Using Generative AI in Software Design Education

Victoria Jackson
University of Southampton
Southampton, UK
v.jackson@soton.ac.uk

Susannah Liu
University of California, Irvine
Irvine, USA
susannl5@uci.edu

André van der Hoek
University of California, Irvine
Irvine, USA
andre@ics.uci.edu

Abstract

This paper provides an experience report on introducing GenAI into an undergraduate software design class. Students were required to use GenAI (in the form of ChatGPT) to help complete a team-based assignment in which they were tasked with designing the core of the solution for how an educational traffic simulator could keep track of the elements in the simulation as well as how to advance the simulation over time. We collected and analyzed both the ChatGPT conversation logs and the teams' reflections on using ChatGPT for the assignment. We report on how the teams used ChatGPT, reflect on how they felt ChatGPT helped them and where they felt it had shortcomings, and identify several key lessons for educators in how to deploy GenAI in a software design class.

CCS Concepts

• **Software and its engineering → Software design engineering**; • **Applied computing → Computer-assisted instruction**.

Keywords

Software Design, Generative AI, Software Engineering Education

ACM Reference Format:
Victoria Jackson, Susannah Liu, and André van der Hoek. 2026. Using Generative AI in Software Design Education. In *3rd International Workshop on Designing Software (Designing '26) (Designing '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3786152.3788589

1 Introduction

The advent of freely available off-the-shelf Generative AI (GenAI) tools such as ChatGPT and GitHub Copilot is having a significant impact on software development. Students, too, have started to use GenAI tools for their own purposes, including assistance in their course work. While some advocate such use should be prohibited (e.g., [23]), we feel that for students to learn to use GenAI well is essential in preparing them for working in the software profession. Our view aligns with many educators who are also determining the best way to incorporate GenAI tools into their curriculum and for what purpose [22]. Some examples include using GenAI to provide automated code feedback to students [27], explaining code [25], and helping students learn code linting [2].

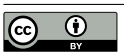
To date, integrating GenAI into software design education has not yet received a lot of attention. This is perhaps unsurprising, given that there is limited research in the first place on how professionals use GenAI in software design [30], which leaves little for educators to draw upon. Nonetheless, several reports on using GenAI in software design education exist, with a primary focus on how it can assist in software modeling (e.g., [11]).

To contribute to the ongoing discussion of how GenAI can be brought into undergraduate software design work, we provide an experience report where 179 students grouped into 36 teams used GenAI to help complete a design studio assignment. The assignment required them to design the core of an educational traffic light simulator, with a specific focus on providing a solution to capture the state of the simulation (e.g., cars, lights, roads, lanes) and a commensurate solution for how to advance the state of the simulation. As part of the assignment, student teams were required to hold a minimum of ten conversations with ChatGPT, through which they were encouraged to explore different aspects of the design problem and solution they were proposing. Teams submitted the conversations as part of the assignment, and were required to write a reflection on their experience that they also submitted. Below, we report on how the students brought GenAI into their design work, what they felt worked well and what not, and from this derive some general lessons learned that others may be able to leverage when they choose to incorporate GenAI in their software design class.

2 Related Work

While Large Language Models, and GenAI more broadly, have been well studied in recent years [16], with many studies focused on coding [18], there is little research that considers how GenAI can aid software design [18]. One area considered is the use of GenAI to generate UML models [28], observing that, while ChatGPT-generated UML models were generally syntactically correct, they were not always semantically correct [10], missed elements such as relationships [7], and required the diagrams to be manually checked for correctness and completeness [17]. Another study introduces a novel, ChatGPT-based tool that shows promise for improving the quality of use cases [13]. Mixed opinions on the value of ChatGPT-generated architectures were expressed in yet another study examining its capabilities to generate C4 architecture diagrams [20].

Beyond software engineering, design is essential in areas like user experience (UX) and product design. UX designers perceive GenAI as helpful for ideation and at the beginning stages of design [31], and helps to automate monotonous work [24]. However, concerns exist about the accuracy of GenAI-generated content due to GenAI not fully understanding the context of the design [31] or its lack of empathy [24]. Thus, humans must remain in charge to check the results and to provide empathy and user involvement [24, 31]. Within product design, GenAI aids ideation processes by generating ideas quickly [33] and augmenting human creativity [8].



