

LLMs across the Software Development Life Cycle in Project-Based Software Engineering Courses: an Empirical Study

Christos Hadjichristofi*, Dimitrios GeroKonstantis*, Panagiotis Papadeas*, Vassilios Vescoukis†

* 0009-0000-7976-8749

0009-0004-0950-8414

0009-0003-3012-0089

Software Engineering Lab

National Technical University of Athens

Zografos 15780, Greece

Email: {hadjichristofi_ch, dgerokonstantis, papadeas}@cs.ntua.gr

† 0000-0002-5360-8349

Software Engineering Lab and Geoinformation Center

National Technical University of Athens

Zografos 15780, Greece

Email: v.vescoukis@cs.ntua.gr

Abstract—Large Language Models (LLMs) are increasingly used in software development and are becoming part of software engineering education. However, there is limited empirical evidence on how students use AI tools across development tasks in project-based Software Engineering courses. This paper presents an empirical study based on approximately 8,500 validated student-submitted AI usage logs collected over five semesters between 2022 and 2026 from two undergraduate courses at the National Technical University of Athens. We examine in which phases of the Software Development Life Cycle (SDLC) students use AI tools, how they evaluate AI assistance, and how their perceptions evolve over time. The results show that AI usage is mainly concentrated in coding and testing, with lower usage in requirements, architecture, and design. Students rate quality higher than knowledge gained, and those with higher task experience report higher quality and knowledge gains. Overall, students view AI positively, report low perceived threat, and tend to use it more for task completion than for learning.

I. INTRODUCTION

LARGE Language Models (LLMs) and other AI-based tools are increasingly being incorporated into software development practice, including in educational environments. In project-based Software Engineering courses, students use such tools not only for code generation, but also for requirements analysis, architecture and design tasks, testing, and documentation across multiple phases of the Software Development Life Cycle (SDLC).

Despite this growing adoption, there is still limited empirical evidence on how students use AI tools throughout the SDLC in course projects, how they evaluate their usefulness across different tasks, and how these perceptions evolve over time. This gap is particularly relevant for rethinking Software Engineering education, where AI tools are becoming part

of everyday student practice while also taking on a distinct educational role.

This paper presents an empirical study based on student-submitted AI-usage logs collected over multiple semesters in two Software Engineering courses at the National Technical University of Athens, starting from the early days of the recent “AI boom”. Students were encouraged to use AI tools throughout their project work, to document these interactions, and to submit one record per interaction in a custom logging service. The resulting dataset includes objective characteristics of AI-assisted interactions (context), such as development phase, action type, programming language, and tool used, as well as subjective assessments, such as perceived quality of help, knowledge acquired, time savings, and perceived threat to their future roles as Software Engineers.

The contributions of this paper are as follows:

- a publicly open dataset of multi-semester student-reported AI usage logs collected in project-based Software Engineering courses ¹,
- an analysis of this dataset to examine how students use AI tools during software development,
- an analysis of how students evaluate AI tools in terms of usefulness, knowledge gained, and perceived risks, and
- observations on how these usage patterns and perceptions change over time.

To guide the analysis, the study addresses the following research questions:

- **RQ1:** On which software development tasks do students focus their use of AI tools?

¹<https://github.com/ntua/aiinsecourses>

- **RQ2:** How do students rank the quality, knowledge gained, and time saved when using AI assistants?
- **RQ3:** Do students perceive AI as a supportive tool or as a potential threat, both currently and in the future?

The remainder of this paper is organized as follows. Section II reviews relevant literature. Section III presents the study context, dataset, and analysis procedure. Section IV presents the empirical findings. Section V discusses the implications of the results. Section VI outlines threats to validity, and Section VII concludes the paper.

II. RELATED WORK

Existing research on LLMs in Software Engineering covers their use across multiple development activities. Prior work reports applications in tasks such as code generation, documentation, testing, and maintenance, while also identifying challenges related to reliability and other limitations [1], [2]. These studies provide an overview of how LLMs are used in Software Engineering and the constraints that accompany their use.

Many of these studies have also been conducted in educational settings, particularly in programming courses, where LLMs are now widely utilized, either just by learners themselves, or by instructors as part of the learning process. Early work examined the implications of AI-based code generation tools, highlighting both opportunities for assisting novice programmers and challenges related to achieving or even re-defining the learning outcomes of programming courses while maintaining academic integrity [3], [4], [5]. Subsequent studies investigated how LLM-based assistants support problem solving and interaction through natural language, enabling students to generate and understand code based on textual input [6], [5]. Other work evaluated their effectiveness in supporting novice learners, including their impact on code comprehension, explanation generation, and performance in programming assessments [5], [7], [8], [9]. Collectively, these studies show the increasing use of LLMs in programming education and their influence on learning practices.

More recent studies examine how students use and experience LLMs in programming contexts. Several studies report generally positive student attitudes toward tools such as ChatGPT, while also noting concerns about answer accuracy and quality, over-reliance, ethics, and possible effects on learning [10], [11], [12]. Related work examines how students incorporate generative AI into programming activities, identifying recurring usage patterns and interaction strategies [13], [14]. Research on human–AI collaboration in programming education further explores how students’ learning, trust, productivity, and coding practices relate to their use of LLM-based assistants [15]. Recent empirical work also examines the effects of ChatGPT on student performance, creativity, and teamwork [16].

