DevCoach: Supporting Students in Learning the Software Development Life Cycle at Scale with Generative Agents

Tianjia Wang
Virginia Tech
Blacksburg, Virginia, USA
wangt7@vt.edu

Ramaraja Ramanujan
Virginia Tech
Blacksburg, Virginia, USA
ramaraja@vt.edu

Yi Lu
Virginia Tech
Blacksburg, Virginia, USA
yilu38@vt.edu

Chenyu Mao
Virginia Tech
Blacksburg, Virginia, USA
mchenyu@vt.edu

Yan Chen
Virginia Tech
Blacksburg, Virginia, USA
ych@vt.edu

Chris Brown
Virginia Tech
Blacksburg, Virginia, USA
dcbrown@vt.edu

ABSTRACT

Supporting novice computer science students in learning the software development life cycle (SDLC) at scale is vital for ensuring the quality of future software systems. However, this presents unique challenges, including the need for effective interactive collaboration and access to diverse skill sets of members in the software development team. To address these problems, we present "DevCoach", an online system designed to support students learning the SDLC at scale by interacting with generative agents powered by large language models simulating members with different roles in a software development team. Our preliminary user study results reveal that DevCoach improves the experiences and outcomes for students, with regard to learning concepts in SDLC's "Plan and Design" and "Develop" phases. We aim to use our findings to enhance DevCoach to support the entire SDLC workflow by incorporating additional simulated roles and enabling students to choose their project topics. Future studies will be conducted in an online Software Engineering class at our institution, aiming to explore and inspire the development of intelligent systems that provide comprehensive SDLC learning experiences to students at scale.

CCS Concepts

• **Applied computing** → **E-learning**; *Interactive learning environments*; *Collaborative learning*.

KEYWORDS

Software Development Life Cycle, Generative AI, Software Engineering, Computer Science Education

ACM Reference Format:
Tianjia Wang, Ramaraja Ramanujan, Yi Lu, Chenyu Mao, Yan Chen, and Chris Brown. 2024. DevCoach: Supporting Students in Learning the Software Development Life Cycle at Scale with Generative Agents. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24), July 18–20, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3657604.3664663

L@S '24, July 18–20, 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-0633-2/24/07.
https://doi.org/10.1145/3657604.3664663

1 INTRODUCTION

The software development life cycle (SDLC), a systematic process used to plan, design, develop, test, deploy, and maintain software, is essential for producing high quality software applications [12]. However, the complex concepts and interactive team activities involved in the SDLC pose significant challenges for students to learn and instructors to teach at scale. Consequently, students frequently lack the necessary resources to understand the different development phases in SDLC and practice the crucial skills required for their future careers [1, 10].

Recent advancements in large language models (LLMs) and generative artificial intelligence (AI) have created significant opportunities for addressing the aforementioned challenges. Existing research demonstrates these models' capabilities in understanding human language, performing coding tasks, and simulating human behaviors [6, 9, 11, 13, 14]. For this project, we developed a system, DevCoach, that uses generative agents to simulate activities across various phases of the SDLC, thereby supporting large-scale student learning. These agents are designed to understand and respond to different aspects of the SDLC with personas to simulate various roles in a software development team. Students can seek guidance, practice skills, ask questions, and receive real-time feedback, fostering an interactive and engaging learning process. By providing a hands-on experience, users can grasp theoretical concepts and gain practical insights into how each decision and action influences the overall software development process.

Our initial work implements a system designed to support student learning of the "Plan and Design" and "Develop" phases of the SDLC. We limited the scope of the team project to developing a movie database and review system. To assess the usability and effectiveness of our system, we conducted a pilot study with five participants. The results show improvements in the participants' quiz scores after using our system, suggesting enhanced learning gains in understanding the SDLC. Also, the system received high ratings from participants for its usability and effectiveness in facilitating learning during these two SDLC phases. Our future plans, discussed in Section 5, aim to motivate scalable intelligent systems to promote more relevant learning experiences in SE education.

