Towards Human-Bot Collaborative Software Architecting with ChatGPT

Aakash Ahmad
School of Computing and Communications, Lancaster University Leipzig, Leipzig, Germany
a.ahmad13@lancaster.ac.uk

Muhammad Waseem*
Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
mwaseem@jyu.fi

Peng Liang
School of Computer Science, Wuhan University, Wuhan, China
liangp@whu.edu.cn

Mahdi Fahmideh
School of Business, University of Southern Queensland, Queensland, Australia
mahdi.fahmideh@usq.edu.au

Mst Shamima Aktar
School of Computer Science, Wuhan University, Wuhan, China
shamima@whu.edu.cn

Tommi Mikkonen
Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
tommi.j.mikkonen@jyu.fi

ABSTRACT

Architecting software-intensive systems can be a complex process. It deals with the daunting tasks of unifying stakeholders' perspectives, designers' intellect, tool-based automation, pattern-driven reuse, and so on, to sketch a blueprint that guides software implementation and evaluation. Despite its benefits, architecture-centric software engineering (ACSE) suffers from a multitude of challenges. ACSE challenges could stem from a lack of standardized processes, socio-technical limitations, and scarcity of human expertise etc. that can impede the development of existing and emergent classes of software. Software Development Bots (DevBots) trained on large language models can help synergise architects' knowledge with artificially intelligent decision support to enable rapid architecting in a human-bot collaborative ACSE. An emerging solution to enable this collaboration is ChatGPT, a disruptive technology not primarily introduced for software engineering, but is capable of articulating and refining architectural artifacts based on natural language processing. We detail a case study that involves collaboration between a novice software architect and ChatGPT to architect a service-based software. Future research focuses on harnessing empirical evidence about architects' productivity and explores socio-technical aspects of architecting with ChatGPT to tackle challenges of ACSE.

CCS CONCEPTS

• **Software and its engineering** → **Human-centered computing**; • **Computing methodologies** → *Artificial intelligence*; • **Philosophical/theoretical foundations of artificial intelligence**; • **Social and professional topics** → Empirical studies;

KEYWORDS

Software Architecture, ChatGPT, Large Language Models, DevBots

ACM Reference Format:
Aakash Ahmad, Muhammad Waseem, Peng Liang, Mahdi Fahmideh, Mst Shamima Aktar, and Tommi Mikkonen. 2023. Towards Human-Bot Collaborative Software Architecting with ChatGPT. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE '23), June 14–16, 2023, Oulu, Finland*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3593434.3593468

*Corresponding author

EASE '23, June 14–16, 2023, Oulu, Finland

ACM ISBN 979-8-4007-0044-6/23/06.
https://doi.org/10.1145/3593434.3593468

1 INTRODUCTION

Architecture of software-intensive systems enables architects to specify structural composition, express behavioural constraints, and rationalise design decisions - hiding implementation complexities with architectural components - to sketch a blue-print for software implementation [12]. Architecture-centric Software Engineering (ACSE) aims to exploit architectural knowledge (e.g., tactics and patterns), architectural languages, tools, and architects' decisions (human intellect) etc. to create a model that drives the implementation, validation, and maintenance phases of software systems [9]. In recent years, ACSE approaches have been leveraged to research and develop emergent classes of software systems such as blockchain applications and quantum computing services [1]. ACSE approaches have been proven useful to systematize software development in an industrial context [9]. Despite its potential, ACSE faces a several challenges such as mapping stakeholders' perspectives to architectural requirements, managing architectural drift, erosion, and technical debts, or lack of automation and architects' expertise in developing complex and emergent classes of software [1, 12]. In such a context, software architects/engineers may enter a phase referred to as the *'lonesome architect'*. A lonesome architect requires non-intrusive support rooted in processes and tools to address the challenges of ACSE by reusing knowledge and exploiting decision support to develop software systems [10].