Within Software Engineering project contexts, Rasnayaka, Wang, Shariffdeen, and Iyer [17] present an empirical study of LLM usage and student perceptions in a course-based Software Engineering project, while Naman, Shariffdeen, Wang,

Rasnayaka, and Iyer [18] analyze interaction patterns between students and LLMs in a similar setting. These studies are among the closest to our context, as they examine LLM use within extended, project-based development environments rather than controlled tasks. Their analyses provide insights into overall usage trends, interaction dynamics, and generated artifacts.

Existing studies tend to focus on controlled programming exercises, high-level usage patterns, or general perceptions of LLMs. Less attention has been given to how students integrate LLMs into specific tasks, especially except coding, within Software Engineering projects. Examining LLM usage at the level of individual tasks is important for understanding how these tools influence development workflows, decision-making, and learning processes in project-based educational settings.

In parallel, prior work has examined limitations associated with LLM-based tools. Studies report usability issues and mismatches between user expectations and actual tool behavior [19], as well as concerns regarding code quality and correctness, and the need for human oversight, particularly when such tools are used by less experienced developers [20]. Research on novice programmers further shows that interacting with AI-based coding assistants can be challenging and may lead to unintended learning outcomes [21]. Broader discussions in Software Engineering also consider the implications of adopting LLMs in development workflows, including both benefits and risks [22]. These findings reinforce the need for more empirical studies on the use of LLMs in software development practice, to provide evidence for re-shaping Software Engineering education.

To address this gap, this paper presents an empirical study of how students use LLMs across different tasks and phases of the SDLC in two project-based Software Engineering courses offered in the Electrical and Computer Engineering School of NTUA. The study examines not only where LLMs are used, but also how students evaluate their usefulness and how their perceptions of quality, learning, and risk evolve over time in relation to their level of task experience, considering that several, especially non-coding, tasks are new to them. By analyzing AI usage logs collected across multiple task submissions, we provide insights into student–LLM interaction patterns in a project-based educational setting.

III. METHODOLOGY

A. Study Context

The study was conducted in two undergraduate courses at the National Technical University of Athens: *Software Engineering* (7th semester), offered in the fall semester, and *Software-as-a-Service Technologies* (8th semester), offered in the spring semester. Both courses follow a project-based structure, where students work in teams of 3 to 6 members to develop software systems across multiple phases of a typical SDLC.

The dataset was collected over five academic semesters between the academic years 2022–2023 and 2025–2026, cov-

ering both courses. Students were encouraged to use AI-based tools during their development process, and submitting AI usage logs was integrated into the course grading rubric to ensure consistent data collection.

Students used a variety of publicly available LLMs and AI assistants, including ChatGPT, Gemini, Claude, DeepSeek, Grok, and Copilot, as shown in Table I.

TABLE I
LLM USAGE DISTRIBUTION ACROSS SEMESTERS

Tool Family	23b (%)	24a (%)	24b (%)	25a (%)	25b (%)
ChatGPT	93.5	87.5	92.5	81.6	66.0
DeepSeek	0.0	1.1	0.0	5.3	25.8
Grok	0.0	0.1	0.0	0.7	8.2
Gemini	0.7	3.4	2.8	4.8	0.0
Claude	0.0	0.0	0.0	6.7	0.0
Copilot	1.8	1.0	1.1	0.0	0.0
Mistral	0.0	0.0	0.0	0.7	0.0
LLaMA	0.0	0.0	0.0	0.1	0.0
Gemma	0.0	0.0	0.0	0.1	0.0
Other	4.0	6.9	3.6	0.0	0.0

ChatGPT was the dominant tool in all semesters, although its share decreased over time. In the final semester, DeepSeek and Grok accounted for a substantially larger proportion of submissions, indicating a more diverse distribution of tool use than in earlier semesters. Students used the model versions available at the time of each semester. However, the table reports tool families rather than exact model versions, since version information was not consistently recorded. The *Other* category includes less frequently used LLMs and locally deployed open-source models.

A total of **507** students contributed to the dataset, producing **11326** submissions.

B. Data Collection

Data were collected through a custom web-based platform that allowed students to document their interactions with AI tools while working on project tasks. Each submission corresponds to a single AI-assisted interaction associated with a specific development task or subtask.

In earlier semesters, students submitted logs as compressed files containing structured JSON data, where responses corresponded to predefined questionnaire fields. Since the platform did not support real-time interaction, submissions were often recorded after completing tasks, which may have introduced delays or reduced accuracy in reporting.

In later semesters, the platform was extended to provide an integrated interface where students could interact with AI models and record their responses within the same environment [23]. This allowed submissions to be captured at the time of interaction and included improved validation mechanisms to ensure consistency between selected fields (e.g., phase, task type, and scope).

C. Dataset Description

The final dataset follows a unified schema after normalization. Each entry includes both objective and subjective attributes describing the AI-assisted interaction.

Objective attributes include:

- development phase (e.g., requirements, design, coding, testing),
- action type (e.g., code authoring, architecture decisions, infrastructure configuration),
- scope (e.g., backend, frontend, database design),
- AI model and tool used,
- time allocated.

Subjective attributes include:

- perceived quality of AI assistance,
- knowledge acquired,
- experience with the task,
- perceived threat level,
- general perception of AI tools (current and future), and
- estimated time saved.

In addition, each submission includes the entire user dialog with the LLM tool and optional notes describing the interaction.

D. Data Processing

Since the data collection process evolved over time, pre-processing steps were applied to ensure consistency across semesters.

First, all submissions were normalized into a common schema. This included mapping earlier ordered textual responses (e.g., “none”, “little”, and “much”) to corresponding values on the common 0–5 ordinal scale. Time-related fields were also transformed into normalized metrics to ensure comparability across entries. In particular, estimated time savings were expressed relative to total time (allocated plus saved) and mapped to the same 0–5 scale.

