approach. Our findings suggest that it is possible to use DevBots in collaboration with a human API designer, but the results are inconclusive regarding the use of RAG for this purpose. Future studies could evaluate this approach better by repeating the experiment with significantly more trials.

References

[1] Fenglin Bi, Zhiwei Zhu, Wei Wang, Xiaoya Xia, Hassan Ali Khan, and Peng Pu. Bothawk: An approach for bots detection in open source software projects. *arXiv preprint arXiv:2307.13386*, 2023.

[2] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. Evaluating a bot detection model on git commit messages. *arXiv preprint arXiv:2103.11779*, 2021.

[3] Rubens Copche, Yohan Duarte Pessanha, Vinicius Durelli, Marcelo Medeiros Eler, and Andre Takeshi Endo. Can a chatbot support exploratory software testing? preliminary results. *arXiv preprint arXiv:2307.05807*, 2023.

[4] Ricardo Moguel-Sánchez, César Sergio Martínez-Palacios, Jorge Octavio Ocharán-Hernández, Xavier Limón, and Ángel J. Sánchez-García. Bots and their uses in software development: A systematic mapping study. In *2022 10th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 140–149, Oct 2022. doi: 10.1109/CONISOFT55708.2022.00027.

[5] Vaishnavi Kulkarni, Anurag Kolhe, and Jay Kulkarni. Intelligent software engineering: The significance of artificial intelligence techniques in enhancing software development lifecycle processes. In Ajith Abraham, Niketa Gandhi, Thomas Hanne, Tzung-Pei Hong, Tatiane Nogueira Rios, and Weiping Ding, editors, *Intelligent Systems Design and Applications*, pages 67–82, Cham, 2022. Springer International Publishing. ISBN 978-3-030-96308-8.

[6] Zhifang Liao, Xuechun Huang, Bolin Zhang, Jinsong Wu, and Yu Cheng. Bdgoa: A bot detection approach for github oauth apps. *Intelligent and Converged Networks*, 2023. doi: 10.23919/ICN.2023.0006. URL https://www.sciopen.com/article/10.23919/ICN.2023.0006.

[7] Glaucia Melo. Designing adaptive developer-chatbot interactions: Context integration, experimental studies, and levels of automation. In *Proceedings of the 45th International Conference on Software Engineering: Companion Proceedings*, ICSE '23, page 235–239. IEEE Press, 2023. ISBN 9798350322637. doi: 10.1109/ICSE-Companion58688.2023.00064. URL https://doi.org/10.1109/ICSE-Companion58688.2023.00064.

[8] Aakash Ahmad, Muhammad Waseem, Peng Liang, Mahdi Fahmideh, Mst Shamima Aktar, and Tommi Mikkonen. Towards human-bot collaborative software architecting with chatgpt. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, EASE '23, page 279–285, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700446. doi: 10.1145/3593434.3593468.

[9] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. A ground-truth dataset and classification model for detecting bots in github issue and pr comments. *Journal of Systems and Software*, 175:110911,

2021.

[10] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. Detecting and characterizing bots that commit code. In *Proceedings of the 17th International Conference on Mining Software Repositories*, MSR '20, page 209–219, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375177. doi: 10.1145/3379597.3387478. URL https://doi.org/10.1145/3379597.3387478.

[11] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. An empirical study of bots in software development: Characteristics and challenges from a practitioner's perspective. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 445–455, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370431. doi: 10.1145/3368089.3409680. URL https://doi.org/10.1145/3368089.3409680.

[12] Fitzroy D. Nembhard and Marco M. Carvalho. Teaming humans with virtual assistants to detect and mitigate vulnerabilities. In Kohei Arai, editor, *Intelligent Computing*, pages 565–576, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-37717-4.

[13] Binayak Parashar, Inderjeet Kaur, Anupama Sharma, Pratima Singh, and Deepti Mishra. *Revolutionary transformations in twentieth century: making AI-assisted software development*, pages 1–18. De Gruyter, Berlin, Boston, 2022. ISBN 9783110709247. doi: doi:10.1515/9783110709247-001. URL https://doi.org/10.1515/9783110709247-001.