Context and motivation: The process to architect software systems (a.k.a., 'architecting process') covers a number of activities that support an incremental, process-centric, and systematic approach to apply ACSE in software development [1, 9]. Empiricism remains fundamental to deriving and/or utilising architecting processes that can support activities, such as analysis, synthesis,

and evaluation etc. of software architetures [10]. To enrich the architecting process and empower the role of architects, research and development has focused on incorporating patterns and styles (knowledge), recommender systems (intelligence), and distributed architecting (collaboration) in ACSE process. Artificial Intelligence for Software Engineering (AI for SE) is an active area of research that aims to synergize solutions of AI and practices of SE to instill intelligence in the processes and tools for software development [4, 19]. From an ACSE perspective, the research on AI primarily focuses on developing decision support systems or development bots that can assist architects with recommendations about design decisions, selection of patterns and styles, or predict points of architectural failure and degradation [8]. Currently, there is a lack of research or solutions to enrich the AI-enabled collaborative architecting process. Collaborative architecting can unify architects' knowledge as human intellect and bot's capability as an intelligent agent who can lead the architecting process based on human conversation and supervision. Such collaboration can allow architects to delegate their architecting tasks to the bot, supervise the bot via dialog in natural language(s) to achieve automation, and relieve architects from undertaking tedious tasks in ACSE.

Objective of the study: Chat Generative Pre-trained Transformer (ChatGPT) has emerged as a disruptive technology, representing an unprecedented example of a bot, that can engage with humans in context-preserved conversations to produce well-articulated responses to complex queries [3, 16]. ChatGPT is not specifically developed to address software engineering challenges, however; it is well capable of generating versatile textual specifications including architectural requirements, Unified Modeling Language (UML) scripts, source code libraries, and test cases [11, 17]. Studies have started to explore the role of ChatGPT in engineering education, software testing, and source code generation [16, 17]. Considering the benefits of ACSE, this study explores the role of ChatGPT as a DevBot in the process of collaborative software architecting. This study aims to understand *how ChatGPT can undertake architecting activities to analyze, synthesize, and evaluate software architecture in a human-bot collaborative architecting?*

Contributions: We followed a process-centric approach [9] and adopted scenario-based method [6] for ChatGPT-enabled architectural analysis, synthesis, and evaluation of a microservices-based software. Preliminary results highlight ChatGPT's capabilities that include but are not limited to processing an architecture story (conversed to it by an architect) for articulating architectural requirements, specifying models, recommending and applying architectural tactics and patterns, and recommending scenarios for architecture evaluation. Primary contributions of this study are to:

- Investigate the human-bot collaborative architecting that can synergize ChatGPT's outputs and architects' decisions to automate ACSE with a preliminary case study.
- Identify the pros and cons of ChatGPT assisted ACSE to identify issues concerning ethics, governance, and socio-technical constraints of collaborative architecting.
- Establish foundations for empirically-grounded evidence about ChatGPT's capabilities and architects' productivity in collaborative architecting (ongoing and future work).

The results can help SE researchers to formulate new hypotheses about the role of ChatGPT in ACSE and investigate human-bot collaborative architecting of emergent and futuristic software. Practitioners can follow the presented guidelines to experiment with delegating their tedious tasks of ACSE to ChatGPT.

2 RESEARCH CONTEXT AND METHOD

We contextualize some core concepts (Section 2.1, Figure 1) and discuss the research method (Section 2.2, Figure 2).