Categorical fields were also normalized by grouping semantically equivalent labels under a unified representation. For example, action types such as “*CI/CD*”, “*Dev-Ops*”, and “*dev-ops*” were grouped under the category “*ci/cd and devops*”. Similarly, entries such as “*container configuration*”, “*container structuring*”, “*network configuration*”, and “*vm configuration*” were mapped to “*infrastructure configuration*”.

A similar process was applied to the scope field, where variations such as “*ER diagram*”, “*Database Design (ER)*”, and “*database design*” were unified under “*database design*”. This normalization ensured that entries originating from different semesters and questionnaire versions could be analyzed consistently. The level of abstraction of categories reflects the structure of the original questionnaire and was preserved to maintain consistency within the data collection process.

Second, a validation process was applied to detect and remove inconsistent or logically invalid entries. For example, entries were considered invalid when combinations of fields violated logical constraints defined by the questionnaire (e.g., assigning a code design-related action to a requirements phase, or mismatched scope and action types). This was necessary for earlier datasets where validation constraints were less strict. In contrast, the final dataset, collected through the integrated

platform, was validated at submission time through built-in constraints and required no additional removal of invalid entries.

After validation, the final dataset consists of **8564** valid submissions, out of an initial **11326** collected entries. Table II summarizes the dataset before and after validation.

The mapping and validation rules are available in the project repository.

E. Ethical Considerations

Participation in data collection was part of the course requirements, and students were informed about the process. For the purposes of this study, all user identifiers were anonymized through hashing, ensuring that individual students cannot be identified in the dataset used for analysis.

F. Data Analysis

TABLE II
DATASET OVERVIEW BEFORE AND AFTER VALIDATION

Course	Collected	Valid
Software Engineering (softeng23b)	1542	950
Software as a Service (saas24a)	1297	705
Software Engineering (softeng24b)	3696	2134
Software as a Service (saas25a)	1711	1695
Software Engineering (softeng25b)	3080	3080
Total	11326	8564

The analysis focuses on understanding how students use LLMs during software development tasks. The variables examined in this paper come from the structured questionnaire fields associated with each submission; the dialog text between the LLM and the student was retained in the logs but was excluded from the analysis. To address the research questions, we examine distributions of AI usage across development phases and task types, as well as aggregate measures such as perceived quality, knowledge acquisition, time savings, and threat perception.

In addition, comparisons are performed across different levels of student experience and across time, in order to identify patterns in how LLM usage and perceptions evolve. In this analysis, students were classified as “experienced” in a given phase if their average experience score for that phase was at least 3 on the 0–5 scale, whereas those with an average score below 3 were classified as “inexperienced”. The analysis is conducted at the level of individual submissions.

IV. RESULTS

A. RQ1: On which software development tasks do students focus their use of AI tools?

Across all semesters, AI usage is highest in the *Coding* phase, followed by *Testing*, *Design*, and *Architecture*, as shown in Fig. 1. In contrast, the *Requirements Gathering* and *Requirements Specification* phases consistently show the lowest levels of AI usage, which indicates how students perceive the role of AI in software development.

This pattern appears in all datasets, although the exact distribution varies between semesters. For example, in *softeng23b*, most interactions occur in coding (42.0%) and testing (32.1%). In later semesters, such as *softeng25b*, usage is more evenly spread across coding (21.8%), testing (18.9%), design (19.2%), and architecture (18.6%). A similar trend is observed in *saas25a*, where design (23.0%) and architecture (22.1%) are also widely used, alongside coding (36.0%).

The distribution of AI usage across SDLC phases was also calculated separately for students classified as experienced (≥ 3) and inexperienced (< 3) in the corresponding phase. Experienced students focus more on coding (e.g., 52.7% in *saas24a*), while inexperienced students distribute their usage more across phases, including architecture, design, and deployment. For instance, in *saas24a*, inexperienced students report higher usage in deployment-related tasks such as Docker configuration (22.9%) compared to experienced students (6.9%), indicating that they possibly needed more assistance in executing such tasks.

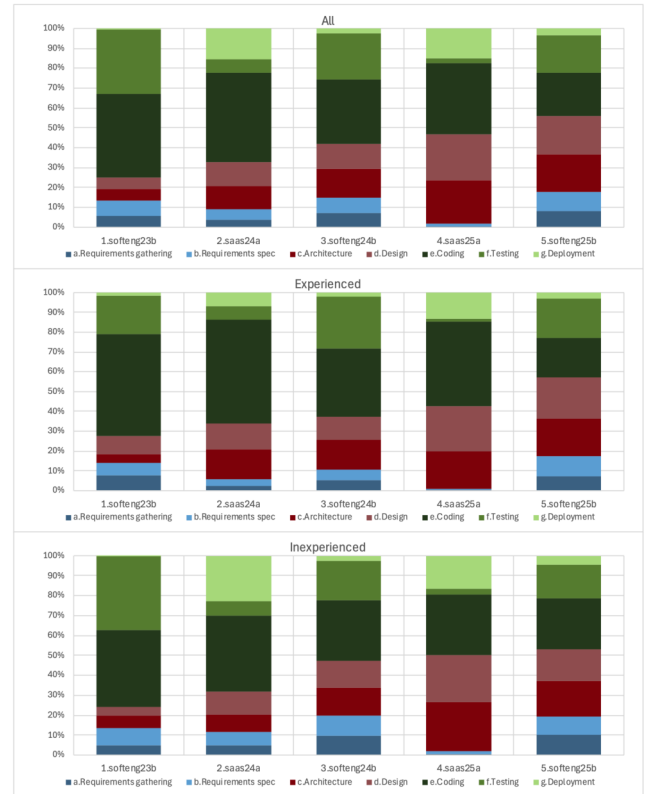


Fig. 1. Distribution of AI usage across development phases and experience levels

The activities within each phase indicate the nature of the tasks (Fig. 2). In coding, most interactions involve code authoring (66.5%) and code management (33.5%). In testing, interactions are mainly related to functional testing (49.7%) and unit testing (22.2%). In architecture and design, most interactions relate to architectural decisions (73.9%) and design decisions (67.1%), respectively. In requirements-related phases,

interactions focus primarily on problem understanding during requirements gathering (49.7%), while use case specification dominates in requirements specification (47.4%).