Designing '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2385-8/2026/04
https://doi.org/10.1145/3786152.3788589

SE educators recognize that software engineering education needs to evolve to incorporate GenAI [22]. Proposals to change course content include the teaching of higher-level conceptual understanding [22], prompt engineering to encourage helpful responses [14], how to critically review and validate the responses to prompts [22], along with a focus on ethics [26]. Various ways in which GenAI can assist students in their learning include providing personalized feedback [25] and interacting with students via a Socratic approach [1] to help them solve problems. Notably, much research on GenAI in software engineering education has focused on using GenAI to aid students in learning to program [6]. Benefits include improvements to computational thinking skills [4], programming self-efficacy [35], and increased performance in code-authoring tasks compared to students who did not use GenAI [21]. While recognizing the benefits, educators have raised concerns about the use of GenAI in the classroom including an over-reliance on the use of GenAI, and whether it is appropriate for beginners to use GenAI [6], with some evidence that learners with some programming experience may benefit more from GenAI for code generation tasks [21].

Little research exists about using GenAI to aid students in learning software design. One paper [11] notes that students who used ChatGPT to produce UML class diagrams in formative assessments performed better in the final summative assessment than those who did not use ChatGPT. Students also observed that, while effort was required to use ChatGPT to generate a complete solution, it helped provide a preliminary solution to be manually improved later. Some work takes a step back, questioning how to best deliver software design education. The philosophy and pedagogical approaches to designing two new software design courses forming part of a new degree program was shared in [3]. Both courses were designed to encourage design-minded thinking and to balance theory and practice. A different approach incorporated design studios to emphasize hands-on, in-class activities to aid students learn aspects of software engineering, often using reflective techniques [5].

3 Setting

Rather than run an experiment to test what differences GenAI may make to learning software design, we decided to integrate GenAI into the classroom and take a more exploratory approach, whereby students would be required to use GenAI for a single assignment and reflect on their experience. The students were asked to use GenAI partway through the course rather than at the start. This delayed introduction allowed the students to receive instruction on foundational expert software design practices (shown in Table 1 and discussed further below) and practice design without GenAI.

The course is a 10-week, mandatory upper-division undergraduate course taught at the University of California, Irvine. Each week, there are two lectures (each 80 minutes) and a compulsory discussion section where students can seek help from the teaching assistants on the course assignments. The course is one of six core requirements for the Software Engineering B.S. degree alongside other courses such as Requirements Analysis and Engineering and Human-Computer Interaction (HCI). It makes extensive use of expert practices [29] used by professional designers to help students learn about strategies beneficial to software design, along with other design topics such as UML modeling. Among the expert design practices taught are focusing on the essence of design problems, generating alternatives to solving a problem, and reusing partial solutions from elsewhere as applicable.

Table 1: Expert Practices From [29] Taught Prior To The GenAI Assignment. The Observed column notes whether students seemed to apply the practice in the GenAI required assignment. Key: Y = Practice was applied, - = no evidence or cannot tell, N/A = not applicable to the assignment.

Expert Practice	Observed
Focus on the essence	Y
Address knowledge deficiencies	Y
Prefer solutions that they know work	-
Generate alternatives	Y
Know design is not done until the code is delivered and running	N/A
Dialog between problem and solution	Y
Solve simpler problems first	Y
Draw the problem as much as they draw the solution	-
Go as deep as needed	-
Simulate continually	Y
Think about what they are not designing	-

Collaboration is also a key learning point, so the course is structured such that topics are introduced in a lecture before students work in small teams to practice the topic within the lecture (e.g., identifying the stakeholders for an example design problem, produce a UML class model representing the design essence) and in completing assignments. In keeping with the practical ethos of the course, the coursework consists of three design studio assignments, each worth 20% of the final grade. Each design studio asks the students to work in small teams of five to solve a design problem, thus allowing students to practice the design techniques learned in class. The use of GenAI was introduced in the second design studio, the results of which form the basis for the observations in this paper. In total, 179 students took the course, grouped into 36 teams for each design studio.