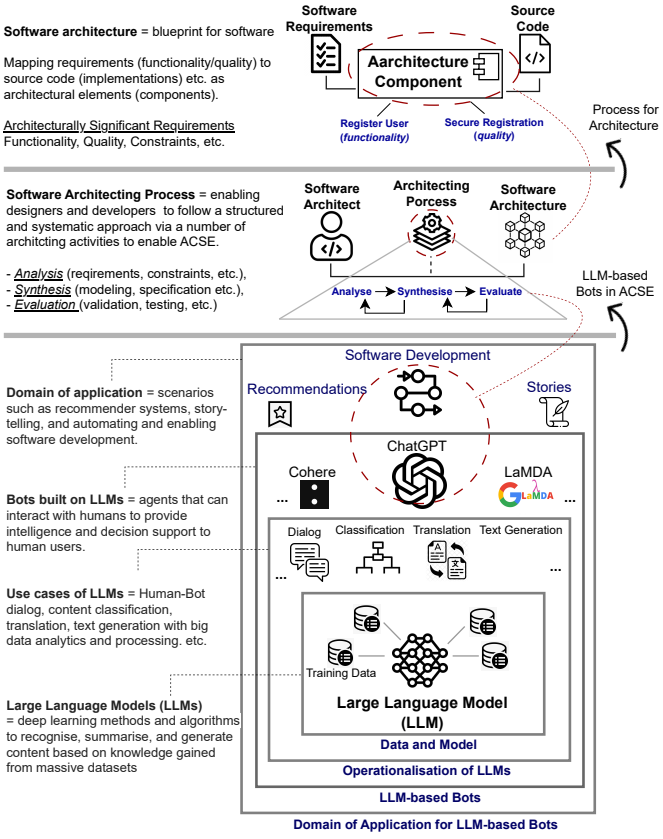

Figure 1: Context: LLMs, DevBots, Process, and Architecture

2.1 Human-Bot Collaborative Architecting

Software Architecture as described in the ISO/IEC/IEEE 42010:2011 standard, aims to abstract complexities rooted in source code-based implementations with architectural components and connectors that represent a blueprint of software applications, services, and systems to be developed [12]. To enable software architects with a systematic and incremental design of software architectures, there is a need for **architecting process** - also referred to as the process for architecting software [1, 9]. Architecting process can have a number of fine-grained architecting activities that support a separation of architectural concerns in ACSE. For example, the architecting process reported in [9] and illustrated in Figure 1 is derived from five industrial projects and incorporates three architecting activities

namely *architectural analysis*, *architectural synthesis*, and *architectural evaluation*. For instance, the architectural evaluation activity in the process focuses on identifying, prioritizing, and documenting architectural scenarios as use cases to evaluate the designed architecture [6]. In the architecting process, an architect can extract and document the requirements that express the required functionality and desired quality of the software, referred to as Architecturally Significant Requirements (ASRs). ASRs need to be mapped to source code implementations via an architectural model that can be visualized or textually specified using architectural languages, such as the UML or Architectural Description Languages (ADLs) [13]. Architecture models that reflect the ASRs need to be evaluated using an architecture evaluation method, such as Software Architecture Analysis Method (SAAM) or Architecture Tradeoff Analysis Method (ATAM) [6].

Software Development Bots (DevBots) represent conversational agents or recommender systems, driven by AI, to assist software engineers by offering certain degree of automation and/or inducing intelligence in software engineering process [14, 15]. One such example of a DevBot is the recently introduced GitHub Copilot, an "AI pair programmer", that can process natural language descriptions provided by programmers to generate corresponding source code in several programming languages [14]. From the software architecting perspective, the role of AI in general and DevBots to be specific is limited to bots answering questions or proposals about providing recommendations to manage architectural erosion and maintenance [8]. There is no research that investigates or any solution that demonstrates an architecting process by incorporating DevBots to enable human-bot collaborative architecting of software systems. Such a collaboration can enrich the architecting process that goes beyond questions & answers and recommendations, and synergizes architects' intellect (human rationale) and bot's intelligence (automated architecting process) in ACSE. Collaborative architecting can empower novice designers or architects, who lack experience or professional expertise to specify their requirements in natural language and DevBots can translate them into ASRs, architectural models, and evaluation scenarios.

2.2 Research Method

It comprises of three main phases, detailed below, as in Figure 2.

Phase 1 - Developing the Architecture Story: Software architecture story refers to a textual narration of the envisaged solution, i.e., software to be developed by expressing the core functionality, desired quality (i.e., ASRs) and any constraints in a natural language. The story is developed based on analyzing software domain that represents an operational context of the system or collection of scenarios operationalised via a software solution. The architect can analyze the domain and identify scenarios to write an architecture story that is fed to ChatGPT via a prompt for phase 2.

Phase 2 - Enabling Collaborative Architecting based on three activities adopted from [9], detailed below.