Overall, AI tools are used more in coding and testing, and less in requirements-related activities. Usage also differs by experience level, with experienced students focusing more on coding, while inexperienced students use AI more across different phases.

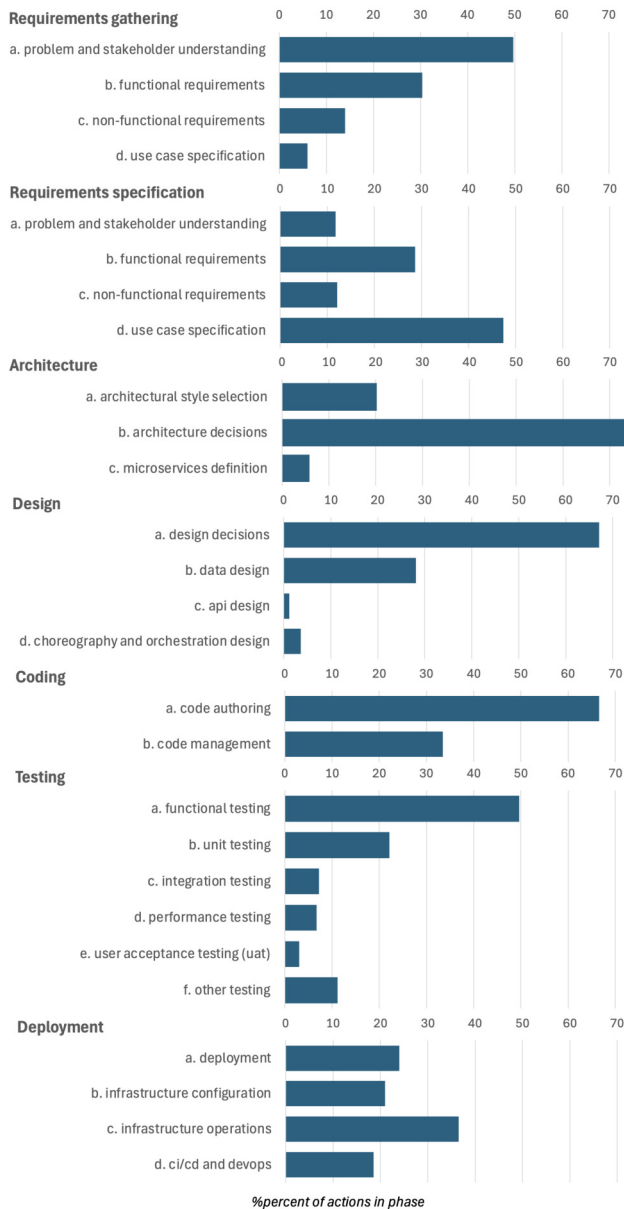


Fig. 2. Distribution of actions across development phases

B. RQ2: How do students rank the quality, knowledge gained, and time saved when using AI assistants?

Across all semesters, students generally provide positive evaluations of AI tools on the three examined subjective

metrics, namely quality of assistance, knowledge gained, and time saved, with scores consistently ranging between 3.0 and 3.7 on a 0–5 scale. However, there are clear differences in which of these metrics students rate most and least positively, as well as in how these evaluations differ between students reporting higher and lower levels of experience across the SDLC phases.

Fig. 3 shows, for each semester, the percentage of students who rated each of the three metrics with the highest score (left), the second-highest score (middle), and the lowest score (right). Quality is most often ranked higher across all semesters. The percentage of students ranking quality first ranges from 35.2% in *softeng23b* to 48.6% in *softeng24b*. In contrast, knowledge gained is least often ranked as the highest metric, with values between 15.2% and 27.5%. It is also most often ranked as the lowest metric (e.g., 54.7% in *softeng23b* and 45.7% in *saas25a*). Time saved shows a more mixed pattern, as many students rank it both as the highest and the lowest metric.

Additionally, differences between experienced and inexperienced students are observed across SDLC phases. To demonstrate this, for each semester, the average evaluations of AI tools across the three metrics were calculated for each SDLC phase, separately for students reporting high and low average experience within that phase. Then, evaluation ratios were computed by dividing the scores of experienced students by those of inexperienced ones in order to make differences more visible. These results are presented in Fig. 4 and Fig. 5 for the pre-development and development phases, respectively.

In the coding phase, experienced students report higher values for both quality and knowledge gained (ratios above 1) in most semesters, while time reduction ratios are usually below 1, meaning that experienced students estimate smaller time benefits compared to inexperienced students. A similar pattern appears in the testing phase, where knowledge ratios remain above 1, while time reduction varies across semesters.