The design studio in which GenAI was introduced consisted of two parts. Part 1 asked the students to design an educational traffic flow simulator to help civil engineering students understand traffic signal timing. The provided requirements described how students would use the simulator to create a visual map of the area, adjust the behavior of the traffic lights at intersections, and control the flow of traffic, including the density. Simplifying constraints were provided, such as all cars traveling at the same speed and no pedestrian crossings. The assignment required the completion of three design artifacts: (i) a list of stakeholders, (ii) a supporting data structure for tracking the state of the simulation at a given point in time in the form of a UML class diagram, and (iii) an algorithm in pseudocode that advances the state of the simulator "one tick". The assignment asked the students to use the expert practices introduced in class, such as solving simpler problems first, generating alternatives (thus encouraging ideation), and focusing on the essence to help them address the design problem (see Table 1).

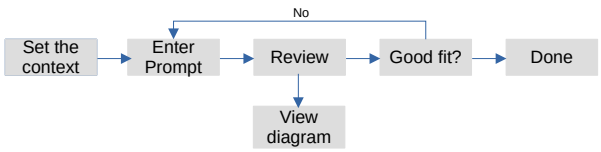


Figure 1: Guidance for generating a design artifact in GenAI

The students were mandated to use GenAI (ChatGPT) by engaging in at least ten conversations with ChatGPT, and each conversation had to be different in purpose. At least two team members were requested to collaboratively write each prompt, review the response, and participate in the GenAI conversation. Students had two weeks to complete the first part of Design Studio 2.

Part 2 asked the students to extend their initial design to incorporate changed requirements. They had one week for this part and were required to submit an amended design document with the revised UML class diagram and updated pseudocode. Using GenAI was optional for this part, though all teams had to submit a reflection after completing parts 1 and 2 on how they felt GenAI helped and hindered them in the design process. All ChatGPT conversations had to be included when submitting parts 1 and 2.

Before the design studio, students were first introduced to UML class diagrams in lecture. Among other topics, they were shown how to produce a starting point for their UML class diagram by identifying key pieces of information in a design prompt, such as nouns and verbs. Principles such as separation of concerns, KISS (Keep It Simple, Stupid), and YAGNI (You Ain't Going to Need It) were also introduced to help the students focus on the most critical part of designing. Students were given time in class to practice specifying UML diagrams, with the instructor reflecting on their attempts. In a subsequent lecture, the students were introduced to a GenAI tool (specifically OpenAI ChatGPT 3.5, as this was the most popular text-based GenAI tool at the time of the course and free to use) and how it could be used to generate a UML class diagram and pseudocode. The lecture was purposefully designed to show how GenAI could be used, but not to be prescriptive in how it should be used, as one of the goals of introducing GenAI in the course was to see how the students would use it creatively. That said, the lecture did introduce the iterative process shown in Figure 1, three high-level prompt engineering strategies (*brute force* – copy/paste the entire design problem into a prompt; *start small and grow* – provide a snippet of the problem and generate a small class model to be refined with subsequent prompts; and *focus on the problem's goal* – share information about the overarching goal of the problem and guide ChatGPT to the problem under consideration), and the usage of ChatGPT in tandem with the publicly available and open-source PlantUML server [32] to visualize generated UML class diagrams. The suggested iterative process of having a conversation with ChatGPT (Figure 1) emphasized the need to first provide some context before iterating through sequential steps of prompting, reviewing the response, and entering a further prompt to help guide ChatGPT into producing an acceptable design artifact.

The training material and the assignment in which ChatGPT was required to be used are available as supplementary material [19].

4 Data Analysis

Despite this being an experience report, we engaged in some data analysis to prepare for presenting our findings. The data used in this report stems from the Fall 2023 iteration of the class. Before the analysis, the assignments were anonymized by removing the student names from the submitted assignments and assigning a random number to each team, hereafter identified as T_x where x is a random number between 100 and 200. All 36 teams submitted conversation logs; 30 provided reflections.

Firstly, thematic analysis [9] was performed on the reflections. One researcher, as guided by a second researcher, read each reflection carefully and authored a short memo highlighting interesting points contained in the reflection. These interesting points were subsequently placed onto a FigJam board. Once all points were on the board, they were iteratively organized into themes. Secondly, the same researcher analyzed the prompts and responses in the conversation logs. In total, 360 conversations were analyzed (36 teams, 10 conversations each). The analysis sought to determine the strategies used by teams when conversing with GenAI, the amount of context put into the prompts, and the types of questions asked. This analysis was performed incrementally, starting with a few conversations at a time, to enable the second researcher to review the observations and further refine the analysis approach together with the first researcher. The observations were then clustered on the same FigJam board containing the themes from the reflections.