- *Architectural analysis* is driven by architecture story fed to ChatGPT for articulating the ASRs via (i) automatically generated and recommended requirements (by ChatGPT), or (ii) manual specification of the requirements (by the architect), or (iii) a continuous dialog between ChatGPT and the architect to refine (add/remove/update) the requirements.
- *Architectural synthesis* consolidates the ASRs to create an architecture model or representation that can act as a point of reference, visualizing the structural (de-)composition and runtime scenarios for the software. We suggest the UML as a representative example for architectural synthesis due to a number of factors, such as available documentation, ease of use, diversity of diagrams, tool support, and wide-scale adoption as a language to represent software systems [13].
- *Architectural evaluation* evaluates the synthesized architecture against ASRs based on scenarios from the architectural story. Architectural evaluation is conducted incrementally for full or partial validation of the architecture based on use cases or scenarios from ASRs. We used the Software Architecture Analysis Method (SAAM) to supervise ChatGPT for evaluating the architecture [6].

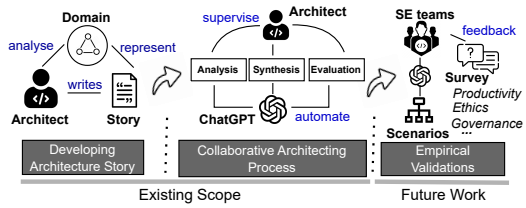

Figure 2: Overview of the Research Method

Phase 3 - Conducting the Empirical Validations complements the initial two phases with empirical validations of collaborative architecting as an extension of this study, outlining future work. The existing scope aims to explore and present the role of ChatGPT in human-bot collaborative software architecting (in Section 3). Future work on empirically grounded guidelines to understand a multitude of socio-technical issues associated with ChatGPT-driven architecting is discussed later (in Section 5).

3 COLLABORATIVE ARCHITECTING

This section details the process of collaborative architecting demonstrated with a case study for scenario-based exemplification as shown in Figure 3. The **case study** aims to develop a software application named CampusBike that allows campus visitors to 'register', 'view available bikes', 'reserve a bike', 'make payments', and 'view usage reports' etc. for eco-friendly mobility in and around the campus. The **architect** has a working knowledge of software design (UML, patterns etc.), implementation (programming and scripting languages), architecture scenarios, and evaluation methods to design and develop the CampusBike application [2]. Figure 3 provides an overview of the collaborative architecting process with three architecting activities adopted from [9] and using SAAM as the architecture evaluation method [6]. Figure 4 - Figure 6 show specific examples of the interaction between the architect and ChatGPT to collaboratively undertake each of the three activities namely architecture analysis, architecture synthesis, and architecture evaluation as shown in Figure 3 with additional details in [2]

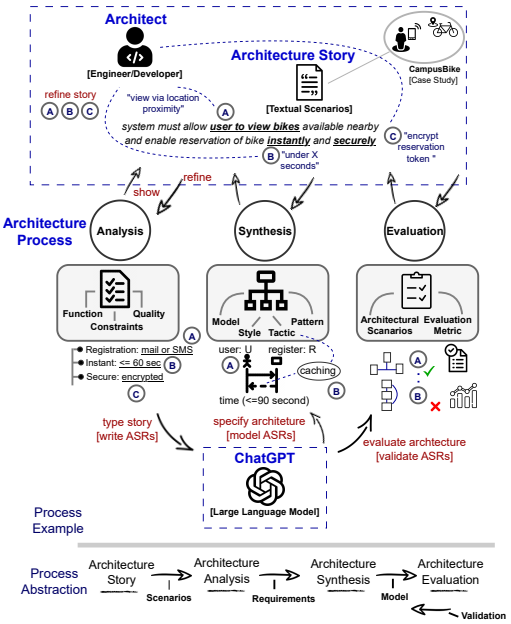

Figure 3: Overview of the Collaborative Architecting Process

3.1 Formulating the Architecture Story

Architecture story refers to a textual narration of the envisaged solution, i.e., software to be developed by expressing the core functionality and any constraints as textual narrations. As per the research method in Figure 2, the story is developed based on analysing software domain that represents an operational context of the system or collection of scenarios operationalized via a software solution. The architect can analyze the domain and identify any scenarios to write an architecture story, fed to ChatGPT, that sets the foundation for architectural analysis activity in the process. A detailed architecture story is available at [2], with its sample snippet and two scenarios highlighted below.

