DOI: 10.15514/ISPRAS-2024-36(1)-13

Exploring the Role of Bots in Software Development

R. Moguel-Sánchez, ORCID: 0000-0001-7735-7110 <*mmoguelrick@gmail.com*>
C.S. Martínez-Palacios, ORCID: 0000-0001-9795-1408 <*delmanclen@hotmail.com*>
J.O. Ocharán-Hernández, ORCID: 0000-0002-2598-1445 <*jocharan@uv.mx*>
X. Limón, ORCID: 0000-0003-4654-636X<*hlimon@uv.mx*>
A.J. Sánchez-García, ORCID: 0000-0002-2917-2960 <*angesanchez@uv.mx*>
School of Statistics and Informatics, Universidad Veracruzana,
Xalapa, Veracruz, México.

Abstract. This research examines the state of applied and proposed software bots in software development through a systematic literature review. Spanning from 2003 to 2022 and encompassing 83 primary studies, the study identifies four bot archetypes: chatbots, analysis bots, repair bots, and development bots. The key benefits of utilizing bots include improved software quality, provision of information to developers, and time savings through automation. However, drawbacks such as limited effectiveness and reliance on third-party technologies are also noted. The study highlights the potential of including bots in software development but emphasizes the need for further exploration and research in this area.

Keywords: bots; development bots; software development; thematic synthesis.

For citation: Moguel-Sánchez R., Martínez-Palacios C. S., Ocharán-Hernández J. O., Limón X., Sánchez-García A. J. Exploring the Role of Bots in Software Development. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 1, 2024. pp. 209-224. DOI: 10.15514/ISPRAS-2024-36(1)-13.

Full text: Moguel-Sánchez R., Martínez-Palacios C. S., Ocharán-Hernández J. O., Limón X., Sánchez-García A. J. Bots in Software Development: A Systematic Literature Review and Thematic Analysis. *Programming and Computer Software*, 2023, Vol. 49, No. 8, pp. 712–734. DOI: 10.1134/S0361768823080145.

Исследование роли ботов в разработке программного обеспечения

Р. Могель-Санчес, ORCID: 0000-0001-7735-7110 <*mmoguelrick@gmail.com*>
С.С. Мартинес-Паласиос, ORCID: 0000-0001-9795-1408 <*delmanclen@hotmail.com*>
Х.О. Очаран-Эрнандес, ORCID: 0000-0002-2598-1445 <*jocharan@uv.mx*>
Лимон К., ORCID: 0000-0003-4654-636X <*hlimon@uv.mx*>
Санчес-Гарсия А.Х., ORCID: 0000-0002-2917-2960 <*angesanchez@uv.mx*>

Университет Веракруса, школа статистики и информатики,
Халапа, Веракрус, Мексика.

Аннотация. В работе рассматривается состояние прикладных и перспективных программных ботов в разработке программного обеспечения посредством систематического обзора литературы. Охватывая период с 2003 по 2022 год и 83 первичных исследования, исследование идентифицирует четыре архетипа ботов: боты-собеседники, боты-аналитики, боты-ремонтники и боты-разработчики. Ключевые преимущества использования ботов заключаются в повышении качества программного обеспечения, в предоставлении информации разработчикам и в экономии их времени за счет автоматизации. Отмечаются такие недостатки, как ограниченная эффективность и использование сторонних технологий. Исследование показывает значительный потенциал ботов при их подключении к разработке программного обеспечения и необходимость дальнейших исследований в этой области.

Ключевые слова: программные боты, боты-разработчики; разработка программного обеспечения; тематический синтез.

Для цитирования: Могель-Санчес Р., Мартинес-Паласиос С.С., Очаран-Эрнандес Х.О., Лимон К., Санчес-Гарсия А. Х. Исследование роли ботов в разработке программного обеспечения. Труды ИСП РАН, том 36, вып. 1, 2024 г., стр. 209–224 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(1)–13.

Полный текст: Могель-Санчес Р., Мартинес-Паласиос С.С., Очаран-Эрнандес Х.О., Лимон К., Санчес-Гарсия А. Х. Боты в разработке программного обеспечения: систематический обзор литературы и тематический анализ. Programming and Computer Software, 2023, т. 49, № 8, с. 712–734 (на английском языке). DOI: 10.1134/S0361768823080145.

1. Introduction

Aligned with the current booming in Artificial Intelligence assisted software development, led by popular projects such as GitHub Copilot and ChatGPT, in Software Engineering (SE), different tools and techniques have been applied to each phase of the software development cycle to increase the quality of the final product, while meeting their requirements. Such tools and techniques respond to the increasing complexity found in modern systems and development environments [1]. Technologies to aid software developers have been arising under Artificial Intelligence (AI) techniques, such as Machine Learning [2] and Natural Language Processing [3]. Development Bots are one such emerging technology born outside the scope of software production.

Due to the broad field of applications for bots, there are multiple types of bots. A Bot can vary in complexity based on its intended objective. "The term bot ranges from describing simple scripts that automate a task in the background to complex applications that interact with one or more humans and autonomously adapt to activities performed by humans and other systems, and even to software applications that use artificial intelligence to mimic human behavior and intelligence" [4].

Bots have been proliferating, motivating research to define and classify bots [4-6], including proposals for new bots that push the use of bots as human assistants [7-8]. Despite the increasing emergence of new bots that aid in software development tasks, little research addresses the practical use of bots in SE. To close this gap, we conducted a Literature Review to analyze published research on bots applied in software development activities, highlighting practical benefits and challenges. The results of this research may help software developers make well-informed decisions regarding

adopting bots in their software development process. Our study can also serve as a steppingstone for future research delving deeper into the emerging field of software bots.

This study is an extension, covering more research papers from 2022 and doing an iteration of Forward and Backward Snowballing to get as many papers as possible that help achieve this research's objective.

This paper is organized as follows. Section 2 presents related studies that contrast the need for this study. Section 3 describes the method used in the planning and execution of the literature review. Section 4 deals with the conduction of the research method. The results are presented in Section 5. Section 6 addresses the discussion of the obtained results. Section 7 includes the threats to validity. Finally, Section 8 draws the conclusions derived from the study.

2. Research method

To meet the stated objective, we conducted a Systematic Literature Review (RSL) following the guidelines of Evidence-Based Software Engineering and Systematic Reviews [9].

2.1 Search Process

We used automatic search as a research strategy to identify primary studies to answer RQs. Afterward, we performed a snowballing search to expand the pool of candidate studies. We applied a process based on Creswell's thematic analysis for data synthesis. The research questions are the following:

RQ-1: In which software development activities are bots involved?