Finally, in reviewing the collated observations, it was evident that some of the expert practices taught in class were visible in the teams' reflections and conversation logs. For example, various teams talked about "essence" or "alternatives" in their reflections, and some teams were asking questions of GenAI that implied they were "seeking knowledge". The conversations were therefore further analyzed to determine if there was evidence of the expert practices being used.

5 Observations

We report our observations along four questions:

- For what purpose did the students engage with GenAI?[1]
- How did GenAI's replies contribute to their final designs?
- What were issues encountered in using GenAI?
- What overarching feelings did the students have on using GenAI for software design?

5.1 Engaging with ChatGPT

The majority of teams used ChatGPT to aid with their UML class diagram and the algorithm necessary to address the design studio, with fewer using it to explore the stakeholders (which the design studio also required). How they used it differed, with the usages described below exemplary of the broader use across the teams.

5.1.1 Ideation. Many teams used ChatGPT to generate ideas, for example by generating a list of stakeholders or asking for a starting set of classes for their UML diagram. Indeed, the ability of ChatGPT to generate ideas was commonly praised by the teams:

> "*We believe that the place that AI really shines is getting initial ideas and the brainstorming phase of design.*

[1] As all the teams used ChatGPT, we will use the term ChatGPT rather than GenAI in the remainder of this section.

> *This is because ChatGPT has so much knowledge and data to pull from, that when asking it a broad question, it can return many potential answers and ideas. This allows our group to explore many different ideas before deciding on the best one to use for our specific design.* (T_{155})"

Teams benefited from the ideas generated by ChatGPT, as they noted it was able to "*Quickly generate a range of ideas*" (T_{130}) and that it provided a "*Diverse range of perspectives*" (T_{141}) and "*Alternate approaches and methods*" (T_{150}). Moreover, it was helpful in identifying potential edge cases that some teams had not considered.

The teams had mixed opinions on the novelty of the ideas generated. Sometimes ChatGPT proposed ideas that the team had not considered, "*We ended up using some of the ideas it suggested that we had not thought of.*" (T_{150}), but other times it generated ideas that were the same as ones already identified by the team. This may reflect the choice of when to engage ChatGPT in brainstorming: either first to contribute ideas that the team then builds upon or the team first generates some ideas and then asks GenAI to contribute.

5.1.2 Knowledge Seeking. Although prior studies have noted the extensive use of GenAI tools as conversational search engines [12], we saw little evidence of this form of usage among the teams. This could be due to the design prompt providing extensive details about the traffic simulator, or it could be because the teams were strongly focused on using ChatGPT to help them with ideas and producing starting points. Nonetheless, a few teams did occasionally seek additional knowledge. For example, one team wanted to understand sensors better:

T_{105}: "How do sensors at traffic lights work?"
ChatGPT: *Describes different sensors such as inductive loop sensors, video cameras, and radar*

5.1.3 Decision Making. Some teams used ChatGPT to aid their decision making. We observed two approaches, varying by ownership. In one approach (illustrated below), the team would ask ChatGPT for various suggestions first and subsequently ask it to take the decision as to which one to select, effectively outsourcing the choice. Notably, ChatGPT always refused to make the decision, instead providing a list of advantages and disadvantages about the various options to better inform a team about potential trade-offs.

T_{177}: "You are an expert software designer tasked with creating a traffic flow simulation program for students to use in a civil engineering class. Give 5 different ways the program can decide which direction each car will go (straight, left, or right)?"
ChatGPT: *Provides five suggestions*
T_{177}: "Would you have the user choose a destination for each car or have the program choose?"

In the second approach, the team made the decision themselves, but still asked ChatGPT to offer one or more alternative viewpoints:

> "*When we were considering two options we realized that we could ask ChatGPT about the situation we were trying to solve and ask it for its solution, comparing it to ours. This leads to another valuable opinion and helps the group overcome hurdles when we were stuck on how and where to proceed.*" (T_{155})

5.1.4 Validation. Some teams used ChatGPT to validate their design by using it as a "*Second-pair of eyes*" (T_{146}). That is, they generated the design themselves, provided this design along with information about the design problem to ChatGPT, and then asked it to validate the design against the problem.