Snippet of Architecture Story

" … as a step towards maintaining a 'Green Campus' - minimising the carbon footprint, congestion, and noise created by vehicles, University's administration has decided to introduce a bike service where campus visitors can avail of a pay-per-use bicycle facility on an hourly or daily basis for enhanced mobility in and around the campus. Potential bikers can register and view available bikes in their proximity (e.g., within 500 meters) and reserve them for a specific time after payment. The administration needs a software application called 'CampusBike' that operates in a networked environment and ensures security and privacy of users' data …"

Scenario Example: View available bikes (using location proximity), reserve a bike for a specific time (pay-and-reserve).

Architect's Refinements
Functionality: View Bike - via location proximity
Quality: Instantly - within 90 seconds, Securely - encrypt reservation token
Constraint: apply data minimization on registration data (GDPR constraint)

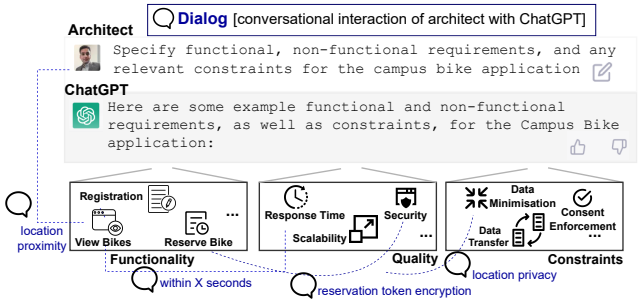

Figure 4: Formulating and Refining the Requirements

3.2 Architectural Analysis

During architectural analysis, the focus is to specify ASRs as required functionality (e.g., view available bikes) and desired quality (e.g., response time < N) along with any constraints (e.g., compliance with relevant data security policies) of CampusBike software. ChatGPT is capable of outlining the ASRs or any necessary constraints if queried by the architect. However, ChatGPT expressed the ASRs and constraints that were refined (add, remove, and modify any requirements) by the architect. For example, the 'Reserve Bike' requirement articulated by ChatGPT read as: '*... system must allow user to view bikes available nearby and enable reservation of the bike instantly and securely*'. The architect refined the requirements:

After narrating the architecture story, Figure 4 shows architects' query and ChatGPT's response (human-bot collaboration) to specify the required functionality, desired quality, and any constraints (i.e., ASRs) to complete architectural analysis. ASRs are iteratively refined via human-bot dialog to complete architectural analysis [2].

3.3 Architectural Synthesis

The ASRs are synthesized into an architectural specification that can be expressed with a modeling language like UML or other architectural languages [13]. The architect used UML class and component diagrams to represent the overall architecture and fine-grained representation of the architectural design. During synthesis, the UML class diagram was refined with the application of singleton pattern to 'UserLogin' class. Singelton helped to restrict a single login session of the user across the devices. Figure 5 shows the architect's instruction for ChatGPT's to create the script for UML class diagram. Additional dialog between the two enabled application of singleton pattern, caching tactic, and data minimisation constraint on class diagram, presented in [2].

3.4 Architectural Evaluation

We used the SAAM method [6] to evaluate the synthesized architecture, as illustrated in Figure 6. For example, the architect specifies the application of SAAM to evaluate the 'View Bike' component. ChatGPT presents the scenario for evaluating the 'View Bike' component individually and also scenarios where it interacts with other components. Using SAAM, ChatGPT identified and prioritized the scenarios to evaluate the ASRs, i.e., three prioritized scenarios each

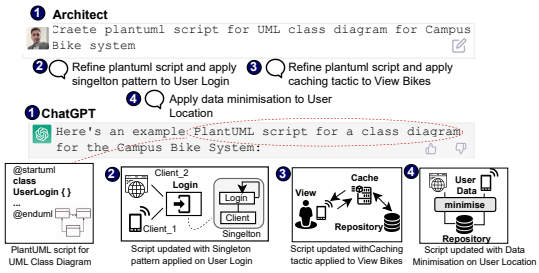

Figure 5: Modeling and Refining the Architecture Design

to evaluate the functionality (e.g., view bikes), quality (e.g., response time < X), and constraints (e.g., data minimization).