RQ-1.1: What is the goal of a bot in the activity it is used in?

RQ-1.2: Which software development activities supported by bots are the most reported in the literature?

RQ-2: What are the benefits of applying a bot in a development activity?

RQ-3: Which problems arise in development activities that use bots?

RQ-4: What are the levels of intelligence of bots used for software development?

2.1.1 Search Strategy

A manual search was performed according to the steps outlined in the Automatic Search Process [9] to identify primary studies in SE. Following the process, a Quasi-Gold Standard was created to evaluate the performance of search strings to select a high-quality search string [10].

Identifying Scientific Databases: The first step was to identify venues from which to select articles through an automatic search. As primary search engines IEEE Xplore and ACM Digital Library were selected since they contain the entire catalog of articles published under the International Conference of Software Engineering (ICSE), particularly its subsidiary conference Bots in Software Engineering (BotSE), which is related to bots in software development. To bolster the rigorousness of the search process, additional databases were included. These databases were Science Direct, Springer Link, and EBSCO Host.

Afterward, a manual search was conducted for studies under ICSE to construct a Quasi-gold Standard [10]. From the manual search, 20 relevant articles were identified. From the candidate studies identified in the manual search, a set of concepts and keywords was extracted to craft search strings to test for automatic pilot searches in IEEE Xplore. After applying the method for search string selection, the selected search string is the following:

("Analysis bot" OR "Chatbot" OR "Conversational bot" OR "Conversational developer assistant" OR "Developer assistant" OR "Repair bot" OR "Automated repair" OR "Review bot" OR "Software bot") AND ("Software Development" OR "Software Project" OR "Open source project" OR "Software Engineering" OR "Repair")

2.2 Selection Process

Afterward, inclusion and exclusion criteria were defined to determine adequate studies for data extraction. We only considered articles published inclusively between 2011 and 2022 for this extraction method. This range of years of publication was chosen to drive the search scope toward recent and current articles regarding bots for software development activities. Additionally, the topic of bots has seen an increase in popularity with the creation of the BotSE conference in 2018, which centers on applying bots in SE. The inclusion and exclusion criteria used to select articles from the automatic search are shown in Table 1.

Table 1. Inclusion and Exclusion criteria for selecting studies on Automatic Search

Inclusion Criteria	Exclusion Criteria
IC-1: The study is published between 2011 and 2022	EC-1: The study is secondary or tertiary.
IC-2: The study is written in English	EC-2: The study is an opinion, presentation, or book chapter.
IC-3: The title and abstract suggest that at least RQ can be answered.	EC-3: The study is a duplicate found on a different database.
IC-4: The full-text answers at least one RQ.	EC-4: The study scores lower than six on the quality evaluation process.

2.3 Data Extraction Process

Publication data related to the demographics of each study were extracted. With this data, a particular study can be quickly identified through its characteristics and group studies, considering its publication year and publisher. For example, by cataloging the data by year, a count of how many studies about development bots were published by year can be elaborated. With this information, it can be determined whether there is a growing interest in the topic of bots in software development. Likewise, by sorting studies by the publisher, conferences with the most papers in academia on the topic of bots for SE can be identified. The Extraction Data fields are related to the RQ to be answered in this research. The type of bot identified in each article is recorded to identify the demographics of the types of bots used in SE. By extracting the development phase and activity in which a bot is used, RQ-1, regarding where a bot is applied in the software development cycle, can be answered. Similarly, the other data fields are related to RQs with Bot Adaptability, Reasoning, and Autonomy associated with different aspects of the level of intelligence observable in bots. The template used to extract the data is shown as part of a Zenodo data set [11] containing artifacts elaborated as part of this research.

2.4 Snowballing Search Process

To expand upon the studies selected from the automatic search, a Snowballing search for SE was executed based on the Guidelines for snowballing in systematic literature studies and replication in software engineering [12]. This search process selects additional papers from the references and citations of studies in an initial batch. This process was applied based on the primary studies collected from the automatic search published between 2011 and 2022. First, an iteration of backward snowballing was performed. Afterward, a single iteration of forward snowballing was executed with the aid of the Google Scholar search engine. The same selection criteria filters for the automatic search were applied to papers obtained through backward and forward snowballing, excluding the IC-1 Inclusion Criteria because we reached some articles back from 2011. Likewise, the same data extraction process was followed in the resulting studies.

2.4.1 Thematic Synthesis

Initially, an analysis was conducted through a narrative synthesis to answer the established RQs. The synthesis is based on the bots identified in 83 primary studies, resulting from the quality

evaluation of data extracted from automatic and snowballing searches. After that, a thematic synthesis was elaborated following the Recommended Steps for Thematic Synthesis in Software Engineering by Cruzes & Dybå [13] to relay the literature review results. As Result, a thematic synthesis of the analyzed data in the form of answers to established RQs can be seen, along with graphics to illustrate our responses.

3. Method conduction

The selection process was executed in five main stages, applying inclusion and exclusion criteria to identify relevant studies. First, an automatic search was conducted. Then, based on the resulting full-text studies, a backward snowballing search was performed. Afterward, a forward snowballing search was applied to the studies obtained from the automatic search. Both referenced studies from backward snowballing and citations from forward snowballing underwent the selection criteria filtering. Lastly, the quality evaluation process was performed. From the automatic search and supplementary snowballing search process, we obtained 83 primary studies. The artifacts produced as part of this research for the method conduction can be seen as a Zenodo data set [11].

4. Results

This section addresses the demographics of the identified Primary Studies (PS) and answers the RQs in order of appearance. For this section, we use the identifier PS-XX to refer to a specific study; the list with all the Primary Studies can be found in the following link *https://drive.google.com/file/d/1kx9DKS6qJ7sGDIaMtZ7DK1LunD83gjjy*.

4.1 Overview of Primary Studies

Fig. 1 shows a trend in the increase of publications of interest, especially between 2016 and 2022, with a notable surge in 2019. This trend can be partially attributed to the BotSE conference, created in 2019, contributing articles detailing the applications of bots in software development activities. It is interesting to note that there were fewer articles in 2020 compared to 2019. This may be attributed to the worldwide disruptions caused by the COVID-19 pandemic, which, to a lesser extent, also affected publications from 2021 and beyond. Nonetheless, Bots in SE achieved high international recognition partly thanks to BotSE.