T_{112}: "For the prompt below, we currently have the classes for the UML Diagram, which are [Listed 12 classes]. Is there any other classes we are forgetting? Here is the prompt: [Pasted in the entire design prompt]."
ChatGPT: *Provides nine additional classes, each with a description*

Teams also used ChatGPT to check consistency between artifacts:

> "*We asked the AI to compare our UML diagram and pseudocode to the prompt to check if our solution may have been missing any components. We also asked it to evaluate the UML diagram and pseudocode ... to ensure that the two were compatible.*" (T_{176})

Using GenAI to check their designs has the potential to help teams to improve their design quality both in terms of: (1) ensuring requirements were satisfied as GenAI was able to identify omissions, and (2) consistency by recommending naming conventions, meaningful functions, and variable names.

5.1.5 Other uses. There were some unexpected uses of GenAI in the assignment. One team used ChatGPT to test the accuracy of their UML model by asking it to "run" the simulation and identify any missing methods. In its answer, ChatGPT identified missing methods, which the team promptly asked it to incorporate.

Reflection is an important aspect of design [29]. Some teams asked ChatGPT to reflect on the design they had worked on thus far—whether with or without ChatGPT—as shown below. This is also another case where the team delegates the decision about what to change to ChatGPT.

Team had previously asked ChatGPT to generate the pseudocode
T_{183}: "Can you talk about the weakness of your algorithm?"
ChatGPT: *Provides weaknesses*
T_{183}: "And also tell me about some of the good points of the algorithm as well"
ChatGPT: *Provides some good points*
T_{183}: "As considering the weakness of the algorithm, can you make more improvements?"
ChatGPT: *Provides revised pseudocode*

Interestingly, when it comes to the expert software design behaviors (Table 1), we mostly observed them playing out implicitly in the prompts to ChatGPT. Even for clear cases, such as ideation to generate alternatives, the prompts being used did not explicitly mention the behavior or objective, which could perhaps have been helpful in setting context for the GenAI to take into consideration. Only on a few occasions did teams explicitly mention the behavior, mostly when they asked ChatGPT to focus on generating a UML diagram for the essence of the design problem they were addressing:

T_{130}: "Especially for UML diagram, it is important not to include extraneous attributes and methods that distract from the essence. Therefore, can you point out any attributes that you mentioned previously that may not be as necessary and can instead be classfied under another class"
ChatGPT: *Provides an updated, simplified design*

5.2 Incorporating Responses into the Design

Building on the earlier discussion of ideation, we identified three distinct models through which teams incorporated ChatGPT's responses into their design process.

5.2.1 Starting Point. Some teams experienced difficulties in getting started and thus appreciated that ChatGPT could generate a beginning for them to develop further, "*It definitely helped us in getting something down on paper right off the bat. This saved us a lot of time since it is often hard to get started in the beginning.*" (T_{199}) Be it as a list of stakeholders, a UML class diagram, or some pseudocode for the algorithm required, in these cases incorporating the response from the LLM into the design was trivial: since the teams did not yet have an artifact of this kind, it adopted whatever resulted at the end of their conversation (since sometimes they iterated some with ChatGPT to generate the final starting point) and then worked from there to improve it.

For textual artifacts, the team would typically copy-and-paste the generated set of stakeholders or initial version of the pseudocode in a text editor. UML class diagrams generated by ChatGPT are also represented as text in its replies, so teams usually would manually "redraw" the diagram in a tool such as LucidChart or Draw.io, "*We moved forward with expanding from what ChatGPT had and what we came up with as well to create our UML diagram.*"(T_{133}), despite knowing that tools such as PlantUML [32] can do this automatically. Sometimes, they copied the diagram from ChatGPT verbatim before they started refining it; other times, refinement started during the copying process.