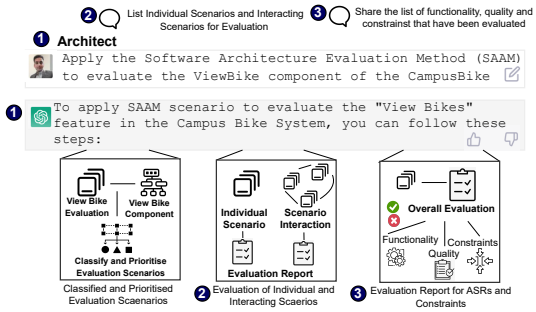

Figure 6: Evaluating the Architecture

4 RELATED WORK

We review the relevant existing research on AI for SE and ACSE (Section 4.1) and the role of ChatGPT in SE (Section 4.2).

4.1 AI in Software Engineering and Architecting

The research on synergizing AI and SE can be generally classified into two dimensions namely AI for SE (artificial intelligence for software engineering) and SE for AI (software engineering for artificial intelligence) [19] [4]. Considering the AI for SE perspective, Xie [19] argued that SE research needs to go beyond traditional efforts of applying AI for tool-based automation and pattern selection with an exploration of methods that instil intelligence in software engineering processes and solutions. Specifically, SE solutions need to maintain the so-called 'intelligence equilibrium' – i.e., unifying and balancing machine intelligence and human intellect – in processes, patterns, and tools etc. for emergent classes of software, such as blockchain and quantum applications [18]. Barenkamp *et al.* [4] combined the findings of a systematic review and interviews with software developers to investigate the role of AI techniques in SE processes. The results of their study pinpoint three areas where SE solutions need intelligence to tackle (i) automation of tedious and complex activities such as code debugging, (ii) big data analytics to discover patterns, and (iii) evaluation of data in neural and software-defined networks. Considering AI for software architecting, Herold *et al.* [8] investigated existing research and proposed a conceptual framework for the application of machine learning to mitigate architecture degradation. From the SE process perspective, an empirical study reported by Nguyen et al. [14] (investigating Github Copilot) and a tool (Chatbot named SOCIO) developed by Pérez-Soler et al.[15] highlight the growing impacts and emerging challenges of using bots in software modeling and programming.

4.2 ChatGPT Assisted Software Engineering

From the SE perspective, ChatGPT is viewed as an unprecedented example of a chatbot that can produce well-articulated responses to complex queries. However, it remains an unexplored area in terms of its potential and perils in the context of software development processes [5, 7]. Most recently, a number of proposals and experimental findings indicate that the research on ChatGPT focuses on supporting engineering education [11, 16], software programming [3, 7], and testing [17]. Avila-Chauvet *et al.* [3] detailed how conversational dialogs of a programmer with ChatGPT enable a human-bot assisted development of an online behavioral task using HTML, CSS, and JavaScript source code. They highlighted that although ChatGPT requires human oversight and intervention, it can write well-scripted programming solutions and reduces the time and effort of a developer during programming. The narrative expressed in [7] advocates for an incremental process (human dialog with ChatGPT) to enable genetic programming - JavaScript code to solve the traveling salesman problem. In addition to developing the source code, a couple of studies have focused on testing and debugging with ChatGPT [11, 17]. Sobania *et al.* [17] evaluated the performance of ChatGPT in automated bug fixing.

Conclusive summary This study complements the most recent research efforts on software test automation and bug fixing with ChatGPT [17] but focuses on architecture-centric development for software systems. Considering AI for SE [19], collaborative architecting can enrich ACSE process by synergizing architects' knowledge with bot's capabilities to architect software systems.

5 DISCUSSION AND VALIDITY THREATS

We highlight some socio-technical issues of collaborative architecting (Section 5.1) and potential threats to validity (Section 5.2).

5.1 Socio-Technical Issues of ChatGPT in ACSE

The socio-technical aspects refer to a unified perspective on issues such as *what can be 'social' concerns* and *what are the 'technical' limitations* of collaborative architecting.