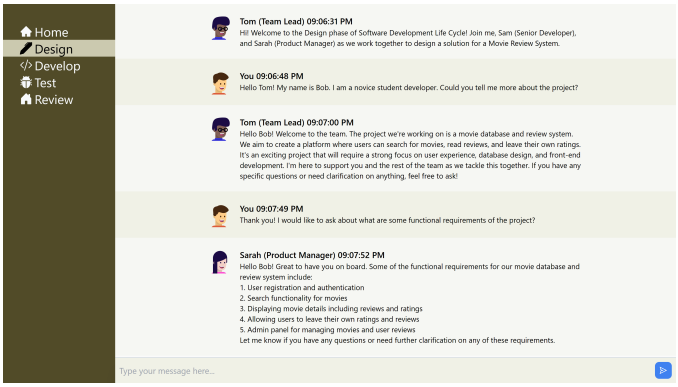

Figure 1: Plan and Design phase in DevCoach

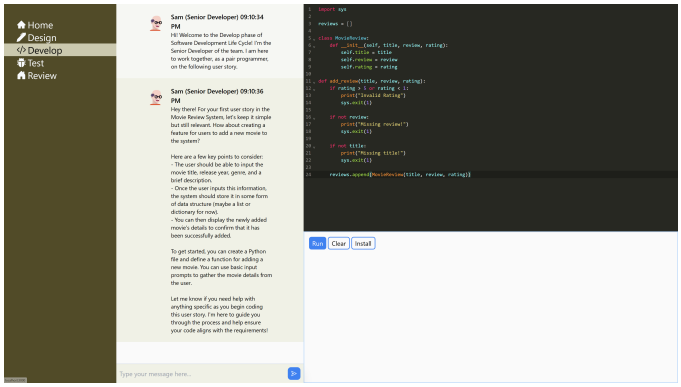

Figure 2: Develop phase in DevCoach

2 BACKGROUND AND RELATED WORK

SDLC. The SDLC is a structured process that guides the software development from an initial idea through to the maintenance of the completed application [12]. It typically includes several key phases: plan, design, develop, test, deploy, maintain, and review. These phases vary among different SDLC models. For instance, the Agile SDLC model refines the traditional SDLC by combining and condensing phases to enhance flexibility and responsiveness, focusing on these essential stages to accelerate product delivery and adapt quickly to changes [2, 8]. In our work, we adopted an agile approach, limiting the SDLC to four phases: "Plan and Design", "Develop", "Test", and "Review".

Generative AI in Education. Recent developments in LLM-powered generative AI have demonstrated capabilities in understanding natural language student inquiries [7], performing coding tasks [5, 17], and supporting student learning within computer science education [4, 9]. Generative AI has also been leveraged in broader educational contexts to enhance learning [3, 11]. The most closely related work to ours is CHATDEV [13], a framework employing generative agents that collaborate on tasks in the software development process via chat-based interactions. However, to our knowledge, no existing work explores human collaboration with generative agents for learning concepts of SDLC phases. Therefore, our work aims to build an effective learning environment with generative AI that can be deployed at scale for novice student developers to grasp workflows and concepts in SDLC.

3 SYSTEM DESIGN

DevCoach leverages LLM-powered generative agents in a full-stack web application to create realistic personas for team members in a software engineering (SE) team. The system allows users to engage with an interactive user interface and understand activities in the SDLC. For the initial implementation, our system will specifically target the "Plan and Design" and "Develop" phases of SDLC, as applied to creating a movie database and review platform.

The system features a front-end developed using ReactJS and a back-end powered by Node.js. The generative agents leverage the GPT-4 Turbo model provided by OpenAI's API, with prompt chain and memory functionalities supported by Langchain [16]. For the pilot study, we deployed the system locally.

3.1 Personas of Generative Agents

For the generative agents' personas, we leveraged three typical roles in SE teams that could be involved during the "Plan and Design" and "Develop" phases of SDLC: Team Lead, Product Manager, and Senior Developer. In order to create the personas, we went through various LinkedIn job descriptions and profiles to ascertain the ideal characteristics and profiles for the different roles. We then translated these characteristics into prompts template that we would send to OpenAI's GPT-4 Turbo API to get the response. The generative agents have a memory to retain information and respond to it later if required. The personas that we used to motivate our generative agents are as follows:

3.1.1 Team Lead Tom is a Team Lead with a decade of experience and a deep understanding of SDLC. Having held a significant leadership role for four years, he excels in guiding his development team through various project phases, ensuring they adhere to schedules, scopes, and budgets. His current focus is leading a project for a movie database and review system, where he aims to foster team coordination, motivation, and clarity on project milestones. His approach involves establishing a clear technical roadmap, setting precise requirements in line with project goals, and effectively navigating challenges. Collaborating closely with the other team members, Tom plays a pivotal role in maintaining productive discussions and steering the project toward success.