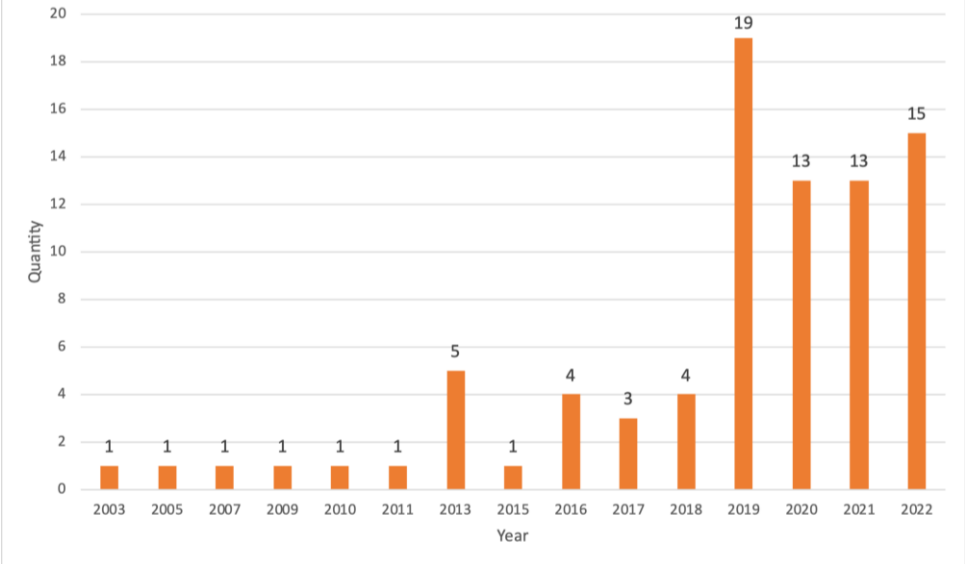

Fig. 1. Distribution of primary studies found by year

Among the different types of articles, 25 come from backward snowballing, and 29 primary studies come from IEEE Xplore Digital Library using the automatic search. The prevalence of the latter is due to the hosting of conferences covered by ICSE, including the BotSE Workshop, one of the leading conferences on this topic. With BotSE, it is evident that the topic of Bots has garnered significant interest at the international level. Prior papers show that the topic existed and only needed an international push to jump-start its popularity. Even so, existing works before 2019, although they only represent 12%, should not be disregarded.

4.2 Answers to Research Questions

As our Research Method section mentioned, we proposed 4 RQs to meet the literature review objectives. In this section, we aim to answer such questions with the results obtained from the 83 primary studies selected.

4.2.1 RQ-1

The software development activities where bots are involved. Fig. 2 presents the specific software development activities found in the primary studies. We can highlight project management as the most common activity involving bots.

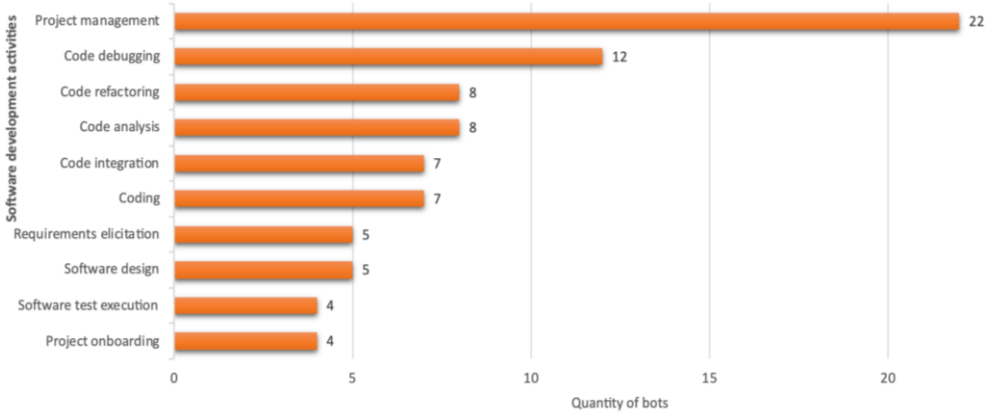

Fig. 2. Number of bots found by development activity

Software development teams spend a significant amount of time discussing solutions through messaging platforms during the development of a project [15]. As project management was found to be the most popular activity for the use of bots, it is clear there exist attempts to find different alternatives to manage software projects. Primary studies covering bots for software management follow. PS-13 showcases a chatbot to support project data analysis and team progress measurement. Similarly, TA Bot in PS-21 is a bot for solving work delays during development, which also provides graphs related to the impact that delays cause in development, recommending actions to mitigate the adverse effects. Some of this type of bots are specific to the GitHub platform, such as Stale Bot in PS-17 for identifying unattended pull requests; JITbot in PS-29 for prioritizing pull requests needing a code review; and DependaBot detailed in PS-37 and PS-40 for updating dependencies in repositories. We found some studies for source code management in GitHub, such as Code Climate Bot in PS-23, which reminds developers about pending fixes, and StyleCI in PS-34, which applies code style preferences. Magalhaes de Lima proposes another bot for code management in PS-33 to update different project branches. Devy Bot in PS-18 is the most ambitious of all management bots, being a conversational development assistant responsible for managing tedious, repetitive tasks with low levels of abstraction, thereby liberating developers to focus on more demanding and complex tasks.

We identified 12 bots related to code debugging. As a representative example, PS-1 by Belskii A. and Itsykson V. M. is a bot that starts by performing analysis, identifying bugs, generating possible solutions, offering patch fixes, and finally notifying the developer of the results.

Among the main activities identified, another one is code analysis, characterized by code reviews and static analyses executed in the software implementation and maintenance stages. The main challenge in code review is the large amount of human effort involved, even for tool-assisted code reviews [16]. This could be due to the excessive complexity of applying coding standards or reviewers prioritizing logical checking [26]. The high number of bots identified shows that this activity has great potential for bot inclusion. Repairnaitor in PS-6 identifies errors in failed builds and proposes solutions autonomously. Other bots that perform code analysis and recommend follow-up actions include Sankie in PS-11, CCBot in PS-25, and Danger Bot in PS-23. For a simple and straightforward bot companion, C-3PR Bot in PS-27 provides details for detected bugs, and Review Bot in PS-24 notifies coding standard violations. For GitHub, Saw-bot in PS-41 invokes SonarQube – a tool for static code inspection, generates a patch and then integrates its solution into a pull request. SapFix in PS-11 runs a code analysis on a pull request and then recommends different DevOps activities to perform afterward.