In pre-development phases (requirements, architecture, and design), the differences between experienced and inexperienced students are less consistent. In earlier semesters, ratios are close to 1, while in later semesters (e.g., *softeng25b* and *saas25a*), both quality and knowledge ratios are above 1 in most phases. Time reduction ratios in these phases are often above 1, showing that experienced students sometimes report higher time benefits in these tasks.

Overall, students rate the quality of AI assistance higher than the knowledge gained, while time saved does not follow a clear pattern. Differences between experienced and inexperienced students are visible, especially in coding and testing, where experienced students report higher quality and knowledge gains.

C. RQ3: Do students perceive AI as a supportive tool or as a potential threat, both currently and in the future?

As mentioned earlier, across all semesters students report generally positive perceptions of AI tools (Fig. 6). Average ratings for current usefulness (*feeling_now*) range between 3.33

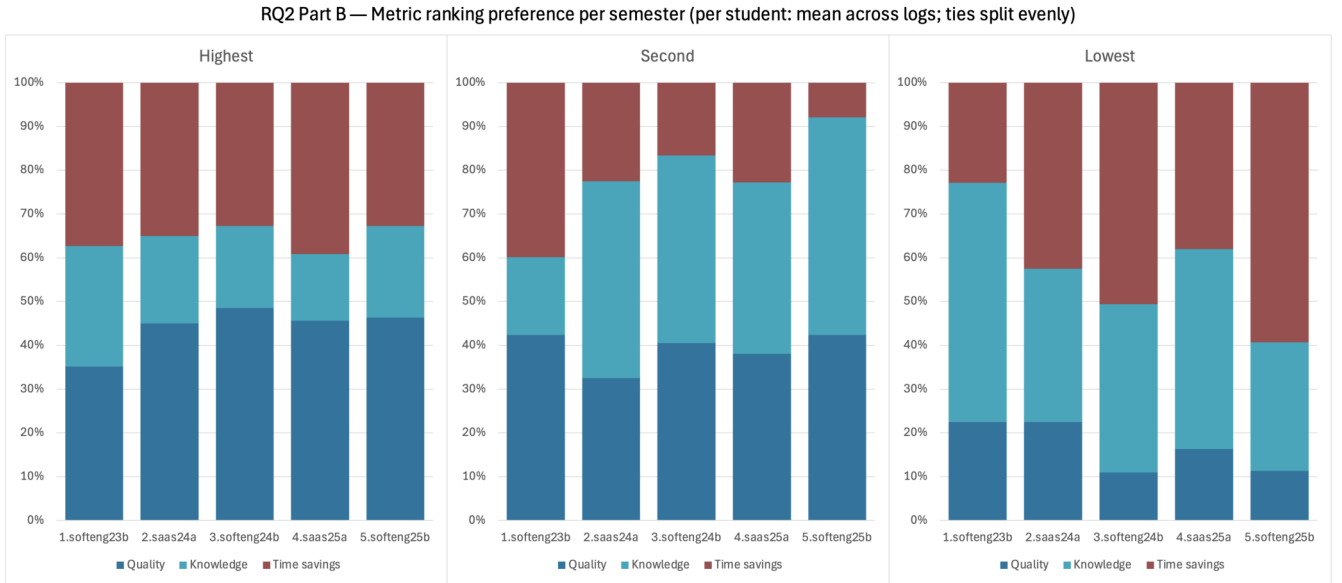


Fig. 3. Distribution of metric preference rankings across semesters

and 3.47, while perceived future usefulness (*feeling_future*) is similar or higher, ranging from 3.21 to 4.00.

In many semesters, future usefulness of AI is rated higher than current usefulness (e.g., 4.00 vs. 3.33 in *saas24a*, and 3.85 vs. 3.44 in *softeng24b*). A clear pattern is also observed in how evaluations of current and future usefulness evolve over time. Current ratings remain relatively stable across semesters, while future usefulness is generally rated higher in earlier semesters but gradually converges toward current usefulness. This suggests that, over time, students' expectations and actual experiences become closer.

Perceived threat remains low in all semesters, with values between 1.43 and 1.87 on a 0–5 scale. This shows that participating students do not see AI as a significant risk.

For context, both perceived quality and knowledge acquisition are above the midpoint of the scale in all semesters (around 3.0–3.7), which supports the overall positive view of AI tools.

Overall, students view AI as a supportive tool, with positive opinions about its current usefulness, slightly higher but gradually decreasing expectations for its future, and low levels of perceived threat.

V. DISCUSSION

A. AI Usage Across Development Phases

The results indicate that AI usage is concentrated in coding and testing phases, while earlier stages of the software development process, such as requirements gathering and specification, architecture and design, show lower usage. This suggests that students rely more on AI tools for tasks they are familiar with and feel easier to describe in prompts.

A possible reason is that requests to LLMs regarding coding and testing tasks can be expressed more easily and allow

students to directly use AI-generated outputs. In contrast, requirements, architecture, and design involve more abstract thinking and context, which, along with inherent AI weaknesses, may make AI less useful in these phases [24]. This is consistent with previous work that shows strong use of LLMs in coding tasks, but more limited use in higher-level Software Engineering activities [1], [2].

Differences between experienced and inexperienced students support this observation. Students tend to trust and use AI tools for coding tasks even when they have prior coding experience. However, in non-coding phases such as architecture and requirements analysis, students with relevant experience are more likely to rely on their own judgment and decisions. In contrast, students tend to use AI more broadly across most phases where they lack experience, reflecting greater expectations from AI support, as well as potential experimentation or uncertainty regarding the usefulness of AI outside the “comfort zone” of coding.