3.1.2 Product Manager Sarah is an adept Product Manager with a unique blend of skills from her Business Administration and Computer Science background, complemented by five years of experience in tech companies specializing in consumer-facing applications. Her role as a Certified master of SDLC underscores her software development management capabilities. She excels in gathering and prioritizing product and customer requirements, a task she performs in close collaboration with engineering, design, and support teams. Her responsibilities extend to ensuring that project goals are met and leading the strategic planning of product releases. Sarah's expertise lies in aligning technical development with customer needs and business objectives, making her a pivotal figure in the lifecycle of product development.

3.1.3 Senior Developer Sam is a highly skilled Senior Developer with a focused background in Computer Science. He brings eight years of valuable experience in software development, particularly in web applications and database design. Sam's broad expertise encompasses Java, Python, SQL, RESTful APIs, and microservices architecture. His primary responsibilities involve transforming product requirements into detailed technical specifications and spearheading the technical design process. Working closely with the Product Manager and Team Lead, he ensures proposed features' technical feasibility and scalability. Sam's current objective is to dive deep into the technical aspects of the project, including discussing data models, APIs, and other technical details, aiming to lay a robust technical foundation for the development team and shape the initial technical design of their current project.

3.2 Phases in Software Development Life Cycle

We focus on the "Plan and Design" and "Develop" phases of the SDLC for the initial implementation of DevCoach, and plan to incorporate additional phases including "Test" and "Review" in future work. We used the textbook Software Engineering: A Practitioner's Approach [12] to inform the tasks, activities, and interactions for each phase of the SDLC in our system.

3.2.1 Plan and Design Phase The "Plan and Design" phase involves outlining the software architecture, defining the overall system structure, designing the user interface and experience, and determining how different software components will interact to determine how the system will meet the specified requirements. In an agile workflow, the output of this phase typically includes detailed requirements, design documents, and system models, which serve as a blueprint for developers in the subsequent development phase. This phase ensures that the technical solution is feasible, efficient, and addresses the project's functional and non-functional requirements. For the design phase in DevCoach, the expected outcome is that users will interact with generative agents to come up with a set of user stories, or short product functional and non-functional requirements that center user needs in design processes [15].

As seen in Figure 1, when the user lands on the "Plan and Design" phase page, they would have a chat interface with a text box at the bottom where they could type their messages, and the area above it would display the history of messages. The Team Lead greets the user and starts the design discussion for a movie database and review system. The relevant persona would provide the appropriate response depending on the user's message. For example, if the user asks, "What are the functional requirements for the project?", then the Product Manager would respond as they are responsible for any discussion related to project requirements, user experience, market needs, and business objectives. Meanwhile, if a user asks about which programming language to use, the Senior Developer would respond as they are responsible for technical specifications, system design, data modeling, and technical feasibility of proposed features. The Team Lead would respond to messages concerning project timelines, resource allocation, team coordination, and adherence to technical standards and best practices. After discussion with the generative agents, the phase is deemed complete once the user comes up with a clear set of user stories of functional and non-functional requirements.

Table 1: DevCoach usage

	Task completion time		**Number of interactions**	
	Plan and Design	*Develop*	*Plan and Design*	*Develop*
P1	13	7	18	5
P2	21	14	27	8
P3	5	8	9	5
P4	25	44	11	8
P5	10	33	6	9
Avg.	14.8	21.2	15.5	7

Table 2: Quiz scores

	Pre-Quiz Score	**Post-Quiz Score**
P1	66.67% (4/6)	100% (6/6)
P2	16.67% (1/6)	83.33% (5/6)
P3	83.33% (5/6)	66.67% (4/6)
P4	50% (3/6)	83.33% (5/6)
P5	66.67% (4/6)	83.33% (5/6)
Avg.	56.67%	83.33%