Response Variation: In the context of human-bot conversational dialogs, ChatGPT may produce varied responses for exact same queries. For example, we observed that a query such as '*... what architectural style can be best suited to CampusBike system*' may yield varied responses, such as microservices, layered, client-server etc. architecture can be best suited for the system. This and alike variation in recommendations or scripted artifacts (UML script, ASR specification etc.) can impact the consistency of architecting process and ultimately varied analysis, synthesis, and evaluation of the architecture. One of the solutions to minimize response variations is an iterative dialog with ChatGPT to refine its output and architects' oversight to ensure that architectural artifacts are consistent.

Ethics and Intellectual Property: Textual specifications, architecture specific scripts, and source codes etc. articulated by ChatGPT could give rise to ethical issues or in some cases copyright or

intellectual property infringements. For example, ChatGPT generated script for a component (GETLOCATION) that senses user location in CampusBike system may lead to leakage of user location privacy and non-compliant software with regulatory guidelines (GDPR etc.) that must be dealt with vigilance. In such cases, the role of the architect is critical to ensure the generated architecture does not violate ethics or intellectual property rights (if any).

Biased Outputs: The biases in outputs of such conversational bots can be attributed to a number of possible aspects including but not limited to input, training data, and/or algorithmic bias. From an architectural perspective, recommendation bias about specific architectural modeling notation, tactic, pattern, or style etc. may be based on its widespread adoption or bias in training data. Architectural recommendations (specific style), design decisions (pattern selection), or validation scenarios (evaluation method) may suffer such bias to produce sub-optimal artifacts in ACSE.

5.2 Threats to Validity

These represent the limitations, constraints, or flaws in the study that affect the generalization, replicability, and validity of results.

Internal validity examines the extent to which any systematic error (bias) is present in the design, conduct, and analysis etc. of the study. To design and conduct this study, and considering the internal validity, we followed a systematic approach and utilized a well-known architecting process [9] and architecture evaluation method [6]. The case study based approach combined with incremental architecting (Figure 3) helped us to analyze and refine the study.

External validity examines whether the findings of a study can be generalized to other contexts. We only experimented with a single case study of moderate complexity that can confine the study's generalization. Scenarios with the increased complexity of architecting process (cross-organisational development), class of software to be developed (mission-critical software), and human expertise (novice/experienced engineers) can affect the external validity of this research. Future work is planned to validate the process of collaborative architecting by engaging teams of architects to understand the extent to which external validity can be minimized.

Conclusion validity determines the degree to which the conclusions reached by the study are credible or believable. In order to minimize this threat, we followed a three-step process (Figure 2) to support a fine-grained process to architect the software and validate the results (future work). Moreover, a case study based approach was adopted to ensure scenario-based demonstration of the study results. However, some conclusions (e.g., architect's productivity, ChatGPT's efficacy) can only be validated with more experimentation and real context scenarios of collaborative architecting.

6 CONCLUSIONS AND FUTURE RESEARCH

This research investigates the role of ChatGPT to assist the architects who in ACSE process. The research advocates that in the context of AI for SE, traditional efforts of applying AI for tool-based automation should focus on a broader perspective, i.e., enriching existing processes by instilling intelligence in them via efforts like human-bot collaborative architecting. The case study reflects a case of (i) how a software can be architected with ChatGPT and (ii) what factors need to be considered in collaborative architecting. Variation in responses and artifacts, types of ethical implications, level of human decision support/supervision, along with legal and socio-technical issues must be considered while integrating ChatGPT in ACSE. The research requires an empirical understanding of architects' productivity in developing software with ChatGPT.

Needs for future research: We plan to extend this study to explore human feedback and validation (i.e., architects' perspective) and integrate ChatGPT in a process to develop software for quantum computing systems. More specifically, quantum software engineering has emerged as a quantum specific genre of SE that faces a lack of human expertise to synergize the skills of engineering software and knowledge of quantum physics. We are currently working on engaging a number of software development teams with diverse demography (e.g., geo-distribution, type of expertise, level of experience, class of software) in controlled experiments to architect software and document architects' responses.