[14] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering - a systematic literature review. *Inf. Softw. Technol.*, 51(1): 7–15, jan 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.09.009. URL https://doi.org/10.1016/j.infsof.2008.09.009.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[16] Sabina-Cristiana Necula. Artificial intelligence impact on the labour force–searching for the analytical skills of the future software engineers. *arXiv preprint arXiv:2302.13229*, 2023.

[17] Xiaojun Wu, Anze Gao, Yang Zhang, Tao Wang, and Yi Tang. A preliminary study of bots usage in open source community. In *Proceedings of the 13th Asia-Pacific Symposium on Internetware*, Internetware '22, page 175–180, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450397803. doi: 10.1145/3545258.3545284. URL https://doi.org/10.1145/3545258.3545284.

[18] Marvin Wyrich, Raoul Ghit, Tobias Haller, and Christian Müller. Bots don't mind waiting, do they? comparing the interaction with automatically and manually created pull requests. In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*, pages 6–10. IEEE, 2021.

[19] Mairieli Wessel, Marco A. Gerosa, and Emad Shihab. Software bots in software engineering: Benefits and challenges. In *Proceedings of the 19th International Conference on Mining Software Repositories*, MSR '22, page 724–725, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393034. doi: 10.1145/3524842.3528533. URL https://doi.org/10.1145/3524842.3528533.

[20] Catherine Tony, Mohana Balasubramanian, Nicolás E. Díaz Ferreyra, and Riccardo Scandariato. Conversational devbots for secure programming: An empirical study on skf chatbot. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, EASE '22, page 276–281, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450396134. doi: 10.1145/3530019.3535307. URL https://doi.org/10.1145/3530019.3535307.

[21] Maxime Savary-Leblanc, Lola Burgueño, Jordi Cabot, Xavier Le Pallec, and Sébastien Gérard. Software assistants in software engineering: A systematic mapping study. *Software: Practice and Experience*, 53(3): 856–892, 2023. doi: https://doi.org/10.1002/spe.3170. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3170.

[22] Ameya Shastri Pothukuchi, Lakshmi Vasuda Kota, and Vinay Mallikarjunaradhya. Impact of generative ai on the software development lifecycle (sdlc). *International Journal of Creative Research Thoughts*, 11(8), 2023. URL https://ssrn.com/abstract=4536700.

[23] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Martin Chukaleski, and Samer Daknache. Challenges and guidelines on designing test cases for test bots. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 41–45, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379632. doi: 10.1145/3387940.3391535. URL https://doi.org/10.1145/3387940.3391535.

[24] Waqas Uzair and Sameen Naz. Six-tier architecture for ai-generated software development: A large language models approach. 2023.

[25] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. Dependency management bots in open-source systems—prevalence and

adoption. *PeerJ Computer Science*, 8, 2022. doi: 10.7717/PEERJ-CS.849. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85128266032&doi=10.7717%2fPEERJ-CS.849&partnerID=40&md5=83062faf179c490146738deb92692235. Cited by: 2; All Open Access, Gold Open Access, Green Open Access.

[26] Jason Blocklove, Siddharth Garg, Ramesh Karri, and Hammond Pearce. Chip-chat: Challenges and opportunities in conversational hardware design. *CoRR*, abs/2305.13243, 2023. doi: 10.48550/arXiv.2305.13243. URL https://doi.org/10.48550/arXiv.2305.13243.

[27] Sandeep Gupta, Gregory Epiphaniou, and Carsten Maple. AI-Augmented Usability Evaluation Framework for Software Requirements Specification. 6 2022. doi: 10.36227/techrxiv.20097701.v1.

[28] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55 (9), jan 2023. ISSN 0360-0300. doi: 10.1145/3560815. URL https://doi.org/10.1145/3560815.

[29] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2023. URL https://doi.org/10.48550/arXiv.2309.03409.

[30] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation, 2023. URL https://arxiv.org/abs/2309.01431.

[31] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, Ann Arbor, USA, 2000.