As a sample from code integration, Sayme Bot, in PS-15, warns of potential direct and indirect conflicts when uploading changes to a GitHub repository.

For Requirements Elicitation, there were five bot proposals. For instance, Dwitam F. and Rusli A. in PS-4 propose a bot capable of conducting interviews with stakeholders in software projects to elaborate user stories. Surana C. et al. in PS-22 propose a chatbot capable of simulating natural conversations with Stakeholders that can extract and classify requirements based on conversations.

For software design, we identified five bots. RAPID bot in PS-31 offers design recommendations based on non-functional requirements, and iContractBot in PS-39 creates models through a step-by-step conversation. Gilson F. and Weyns D. envisioned a more social bot in EP-9. Their bot generates design artifacts based on the chat logs of a development team instead of a single member.

Another notable activity is Onboarding, where new developers are introduced to the project, with four bots identified. MSR Bot in PS-7 is envisioned as an assistant focused on answering questions about a GitHub repository's documentation for project newcomers. The remaining identified bots for Onboarding address questions from development team members, such as APIBot in PS-26 and Smart Advisor in PS-16.

4.2.2 RQ-2

The benefits of applying a bot in a development activity are shown in Fig. 3.

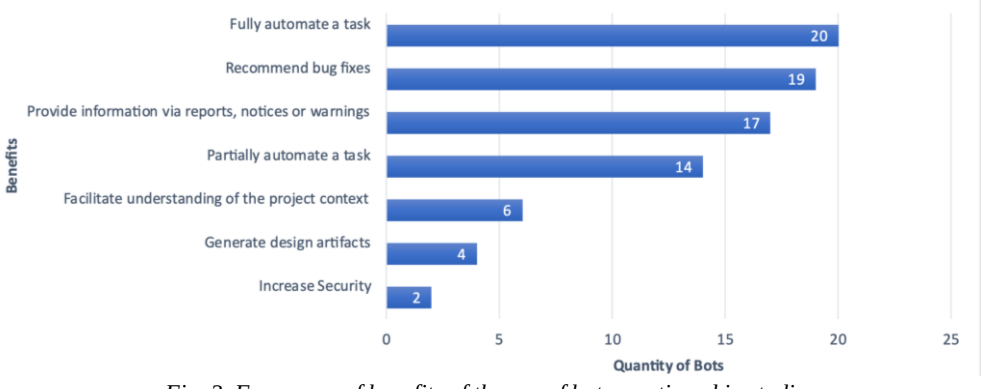

Fig. 3. Frequency of benefits of the use of bots mentioned in studies

Fig. 3 describes the reported benefits of using bots, with task automation being the most frequently observed benefit. The valuable benefit of time-saving bots is that bots help developers save time by

partially or fully automating tasks. We found 20 bots that can completely automate tasks. For example, Devy in PS-8 automates Git and GitHub tasks by creating commits to a repository, assigning code reviews, and managing a repository via voice commands. Refactoring Bot in PS-18 saves time by fully automating code refactoring to resolve code smells identified by the SonarQube static analysis tool; it then submits the changes to the developer for review via pull requests on GitHub. Lastly, Review Bot in PS-24 automates static analyses by generating checks for coding standard violations and common defect patterns and publishing the results of its analysis. In contrast, we identified 14 bots that automate parts of a task. This includes a chatbot by Surana C. et al. in PS-22, Travis CI in PS-23, and Stale Bot in PS-17. These bots partially automate tasks such as user interviews, continuous integration, and pull requests. Another bot, by Magalhaes de Lima in PS-33, updates different branches based on changes pushed to a specific repository branch. These bots provide a time-saving benefit, though to a lesser extent.

Another prevalent benefit of bots was the recommendation of fixing candidates for bugs. By getting possible solutions to a faced problem using a bot, developers can make appropriate decisions regarding the patches generated by the bot instead of spending time analyzing the problem on their own. These bots are predominantly repairing bots that help the developer identify bugs and give possible solutions that save time and improve the quality of the final product. For example, the Repairnator bot in PS-6 generates patches for undetected bugs. When it finds a bug, Repairnator attempts to replicate and repair the flaw, reporting the results to developers. Another example is R-Hero in PS-36, which repairs code with compile errors. Similarly, C-3PR Bot in PS-27 suggests bug fixes identified through static analysis on pull requests from a project repository. ConE in PS-32 also proposes recommendations for conflict resolution in pull requests, evaluating the danger of introducing defects in open pull requests.

It is also reported that 17 bots proactively provide helpful information without requiring a direct request from a developer. For instance, Sayme in PS-15 is a chatbot that provides notifications of potential direct and indirect conflicts when pushing changes to a GitHub repository. Alternatively, a chatbot can provide valuable information on demand, supporting developer decision-making, as is the case for Smart Advisor in PS-16, with the ability to answer developer questions and give appropriate alerts and recommendations during coding and maintenance.

Improvement in understanding the context of a software project is also a benefit found in six cases. In three cases, chatbots are used for training new development team members. The first was API Bot in PS-26, which introduces a developer to a documented API, answering questions about the API. Similarly, Dominic J., Ritter C., and Rodeghero P.'s chatbot in PS-5 answers questions from onboarding members. MSR Bot in PS-7 makes it easy to understand the context of a software repository and the changes made by the team. Lastly, we found two bots that increase security. These were PAutoBotCatcher in PS-35, designed to detect botnets attempting to breach a system, and SEADER in PS-76, designed to detect and generate fixes to misuses of APIs for Java.

4.2.3 RQ-3

Problems that arise in development activities that use bots are shown in Fig. 4.

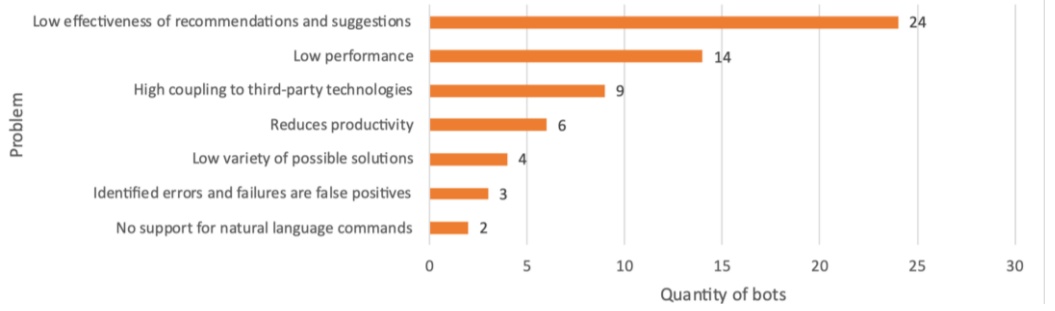

Fig. 4. Frequency of challenges with the use of bots mentioned in studies

Fig. 4 details difficulties that emerge from using bots in software development activities. We can identify two main complications: the low effectiveness of generated solutions and the poor performance of a bot in a designated activity.