ACKNOWLEDGMENTS

This work is funded by the University of Jyväskylä, Finland research grant, and the NSFC China with grant number 62172311.

REFERENCES

[1] Aakash Ahmad, Arif Ali Khan, Muhammad Waseem, Mahdi Fahmideh, and Tommi Mikkonen. 2022. Towards process centered architecting for quantum software systems. In *Proceedings of the 1st IEEE International Conference on Quantum Software (QSW)*. IEEE, 26–31.

[2] Anonymous. 2023. Replication Package: Towards Human-Bot Collaborative Software Architecting with ChatGPT. In *https://anonymous.4open.science/r/ChatGPT4SA-2B61*.

[3] Laurent Avila-Chauvet, Diana Mejía, and Christian Oswaldo Acosta Quiroz. 2023. ChatGPT as a Support Tool for Online Behavioral Task Programming. *SSRN preprint SSRN:4329020* (2023).

[4] Marco Barenkamp, Jonas Rebstadt, and Oliver Thomas. 2020. Applications of AI in classical software engineering. *AI Perspectives* 2, 1 (2020), 1.

[5] Ali Borji. 2023. A Categorical Archive of ChatGPT Failures. *arXiv preprint arXiv:2302.03494* (2023).

[6] Liliana Dobrica and Eila Niemela. 2002. A survey on software architecture analysis methods. *IEEE Transactions on Software Engineering* 28, 7 (2002), 638–653.

[7] Fernando Doglio. 2022. The Rise of ChatGPT and the Fall of the Software Developer — Is This the Beginning of the End? https://tinyurl.com/3mxrfmjh

[8] Sebastian Herold, Christoph Knieke, Mirco Schindler, and Andreas Rausch. 2020. Towards Improving Software Architecture Degradation Mitigation by Machine Learning. In *Proceedings of the 12th International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE)*. IARIA, 36–39.

[9] Christine Hofmeister, Philippe Kruchten, Robert L Nord, Henk Obbink, Alexander Ran, and Pierre America. 2007. A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software* 80, 1 (2007), 106–126.

[10] Johan F Hoorn, Rik Farenhorst, Patricia Lago, and Hans Van Vliet. 2011. The lonesome architect. *Journal of Systems and Software* 84, 9 (2011), 1424–1435.

[11] Sajed Jalil, Suzzana Rafi, Thomas D LaToza, Kevin Moran, and Wing Lam. 2023. ChatGPT and Software Testing Education: Promises & Perils. *arXiv preprint arXiv:2302.03287* (2023).

[12] Philippe Kruchten, Henk Obbink, and Judith Stafford. 2006. The past, present, and future for software architecture. *IEEE Software* 23, 2 (2006), 22–30.

[13] Ivano Malavolta, Patricia Lago, Henry Muccini, Patrizio Pelliccione, and Antony Tang. 2012. What industry needs from architectural languages: A survey. *IEEE Transactions on Software Engineering* 39, 6 (2012), 869–891.

[14] Nhan Nguyen and Sarah Nadi. 2022. An empirical evaluation of GitHub copilot's code suggestions. In *Proceedings of the 19th International Conference on Mining Software Repositories (MSR)*. IEEE, 1–5.

[15] Sara Pérez-Soler, Esther Guerra, and Juan de Lara. 2018. Collaborative modeling and group decision making using chatbots in social networks. *IEEE Software* 35, 6 (2018), 48–54.

[16] Junaid Qadir. 2022. Engineering Education in the Era of ChatGPT: Promise and Pitfalls of Generative AI for Education. *TechRxiv preprint techrxiv.21789434* (2022).
[17] Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. 2023. An Analysis of the Automatic Bug Fixing Performance of ChatGPT. *arXiv preprint arXiv:2301.08653* (2023).
[18] Eoin Woods. 2016. Software architecture in a changing world. *IEEE Software* 33, 6 (2016), 94–97.
[19] Tao Xie. 2018. Intelligent software engineering: Synergy between AI and software engineering. In *Proceedings of the 11th Innovations in Software Engineering Conference (ISEC)*. ACM, 1–1.