3.2.2 Develop Phase The "Develop" phase in SDLC is where the actual construction of the software takes place, translating design documentation and specifications into source code. In this phase, developers write the code using the chosen programming languages, frameworks, and tools, guided by the architectural patterns and design principles established in the design phase. It requires careful coordination among the development team adherence to coding standards, and often involves regular reviews and updates to the codebase. The ultimate goal of this phase is to develop a functional software product that fulfills the defined requirements and is ready for testing before deployment. Effective implementation demands a balance between following the planned design and adapting to challenges or changes that arise during the coding process. The "Develop" phase in DevCoach is set up as a pair-programming simulation [18], where users collaboratively code with the Senior Developer persona. The senior developer assigns a user story from the "Plan and Design" phase, and then helps the user by providing hints to guide the user to the correct coding implementation of the functionalities described in the user story. For the scope of our system, the expected outcome is that the user provides the code that satisfies the user story and the acceptance criteria provided.

As seen in Figure 2, the "Develop" phase page has a chat window on the left, an editor that supports Python programming on the top right, and a terminal to run the code and install dependencies on the bottom right. The phase intends to simulate a scenario where the user writes code and gets their code reviewed and evaluated by another team member. The Senior Developer greets the user and then assigns a random user story along with its acceptance criteria for a movie database and review system. The user then begins to provide a coding implementation that satisfies the user story. Each time the user interacts with the Senior Developer agent, the current code in the editor will also be sent in the prompt for review. The user will be able to ask for help from the Senior Developer, who will provide helpful hints to guide the user in completing the user story. The phase is completed once the Senior Developer agent reviews the user's code and indicates that it satisfies the given user story.

Table 3: Usability scores

	Usability of the system	Helpfulness in understanding different software development phases	Quality of the communication and feedback from the generative agents	Overall learning experience with DevCoach
P1	5	4	5	5
P2	4	5	4	4
P3	4	4	4	4
P4	5	5	4	5
P5	4	3	3	3
Avg	4.4	4.2	4	4.2

4 PRELIMINARY EVALUATION AND RESULTS

Methods. To evaluate our implementation of DevCoach, we conducted a pilot user study with five student participants with skills in Python programming but inexperienced in SDLC. During the study sessions, we first obtained consent, collected demographic information, and documented their educational backgrounds. We then administered a pre-quiz to assess their baseline understanding of the SDLC, followed by a tutorial on utilizing our system for the tasks required in each phase. We monitored and recorded their deliverables, the time taken to complete tasks, and the number of interactions(numbers of communication via chat) with the generative agents necessary to complete the task. After the task, a post-quiz was conducted to reassess their understanding of the SDLC. The pre-quiz and post-quiz each contained four multiple-choice questions worth one point each, and a short-answer question worth two points. The quiz questions are designed to measure student understanding of fundamental software engineering concepts, the roles and responsibilities of different software team members, and activities and goals in SDLC phases. Finally, a post-survey was conducted to collect participants' opinions on DevCoach's usability, effectiveness in helping them understand SDLC, quality of communication and feedback, and their overall experience. Our study materials are available online.[1]

Results. Table 1 presents the completion times and number of interactions with agents required for participants to complete the task. All participants successfully completed the tasks. On average, it took 14.8 minutes to finish the task in the "Plan and Design" phase and 21.2 minutes in the "develop" phase. The users had an average of 15.5 interactions with the generative agents to formulate user stories during the "Plan and Design" phase, and an average of 7 interactions to complete the coding necessary to implement functionalities described in a user story. Table 2 presents the quantitative results of the pre-quiz and post-quiz. The average accuracy in answering the quiz questions improved from 55.67% to 83.33%, with 4 out of 5 participants showing an increase in their quiz scores after using DevCoach. The post-survey results in Table 3 indicate that DevCoach was highly rated by participants. Using a 5-point Likert Scale, users on average rated the system 4.4 for usability, 4.2 for the effectiveness of helping them learn SDLC concepts, 4.0 for quality of communication and feedback, and 4.2 for overall experience.