From the 24 identified cases of bot ineffectiveness, C-3PR Bot in PS-27 exemplifies a typical scenario where developers reject bot fix suggestions. In this case, C-3PR Bot removed commented-out code that developers wanted to keep. Another instance of low effectiveness is PTracer in PS-19. It had a rejection rate of 32.45% for its generated solutions, as developers noted that the produced patch failed to solve identified bugs. Similarly, the bot by Brown C. and Parnin C. in PS-12 reportedly had ineffective recommendations regarding the installation of a static analysis tool; developers only accepted 3% of the bot recommendations, as the bot does not recognize the developer's working context; when trying to install the tool, source code no longer compiled correctly because it violated the file formats followed by developers. On the other hand, PS-23 reports that Travis CI in PS-23 was not beneficial to developers, as they had difficulties executing tests, preferring to debug manually.

Regarding difficulties due to bots' performance, creating fixes and patches for bugs is the limited variety of fixes. In two sample cases, bots are reported to have a limited number of possible solutions. This is the case of RefBot in PS-20, with a low variety of solutions, as the bot only focuses on recent Pull Requests and not on the current developer context; furthermore, this bot focused only on quality attributes based on the QMOOD Model, which can be counterproductive for developers who do not use this model. Another case lacking recommendation variety is the TA Bot in PS-21, as this chatbot did not have enough types of delays to give practical recommendations. The recommendations of TA Bot did not address the cause of the delays, only the symptoms.

High dependence on third-party technologies is a reported drawback in nine other cases. For example, various studies report that bots require a constant Internet connection, as in the case of the MSR Bot chatbot in PS-7. The Sankie bot from PS-11 also had a high dependency on external technologies. In this case, the bot analyzes coupling in the continuous integration and deployment pipeline, depending on the DevOps services provided by the Microsoft Azure platform. Sankie bot is also prone to respond with recommendations determined to be false negatives.

Regarding problems with false positives, the bots in PS-3 by Padgham L. et al., StaleBot in PS-38, and Task Navigator in PS-46 all reportedly flagged as error actions by developers outside of the original scope of the bot. In the case of the bot by Padgham L. et al., for categorizing warnings, exceptions, and failures during test execution, it was deemed unreliable to trust the bot’s categorization in case few unit tests were executed. StaleBot would flag as obsolete old pull requests that developers would still find relevant, and TaskNavigator would not query questions from developers in the case there were misspellings. However, the problems with TaskNavigator were alleviated by implementing an autocorrect function.

Another rare but identified problem is bots lacking support for specific commands in natural language. This is the case of the Refactoring Bot reported in PS-18, as commands in natural language were not supported. Therefore, the learning curve to use the bot was high since a developer must learn the specific commands of the chatbot. In the case of Devy in PS-8, the bot could not interpret commands from specific phrases; for example, it did not interpret the question "Who has made changes?" as a command to identify who has edited a file. The participants did not consider that the bot can interpret commands more ambiguously, needing more specific commands.

4.2.4 RQ-4

The intelligence characteristics we identify in this work are Adaptability, Reasoning, and Autonomy [7]. The subsections that follow will specify each characteristic.

Capacity for Adaptability. Fig. 5 shows a pie chart detailing whether the bots identified are cable of Adaptability, whether they can recognize changes in their environment, such as different project configurations, or if they change their behavior to adapt functionality, supporting the developer's

needs. To be able to adapt, a bot needs to be aware of the context of its development activities. In other words, the bot refines its actions for the developer's benefit as the bot learns from their mutual interactions. Of the 83 bots extracted from the primary studies, in 65% of the cases, bots can adapt and recognize a context. In contrast, 27% of the studies report that the bot does not have this capacity. The bot's intelligence level is not mentioned in the rest of the cases.

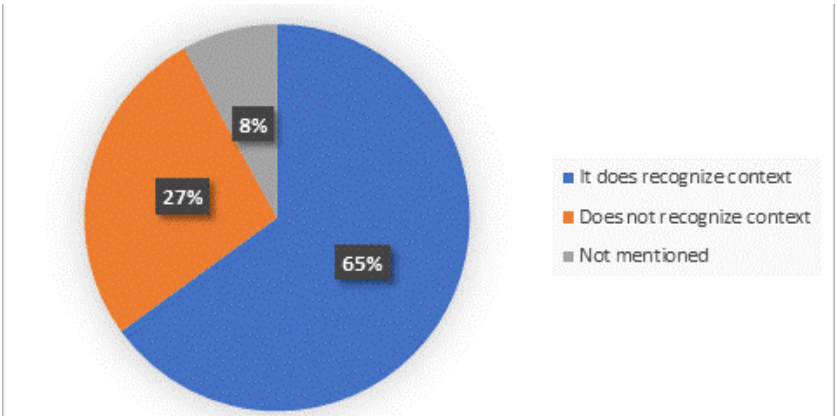

Fig. 5. Percentage of bots found with the capacity for Adaptability

Bot reasoning ability. In this case, reasoning refers to the existence of an internal bot mechanism to parse input data, perform a task to assist the developer, and provide output data to notify developers about the results of actions. Natural Language Processing is the most mentioned technique associated with bot reasoning, particularly for chatbots. Fig. 6 shows a pie chart detailing whether the bots identified are cable of Reasoning. Of the 83 bots identified, in 74% of the cases, it was possible to extract a reasoning mechanism for a bot. Of the remaining cases, in 11% of the cases, the bots do not have this ability, and in 15% of the cases, a reasoning mechanism is not mentioned.