B. Perceived Benefits: Quality, Learning, and Time

Students consistently rate the quality of AI assistance higher than knowledge gained and time saved. This suggests that AI tools are mainly used to support task completion rather than learning. This observation reflects the behavior of students who did not have continuous access to AI tools throughout their studies, and may differ for students who are exposed to such tools from the beginning of their education.

The lower ranking of knowledge gained is important in an educational context. Even though students use AI frequently, they do not always see it as contributing to their learning of Software Engineering concepts. This may indicate a risk that students rely on AI outputs without fully understanding or evaluating them. Similar behavior has been reported in

previous studies, highlighting the risk of over-reliance on AI tools [21]. It may also be related to the fact that many Software Engineering tasks require prior knowledge and the integration of concepts from multiple areas, which may make it harder for students to learn effectively from AI-generated outputs.

Time saved shows a more mixed pattern, reflecting the inherently subjective nature of this metric. The assessment of time savings from AI is challenging even for experienced developers, and particularly so for those working on a task for the first time. In addition, the time needed to understand and adjust AI-generated outputs may reduce the overall benefit.

RQ2 - Pre-Dev: high_phase_experience + low_phase_experience | threshold = 3.0
(high ≥ / low <)

Quality	a.Requirements Gathering	0.93	0.74	1.74	2.31	1.81
	b.Requirements Spec	0.84	2.06	2.06	2.25	0.81
	c.Architecture	0.64	0.77	1.79	1.90	1.75
	d.Design	2.04	0.70	1.81	1.66	1.85
Knowledge	a.Requirements Gathering	2.01	0.42	0.80	0.60	1.87
	b.Requirements Spec	0.87	3.00	1.89	2.64	0.80
	c.Architecture	2.18	0.75	1.83	2.08	1.83
	d.Design	2.45	0.63	1.96	1.72	1.91
Time savings	a.Requirements Gathering	1.47	1.61	1.38	1.69	1.79
	b.Requirements Spec	0.71	0.74	1.45	0.59	1.47
	c.Architecture	0.79	1.63	0.77	1.41	1.49
	d.Design	0.57	0.80	1.47	1.51	1.52

1.softeng23b 2.saas24a 3.softeng24b 4.saas25a 5.softeng25b

Fig. 4. Experienced-to-inexperienced ratio across pre-development phases

Differences between students with higher and lower task experience further highlight the role of prior familiarity with the task itself. In pre-development phases, the results are less clear, which is logical, as not many students are expected to have strong familiarity with tasks such as requirements engineering or architecture in the context of these courses. However, in development phases, students who report higher experience in a given task also report higher quality and knowledge gains, especially in coding and testing. Overall, this suggests that prior task experience helps students make more effective use of AI.

A possible explanation is that students with higher task experience are better able to interpret, adapt, and extend AI-generated responses, for example by asking more specific questions or modifying the suggested solutions. In contrast, students with lower task experience reported lower assistance quality and lower knowledge gains. One possible interpretation is that limited prior experience makes it more difficult to formulate effective requests and evaluate or adapt the resulting outputs, potentially leading to more iterative trial-and-error interactions and less effective use of the assistance provided. This also suggests that using AI for tasks that are not well understood or cannot be properly evaluated is less effective.

C. Student Perceptions and Trust in AI

Students generally have a positive view of AI tools; however, in the context of this study, there is notably less enthusiasm about AI than one might expect. At the same time, there are slightly higher expectations for their future usefulness and, interestingly, a low perceived level of threat. This indicates that students see AI as a helpful tool rather than a replacement for their skills and future role.

The low perceived threat is notable and aligns with several public arguments regarding the future of software development jobs in the AI era. Students do not seem to consider AI as a risk, which may reflect how they use it as part of their workflow. At the same time, this may also suggest that students are not fully aware of potential long-term possible risks, such as over-reliance on AI.

RQ2 - Dev: high_phase_experience + low_phase_experience | threshold = 3.0 (high ≥ / low <)

Quality	a.Coding	0.77	1.74	1.69	1.62	1.77
	b.Testing	1.98	0.74	1.79	0.70	1.79
	c.Deployment	0.75	0.74	1.83	1.62	1.89
Knowledge	a.Coding	1.27	1.82	1.89	1.87	1.78
	b.Testing	2.33	1.83	1.72	1.93	1.86
	c.Deployment	0.90	0.69	1.98	1.83	1.90
Time savings	a.Coding	0.80	0.77	0.78	0.79	1.50
	b.Testing	1.50	1.42	0.76	0.79	1.45
	c.Deployment	0.69	1.52	1.49	0.77	0.78

1.softeng23b 2.saas24a 3.softeng24b 4.saas25a 5.softeng25b

Fig. 5. Experienced-to-inexperienced ratio across development phases

The difference between ratings of current and future usefulness suggests that students in the earlier semesters expected AI tools to improve substantially. Across later semesters, ratings of future usefulness decreased and moved closer to ratings of current usefulness. This suggests that students became less enthusiastic about the future potential of LLMs, reflecting a more realistic understanding of the capabilities and limitations of AI tools. These findings are in line with previous studies that report positive attitudes toward AI, along with some concerns about its limitations [10], [11].

D. Implications for Software Engineering Education

The results of this study have several implications for Software Engineering education.

First, the high use of AI in coding and testing shows that students use AI most frequently in the phases where they perceive it as useful. However, the lower use in requirements, architecture, and design suggests that students may either need guidance on how to use AI in these earlier phases, or feel that there is little that AI can achieve in such tasks. This aligns with complementary work, highlighting the limitations of AI tools when utilized in architecture-related tasks [24].