5 DISCUSSION AND FUTURE WORK

Our findings show DevCoach has the potential to enhance learning outcomes and experiences for students. The results from the pre-quiz and post-quiz demonstrate that the system enhanced participants' learning gains regarding the SDLC after using our system. We observed that participants achieved higher accuracy in answering questions related to the goals and activities of the "Plan and Design" and "Develop" SDLC phases, indicating increased familiarity with the fundamental concepts and processes involved. Our results highlight the potential of employing LLM-based generative agents to simulate support student learning at scale. By creating simulated members of the software development team with generative AI, students can gain experience interacting and communicating with members of different roles and backgrounds. The system has the potential to reduce labor requirements and costs associated with teaching students about SDLC. It also supports scalable deployment, making it feasible to implement in larger educational settings without significant increases in resource allocation.

Our long-term goal is to develop a system that supports all phases of the SDLC and allows students to select their own project topics. Furthermore, as generative AI now possesses the capability to interpret images, we propose revising the current "Plan and Design" phase into two distinct phases: "Plan" and "Design". The "Plan" phase will focus on gathering project requirements and drafting user stories, while the "Design" phase will support students in learning how to create various design diagrams, such as class diagrams and system flow diagrams. In the "Testing" phase, we aim to design activities that teach students various software testing methods and explore how quality assurance impacts the quality of software projects. During the "Review" phase, we intend for both users and generative agents to identify existing issues and devise a plan for subsequent iterations. We also plan to add more roles with generative agents in different phases, such as a stakeholder in the "Plan" phase and quality assurance in the "Test" phase. Once all phases are developed, we will incorporate the concept of agile development into the workflow, enabling students to experience flexible requirement changes across multiple iterations of the SDLC. We plan to conduct a formal user study by implementing this system in an online SE course offered at our institution to observe its capabilities for teaching SDLC concepts at scale.

[1] https://anonymous.4open.science/r/DevCoachStudy-777F/

References

[1] Deniz Akdur. 2022. Analysis of software engineering skills gap in the industry. *ACM Transactions on Computing Education* 23, 1 (2022), 1–28.

[2] Samar Al-Saqqa, Samer Sawalha, and Hiba AbdelNabi. 2020. Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies* 14, 11 (2020).

[3] David Baidoo-Anu and Leticia Owusu Ansah. 2023. Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning. *Journal of AI* 7, 1 (2023), 52–62.

[4] Charis Charitsis, Chris Piech, and John C Mitchell. 2022. Using nlp to quantify program decomposition in cs1. In *Proceedings of the Ninth ACM Conference on Learning@ Scale*. 113–120.

[5] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference*. 10–19.

[6] Muhammad Imran and Norah Almusharraf. 2023. Analyzing the role of ChatGPT as a writing assistant at higher education level: A systematic review of the literature. *Contemporary Educational Technology* 15, 4 (2023), ep464.

[7] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J Malan. 2024. Teaching CS50 with AI: leveraging generative artificial intelligence in computer science education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 750–756.

[8] Peter Manhart and Kurt Schneider. 2004. Breaking the ice for agile development of embedded software: An industry experience report. In *Proceedings. 26th International Conference on Software Engineering*. IEEE, 378–386.

[9] Julia M Markel, Steven G Opferman, James A Landay, and Chris Piech. 2023. GPTeach: Interactive TA Training with GPT Based Students. (2023).

[10] Damla Oguz and Kaya Oguz. 2019. Perspectives on the Gap Between the Software Industry and the Software Engineering Education. *IEEE Access* 7 (2019), 117527–117543. https://doi.org/10.1109/ACCESS.2019.2936660

[11] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–22.

[12] Roger S Pressman. 2005. *Software engineering: a practitioner's approach*. Palgrave macmillan.

[13] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924* (2023).

[14] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 27–43.

[15] Srikar Suresh. 2023. Understanding user stories in UX design. *LogRocket Blog* (2023). https://blog.logrocket.com/ux-design/understanding-user-stories/.

[16] Oguzhan Topsakal and Tahir Cetin Akinci. 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, Vol. 1. 1050–1056.

[17] Tianjia Wang, Daniel Vargas Díaz, Chris Brown, and Yan Chen. 2023. Exploring the Role of AI Assistants in Computer Science Education: Methods, Implications, and Instructor Perspectives. In *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 92–102.

[18] Laurie Williams. 2001. Integrating pair programming into a software development process. In *Proceedings 14th Conference on Software Engineering Education and Training.'In search of a software engineering profession'(Cat. No. PR01059)*. IEEE, 27–36.