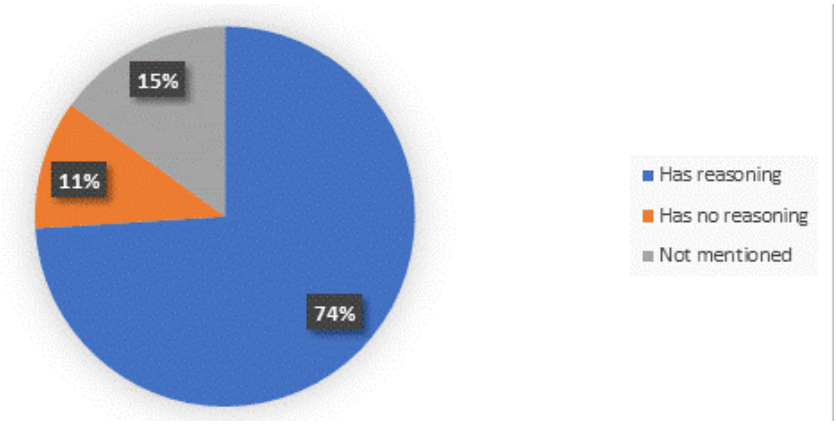

Fig. 6. Proportion of bots found with the capacity for Reasoning

Autonomy capacity. Fig. 7 presents a pie chart detailing whether the bots identified are capable of Autonomy, which is the ability to provide a beneficial service without the need for explicit human instructions. Bots that demonstrate Autonomy have a degree of independence and perform their tasks on their own, contacting the developer via messages and notifications to provide information, warnings, recommendations, or status reports, either periodically on their own or when requested by a human. A case that illustrates high independence is JIT Bot from PS-29. After performing its initial analysis, it keeps track of the repository and its changes without human intervention. Analysis Bots can also be autonomous, like the C-3PR bot in PS-27. On its own, C-3PR suggests fixes for static analysis violations via pull requests, which constructs after running an independent static analysis.

A plurality of bots at 47% lack autonomy, whereas only 42% of bots identified were determined to display it. Notably, in 11% of the group of bots identified autonomy was not mentioned and thus could not be determined.

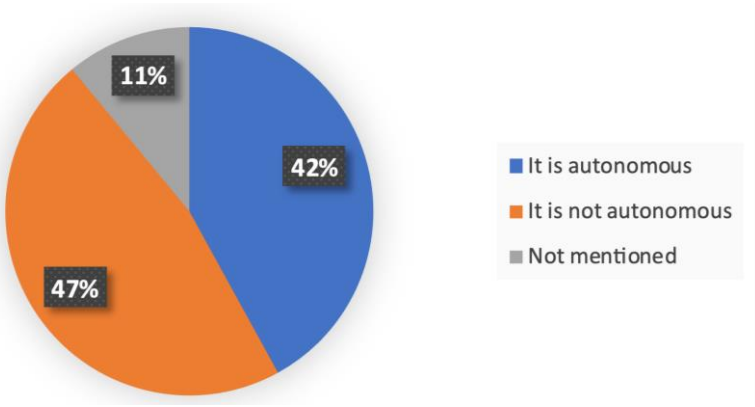

Fig. 7. Proportion of bots found with the capacity for Autonomy

4.3 Thematic Synthesis Results

To produce a thematic map, we used MaxQDA, a tool for data analysis, integrating qualitative methods. With this software, we identified and organized coded segments of data. Fig. 8 shows the higher-order thematic map produced following the thematic synthesis process for software engineering [13].

The diagram is divided into themes derived from codes based on excerpts from the 83 primary studies found in the literature review. Eight higher-order themes, representing different concepts, branch from the main theme about bots for software development. These themes, categorized in clockwise order, are Reasons why developers use bots; Bot users; Benefit from the use of bots; Activities that apply bots; Types of bots, classified by the platform for their use, and Lebeuf et al.'s Taxonomy [9]; Mechanisms for the implementation of bots; Problems encountered with the use of bots; and Areas of improvement proposed by authors. Each lower-order theme is linked to other themes in different categories.

5. Discussion

This section presents a discussion regarding the results associated with each RQ. A significant general observation is that primary studies on bots applied to software development before 2019 are scarce. The surge of research articles afterward can be attributed in part to the creation of the first international workshop on using Bots in software engineering (BotSE), a subsidiary of ICSE. It is apparent from the results that this workshop significantly boosted the research on this topic. It is also interesting to note that following the surge in 2019, in the following years, primary studies decreased slightly. This may be due to the disruptions caused by COVID-19 at a global level from 2020 onwards.

5.1 Software development activities where bots are involved

The most popular activity found is project management, in which traceability of actions and communication to facilitate collaboration between members is crucial. Other aspects include monitoring the development activities performed by a team and maintaining control in online repositories such as GitHub, where source code and documentation can be uploaded. Another frequently cited activity is onboarding, where the team members receive training, either when a new member joins and needs to understand the context of the project or even to learn the workflow of the rest of the team. Notably, bots can also support activities related to source code, such as coding,

static analysis, debugging, and code refactoring; but in this kind of activity, bots also present some issues.

The results are likely to be expanded as more types of bots arise. We believe new bots will appear focusing on other software development activities not seen in the results, such as bots for software validation, generation of test suites, monitoring a Personal Software Process (PSP), or other software development activities. This will significantly increase the usage and opportunities of bots in the context of software development.