Second, the lower ratings for knowledge gained highlight the need for teaching approaches that encourage students to learn from AI, rather than use it only to complete tasks. Students appear to need more guidance on how to evaluate AI outputs and understand how to effectively use them.

Third, the differences between students with higher and lower task experience show that the reported value of AI depends in part on prior familiarity with the task being performed. This suggests that AI use in education should be considered together with students' existing task-level competence. Instructors may therefore need to provide different kinds of support depending on whether students are using AI for familiar tasks or for tasks they are still learning to perform manually.

AI perception & usage evolution over time (by semester)					
Quality	3.37	3.44	3.69	3.50	3.58
Feeling, now	3.47	3.33	3.44	3.35	3.34
Feeling, future	3.22	4.00	3.85	3.72	3.22
Knowledge acquired	3.10	3.12	3.38	3.06	3.47
Threat	1.49	1.87	1.73	1.44	1.43
	1.softeng23b (946)	2.saas24a (703)	3.softeng24b (2121)	4.saas25a (1688)	5.softeng25b (3005)
Mean score (0 - 5)					

Fig. 6. Evolution of student perceptions across semesters

Finally, the positive attitude toward AI shows that students are open to using these tools. This creates an opportunity to include AI more systematically in courses, while also addressing risks such as over-reliance and shallow learning.

VI. THREATS TO VALIDITY

A. Generalizability

The study is conducted within the context of two undergraduate Software Engineering courses at a single institution with a strong academic student background. As a result, the findings may not generalize to other educational settings, institutions, or student populations. In particular, differences in curriculum structure, project complexity, and student background may influence how AI tools are used. Furthermore, the use of project-based courses may lead to different usage patterns compared to more structured or assignment-based learning environments. The findings may also differ in other educational contexts, such as introductory courses or professional training environments.

B. Self-Reported Data Limitations

The dataset is based on self-reported logs submitted by students. As such, the data may be subject to biases, including inaccurate reporting, overestimation or underestimation of time savings, and subjective evaluation of AI assistance. Although submissions were part of the course requirements, students may still have provided incomplete or imprecise information. These factors may affect the reliability of subjective measures such as perceived quality, knowledge acquisition,

and threat perception. However, the structured questionnaire and its integration into the course workflow help mitigate some of these issues by encouraging consistent reporting.

C. Logging Constraints

Each submission captures a single AI-assisted interaction associated with a specific task, which may not fully represent the complete sequence of interactions that occurred during task completion. Students may have used AI tools without logging all interactions, or may have summarized multiple interactions within a single submission. Additionally, changes in the data collection platform across semesters may have influenced how students recorded their interactions, despite efforts to normalize and validate the dataset. Interactions involving the agentic development tools, such as Claude Code, Codex, or similar IDE-based assistants, could not be captured adequately through the existing logging process, which was designed for chat-based interactions. Capturing their use would require a different logging mechanism, for example, an extension or integration with the agentic tool itself.

D. Study Context

The use of AI tools was encouraged and partially integrated into the course grading rubric, which may have influenced student behavior. In particular, the requirement to submit a minimum total number of logs, although aiming to encourage the use of AI, may have affected how and when students chose to use AI tools. These contextual factors are inherent to many educational settings and should be considered when interpreting the results. Because the analysis was organized around the predefined research questions and questionnaire fields, the findings describe these selected dimensions of student AI use rather than every possible pattern contained in the logs.

VII. CONCLUSION

In this empirical study, we examined how students used AI tools in project-based Software Engineering courses. The results show that AI is used mainly in coding and testing tasks, while its use in earlier phases of development remains limited. Students consistently rate the quality of AI assistance higher than the knowledge gained, suggesting that they perceive AI as more useful for task completion than for learning. In addition, students with higher task experience report greater quality and knowledge gains, indicating that effective use of AI depends on prior familiarity with the task. This suggests that effective use of AI does not eliminate the need for users to develop their own understanding of the tasks they aim to accomplish.

Students have a positive view of AI and report low perceived risk. At the same time, considering AI as more useful for task completion than for learning suggests that students may use AI primarily to obtain working solutions even without fully understanding the underlying concepts. These indicate that students may underestimate not only the limitations of AI but also the potential consequences of excessive reliance on it, including weakening their own capacity for conceiving,

evaluating and refining AI solutions. In addition, the results show that expectations about the future usefulness of AI decrease in later semesters compared to earlier ones. The gap between students' ratings of AI's current and future impact also decreased across the observed semesters, suggesting a more grounded assessment of the future potential of LLMs.

These findings suggest that Software Engineering courses should address both the productive use of AI and the skills required to evaluate its outputs. Students should be encouraged to verify generated solutions, identify limitations, and seek clarification when they do not understand a response. This is particularly important when teaching concepts such as requirements engineering and software architecture, where the educational objective extends beyond producing an immediate solution. Building on these findings, further research should examine whether students who are introduced to LLM-based tools from the beginning of their studies develop different usage and evaluation practices from those who adopt them later. Future work should also examine how agentic development tools, such as Claude Code and Codex, affect students' productivity, learning, and ability to assess generated solutions.

ACKNOWLEDGMENT

The authors would like to thank all students who participated in the courses over the years and contributed to the dataset used in this study. In addition, we would like to thank Christodoulos Stylianides for conducting initial BI-focused analysis on early collected data.

REFERENCES

- [1] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," in *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FOSE)*, 2023, pp. 31–53.
- [2] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 8, Dec. 2024. [Online]. Available: <https://doi.org/10.1145/3695988>
- [3] J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly, and J. Prather, "The robots are coming: Exploring the implications of openai codex on introductory programming," in *Proceedings of the 24th Australasian Computing Education Conference*, ser. ACE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 10–19. [Online]. Available: <https://doi.org/10.1145/3511861.3511863>
- [4] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos, "Programming is hard - or at least it used to be: Educational opportunities and challenges of ai code generation," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 500–506. [Online]. Available: <https://doi.org/10.1145/3545945.3569759>
- [5] E. L. Ouh, B. K. S. Gan, K. Jin Shim, and S. Wlodkowski, "Chatgpt, can you generate solutions for my coding exercises? an evaluation on its effectiveness in an undergraduate java programming course," in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 54–60. [Online]. Available: <https://doi.org/10.1145/3587102.3588794>
- [6] P. Denny, V. Kumar, and N. Giacaman, "Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 1136–1142. [Online]. Available: <https://doi.org/10.1145/3545945.3569823>
- [7] M. Kazemitabaar, J. Chow, C. K. T. Ma, B. J. Ericson, D. Weintrop, and T. Grossman, "Studying the effect of ai code generators on supporting novice learners in introductory programming," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3544548.3580919>
- [8] J. Leinonen, P. Denny, S. MacNeil, S. Sarsa, S. Bernstein, J. Kim, A. Tran, and A. Hellas, "Comparing code explanations created by students and large language models," in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 124–130. [Online]. Available: <https://doi.org/10.1145/3587102.3588785>
- [9] J. Savelka, A. Agarwal, C. Bogart, Y. Song, and M. Sakr, "Can generative pre-trained transformers (gpt) pass assessments in higher education programming courses?" in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 117–123. [Online]. Available: <https://doi.org/10.1145/3587102.3588792>
- [10] A. Shoufan, "Exploring students' perceptions of chatgpt: Thematic analysis and follow-up survey," *IEEE Access*, vol. 11, pp. 38 805–38 818, 2023.
- [11] P. Haindl and G. Weinberger, "Students' experiences of using chatgpt in an undergraduate programming course," *IEEE Access*, vol. 12, pp. 43 519–43 529, 2024.
- [12] R. Budhiraja, I. Joshi, J. S. Challa, H. D. Akolekar, and D. Kumar, "it's not like jarvis, but it's pretty close!" - examining chatgpt's usage among undergraduate students in computer science," in *Proceedings of the 26th Australasian Computing Education Conference*, ser. ACE '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 124–133. [Online]. Available: <https://doi.org/10.1145/3636243.3636257>
- [13] C. Rahe and W. Maalej, "How do programming students use generative ai?" *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, Jun. 2025. [Online]. Available: <https://doi.org/10.1145/3715762>
- [14] R. Chen, S. Jiang, J. Shen, A. J. Moon, and L. Wei, "Examining the usage of generative ai models in student learning activities for software programming," in *2025 2nd IEEE/ACM International Conference on AI-powered Software (AIware)*, 2025, pp. 76–86.
- [15] H. Alquran and S. Banitaan, "Human-ai collaboration in programming education: Student perspectives on llm-based coding assistants," *Computers*, vol. 15, no. 3, 2026. [Online]. Available: <https://www.mdpi.com/2073-431X/15/3/154>
- [16] D. Stoyanova, S. Stoyanova-Petrova, S. Shotarova, S. Lyubomirov, and N. Mileva, "Chatgpt in programming education: An empirical study on its impact on student performance, creativity, and teamwork," *Education Sciences*, vol. 16, no. 1, 2026. [Online]. Available: <https://www.mdpi.com/2227-7102/16/1/19>
- [17] S. Rasnayaka, G. Wang, R. Shariffdeen, and G. N. Iyer, "An empirical study on usage and perceptions of llms in a software engineering project," in *Proceedings of the 1st International Workshop on Large Language Models for Code*, ser. LLM4Code '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 111–118. [Online]. Available: <https://doi.org/10.1145/3643795.3648379>
- [18] A. Naman, R. Shariffdeen, G. Wang, S. Rasnayaka, and G. N. Iyer, "Analysis of student-llm interaction in a software engineering project," in *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, 2025, pp. 112–119.
- [19] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: <https://doi.org/10.1145/3491101.3519665>
- [20] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, "Github copilot ai pair programmer: Asset or liability?" *J. Syst. Softw.*, vol. 203, no. C, Sep. 2023. [Online]. Available: <https://doi.org/10.1016/j.jss.2023.111734>

- [21] J. Prather, B. N. Reeves, P. Denny, B. A. Becker, J. Leinonen, A. Luxton-Reilly, G. Powell, J. Finnie-Ansley, and E. A. Santos, ““it’s weird that it knows what i want”: Usability and interactions with copilot for novice programmers,” *ACM Trans. Comput.-Hum. Interact.*, vol. 31, no. 1, Nov. 2023. [Online]. Available: <https://doi.org/10.1145/3617367>
- [22] I. Ozkaya, “Application of large language models to software engineering tasks: Opportunities, risks, and implications,” *IEEE Software*, vol. 40, no. 3, pp. 4–8, 2023.
- [23] S. Christodoulakis, “Design and deployment of a multi-ai model online wrapper with integrated data collection,” Master’s thesis, School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece, 2025. [Online]. Available: <http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/20048>
- [24] C. Hadjichristofi, M. Tsilimigkounakis, G. Sotiropoulos, and V. Vescoukis, “An empirical evaluation of large language models applying software architectural patterns,” *AI*, vol. 7, no. 6, 2026. [Online]. Available: <https://www.mdpi.com/2673-2688/7/6/195>