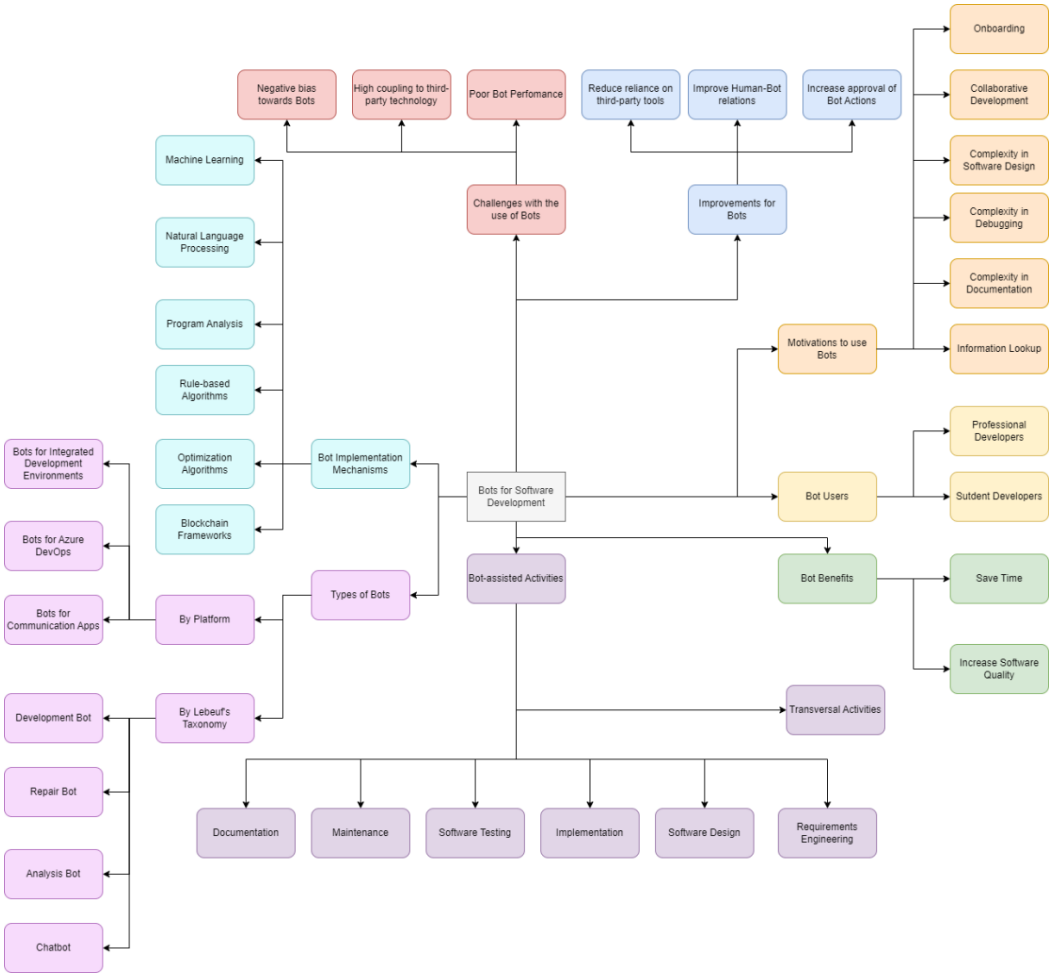

Fig. 8. Thematic map of higher-order themes about bots

5.2 Benefits of applying a bot in a development activity

The most widely reported benefit of using bots is time-saving, as bots can perform tasks or assist developers, decreasing the time to complete different tasks. As mentioned earlier, bots for project management are the most popular, helping to save time, although they come with a learning curve and have inherent drawbacks related to adopting new tools for development. As an example of time-saving, bots for repository management activities, such as organizing pull requests, code reviews, and project builds, have the main benefit of saving time by reducing developer workload and overseeing menial and relatively straightforward tasks.

A limited benefit of bots is that they can provide helpful information through recommendations. This benefit applies to bots that identify bugs and recommend a suitable fix. The caveat is that sometimes the bot's solutions are incorrect. Therefore, humans should always review bot solutions before committing them to a repository.

Another benefit we want to highlight comes from bots for onboarding. These bots help new developers in a team to learn about the project domain and workflow of existing team members so that they can integrate into the team with minimal disruptions and better productivity. We believe that these bots could be adopted to teach SE students about the context of professional software development since they will be the new software developers in a team.

The benefits provided by bots depend on the functions performed by them related to development activities. Generally, the more involved a bot is in an activity, the greater the benefits. However, a developer should weigh the risks that come with the use of bots.

5.3 Problems that arise in bot-aided development activities

We also analyzed challenges encountered by adopting bots for software development to contrast benefits. The most frequent is the poor performance of the bot contributions. The recommendations or warnings bots provide can sometimes be of little to no use to a developer.

There can also be developers misunderstanding regarding the scope and limitations of bots. This, in turn, leads to developer frustration, causing reluctance to use bots.

A different technical limitation is that bots cannot communicate on the same verbal level as humans, although significant advances in chat and voice bots aim to close this gap, as is the case in emerging technologies such as ChatGPT.

We believe that the difficulties that arise from using bots are inherent to the learning curve associated with them. This is especially true for some chatbots, as they need structured commands to perform tasks, requiring developers' memorization of them to be productive. In this sense, we can argue that bots have not yet fully achieved the goal of becoming seamless developer assistants.

5.4 Bots' intelligence level for software development

Regarding the intelligence of bots, we observed that, from the pool of 83 studies, most of the bots display all 3 of the intelligence metrics. These metrics, following [9], are the ability to exhibit Adaptability, Reasoning, and Autonomy. However, in a minority of cases, bots either did not present a particular intelligence or the study did not mention it, resulting in ambiguity, not allowing us to determine if a bot had or did not have a metric for intelligence.

We also note that these intelligence criteria were subject to our interpretation since, in most primary studies, intelligence metrics are not explicitly mentioned. Thus, we derived the appropriate intelligence metric from the bot function, the interactions of the bot with humans, and the bot implementation mechanisms.

6. Conclusions

In this paper, we conducted a literature review on bots in Software Engineering. The study selection consisted of an automatic search, followed by a supplementary Snowballing search and a quality evaluation filter. From this process, we identified 83 primary studies which answered our RQs. The primary objectives of our RQs were to outline SE activities supported by bots and the benefits and challenges that this support entails.

We found in our research that bots are mainly used for project management, to automate tasks with a low level of abstraction, such as tagging pull requests and commits, assigning team members to code reviews, performing static code analyses, and tracking changes in project repositories.

The primary benefit of including bots in software development is time-saving, automating tasks, and expediting activities such as debugging through bug identification and solution recommendation.

Time-saving in these activities means developers have more time for complex development activities related to Requirements Elicitation, Design, and Implementation.

Despite the benefits, challenges such as low bot effectiveness and performance arise. These problems come along with a perceived high learning curve for using bots, making it challenging to use bots to their full extent and understand their limitations. All the mentioned problems diminish time-saving and may detract software developers, but we believe that developers and researchers will overcome such challenges in the future as better and more sophisticated Artificial Intelligence techniques appear.

Our research points out that most bots exhibit, to some extent, adaptability, and autonomy, which may indicate that most bots have a high level of intelligence. Even so, limitations in language processing still prevent bots from being true developer assistants with the capacity to communicate and participate in the development as team member colloquially. However, the mentioned limitations may soon change thanks to breakthrough new technologies like ChatGPT.

As for future work, we envision delving deeper into the mechanisms and Artificial Intelligence principles in which bots for software development rest. This complements the current study, allowing us to associate trends in Artificial Intelligence with Software Engineering advancements to create a better picture for software developers and researchers. We believe software development assisted by Artificial Intelligence will be an ongoing trend in Software Engineering, like sign and overflow detection.

References

[1]. Nagaria B., Hall T. How software developers mitigate their errors when developing code. IEEE Transactions on Software Engineering, 2020.

[2]. Suta P., Lan X., Wu B., Mongkolnam P., Chan J. An overview of machine learning in chatbots. International Journal of Mechanical Engineering and Robotics Research, vol. 9, no. 4, pp. 502–510, 2020.

[3]. Rainey S. K., Brown B., Kirk D. B. Bots, natural language processing, and machine learning. Tax Executive, vol. 69, p. 39, 2017.

[4]. Lebeuf C. R. A taxonomy of software bots: towards a deeper understanding of software bot characteristics. Ph.D. thesis, 2018.

[5]. Lebeuf C., Zagalsky A., Foucault M., Storey M.-A. Defining and classifying software bots: A faceted taxonomy. in 2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE), pp. 1–6, IEEE, 2019.

[6]. Erlenhov L., de Oliveira Neto F. G., Scandariato R., Leitner P. Current and future bots in software development. in 2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE), pp. 7–11, IEEE, 2019.

[7]. Orgeolet L., Foulquier N., Misery L., Redou P., Pers J.-O., Devauchelle-Pensec V., Saraux A. Can artificial intelligence replace manual search for systematic literature? Review on cutaneous manifestations in primary Sjögren's syndrome. British Journal of Rheumatology, vol. 59, no. 4, pp. 811–819, 2020.

[8]. A. Ciupe, S. Meza, and B. Orza, "Systematic assessment of interactive instructional technologies in higher engineering education," in International Conference on Interactive Collaborative Learning, pp. 797–804, Springer, 2020.

[9]. Kitchenham P., Budgen D., Brereton P. (2015). Evidence-based software engineering and systematic reviews. CRC Press, DOI: 10.1201/b19467.

[10]. Zhang H., Babar M. A., Tell P. (2011). Identifying relevant studies in software engineering. Information and Software Technology, 53(6), 625–637. DOI: 10.1016/j.infsof.2010.12.010.

[11]. Moguel-Sánchez R., Martínez-Palacios C. S., Ocharán-Hernández J. O., Limón X., Sanchez Garcia A. J. (2022). Zenodo: Bots and their Uses in Software Development: A Systematic Mapping Study. [Data Set] https://doi.org/10.5281/zenodo.7872403.

[12]. Wohlin C. (2014). Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering.

[13]. Cruzes D., Dybå T. Recommended Steps for Thematic Synthesis in Software Engineering, Dept. of Computer and Information Science, Trondheim, Norway, 2011.

[14]. Matthies C., Dobrigkeit F., Hesse G., An additional set of (automated) eyes: chatbots for agile retrospectives. In 2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE), pp. 34–37, IEEE, 2019.

[15]. Balachandran V. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In 2013 35th International Conference on Software Engineering (ICSE), pp. 931–940, IEEE, 2013.

[16]. McConnell S. Professional software development: shorter schedules, higher quality products, more successful projects, enhanced careers. Addison-Wesley, 2004.

Информация об авторах / Information about authors

Рикардо МОГЕЛЬ-САНЧЕС имеет степень бакалавра по программной инженерии. С 2024 года является профессором Младшего колледжа Маффлс. Сфера научных интересов: кибербезопасность, искусственный интеллект, технологическая доступность.

Ricardo MOGUEL-SÁNCHEZ – Bachelor of Science in Software Engineering. Professor, Muffles Junior College since 2024. His research interests include Cybersecurity, Artificial Intelligence, and Technological Accessibility.

Сесар Серхио МАРТИНЕС-ПАЛАСИОС окончил Университет Веракруса, получив степень бакалавра по программной инженерии. Работает над комплексным созданием веб-продуктов по технологии .NET. Сфера научных интересов: искусственный интеллект, технологическая доступность, трехмерная графика.

César Sergio MARTÍNEZ-PALACIOS – graduated from the University of Veracruz with a Bachelor’s degree in Software Engineering. Dedicated to full stack .NET development in different environments. With Research interest in Artificial intelligence, accessibility and 3D Graphics.

Хорхе Октавио ОЧАРАН-ЭРНАНДЕС – профессор факультета статистики и информатики Университета Веракруса. Получил степень бакалавра по стратегическим информационным технологиям в кампусе Веракруса сети университетов Анауака, степень магистра по программной инженерии и степень PhD по программированию в Университете Веракруса. В сферу его научных интересов входят программная инженерия, инженерия требований, проектирование и разработка архитектуры программного обеспечения, разработка прикладных программных интерфейсов, гуманитарные аспекты программной инженерии. Является членом ассоциаций ACM и IEEE.

Jorge Octavio OCHARÁN-HERNÁNDEZ – is a Full Professor at the Faculty of Statistics and Informatics of Universidad Veracruzana. He holds a Ph.D. in Computer Science and a Master’s in Software Engineering from Universidad Veracruzana and a BC in Strategic Information Technologies from Universidad Anáhuac Campus Veracruz. His research interests include Software Engineering, Requirement Engineering, Software Design and Architecture, API Design, and Human Aspects of Software Engineering. He is a member of the ACM and IEEE Computer Society.

Ксавьер ЛИМОН – профессор факультета статистики и информатики Университета Веракруса. Является членом мексиканской Национальной системы поддержки исследователей. Имеет степень бакалавра по информатике, а также степени магистра и PhD по искусственному интеллекту. Сфера научных интересов: мультиагентные системы, добыча данных, кибербезопасность и распределенные системы. Является автором многочисленных публикаций в этих предметных областях.

Xavier LIMÓN – works as a full-time professor at the Faculty of Statistics and Informatics at the University of Veracruz. He is currently a member of the National System of Researchers. He holds a Bachelor's degree in Informatics, as well as a Master's and Doctorate degree in Artificial Intelligence. His areas of research interest include Multi-Agent Systems, Data Mining, Cybersecurity, and Distributed Systems. He has numerous publications in these fields.

Анхель Хуан САНЧЕС-ГАРСИЯ имеет степень PhD по искусственному интеллекту. Профессор и исследователь Школы статистики и информатики Университета Веракруса с 2012 года. Научные интересы: машинное обучение в приложении к программной инженерии,

эволюционные вычисления, измерения программного обеспечения, машинное зрение и робототехника.

Angel Juan SÁNCHEZ-GARCÍA – Doctor of Artificial Intelligence, Professor and researcher of the School of Statistics and Informatics of the Universidad Veracruzana since 2012. Research interests: machine learning applied to software engineering, evolutionary computation, software measurement, computer vision and robotics.