5.2.2 Cherry Pick. Such cherry picking would also take place when teams asked ChatGPT for improvements to the designs they had made thus far. Teams would engage in a conversation, at the end of which they would update the design that they are maintaining elsewhere manually with just the pieces they wanted, ignoring the rest. For a few teams, however, choosing the parts to keep and not keep was more difficult, because they were attempting to refine a UML class diagram or pseudocode within ChatGPT. In such cases, they attempted to steer ChatGPT with details instructions, as in the example below.

Team had already asked ChatGPT to generate the class model
T_{101}: "What about students as a class?"
ChatGPT: *Updates the class model with students*
T_{101}: "we aren't gonna have any roads that start within the middle of the map. All roads start and end on the edge of the map"
ChatGPT: *Updates the class model*
T_{101}: "I think that road should be separated from the intersection class, intersection can be a child right?"
ChatGPT: *Updates the class model*

This did not always produce the right result, sometimes incorporating unwanted additions and other times not doing enough. This let to further detailed back-and-forth with ChatGPT, with some teams eventually giving up and doing what they wanted by hand.

5.2.3 Reject. At times, teams were unhappy with the response, so ended up doing the work themselves and ignoring the suggestions by ChatGPT, "*How a tick should be measured within the simulation was hard for the model to understand and implement in a logical manner, so it was designed manually.*" (T_{143}) Even in such cases, at times some of the suggestions made by ChatGPT would shine through in the discussions the teams held, though direct traceability is more difficult to establish.

5.3 Issues with ChatGPT's Responses

Many teams commented on issues encountered when reviewing ChatGPT's responses. These issues ranged from somewhat irksome ones (e.g., syntax errors) to more fundamental ones where the teams questioned the value and usefulness of the response (e.g., undue complexity, superficiality). These issues are not necessarily unique to our setting, but bear bringing out for they impact what instructors may need to educate students about in terms of GenAI limitations in this context.

5.3.1 Syntax Errors. More an annoyance than anything else, but teams observed syntax errors in generated PlantUML code and also the pseudo-code. This led to additional work for the teams to correct the generated code, "*Sometimes it will generate things with index errors and you need to debug the PlantUML codes.*" (T_{112})

5.3.2 Reliability. Teams noted that ChatGPT could be unreliable. Sometimes ChatGPT would forget things discussed earlier in the conversation, made inaccurate assumptions instead of asking for additional information, disagreed with its own earlier responses, or left out details. This resulted in additional labor, such as reminding it of previously discussed requirements:

> "*ChatGPT will sometimes contradict itself or leave out details from earlier in the discussion, as the conversation progresses, so we had to keep double checking its work and reassessing that the work we incorporated from it was correct.*" (T_{171})

One recurring issue was the struggles ChatGPT had with relationships. It was unable to distinguish dependency and aggregation relationships, and omitted key relationships between classes. Even when teams attempted to correct this via detailed prompts, ChatGPT was unable to make the necessary changes to the UML.

5.3.3 Introducing Complexity. The teams noted that the responses often contained, from their perspectives, unnecessary complexity that they therefore did not incorporate into their design. Two such examples are the following: (1) offering suggestions deemed by the team to be outside the scope of the requirements, and (2) including additional classes such as helper classes that were not needed to address the essence of the problem in UML diagrams. Having to review the designs for such complexity caused an unnecessary distraction for some, "*It often began making unnecessary changes or additions that distracted from tackling the real problem.*" (T_{199}) Teams applied the expert practices (Table 1) to help them identify

potential complexities, specifically by focusing on the essence of the problem and preferring simpler solutions:

5.3.4 Superficiality. Teams noted that the responses returned by ChatGPT were sometimes unhelpful, as they were too general or lacked depth to really address the prompt. The use of the taught expert practices particularly helped them realize when responses that initially seemed good and to the point actually lacked, "*The generality of the answers meant that they often strayed from the essence of the problem.*" (T_{183}) or when responses focused on functionality that distracted from the task at hand "*Adding in sound effects, gamification, and a tutorial/help system strays too far from the essence of the software.*" (T_{154})

5.4 Student Reflections

When reflecting on their use of ChatGPT for the assignment (as was a mandatory part of the design studio), there was a consensus that it was beneficial, but many teams felt they themselves still had a critical role to play in shaping the design. Below, we introduce five themes that emerged from across the reflections by the teams.

5.4.1 Helpful for Some Aspects of the Design Process. Many teams noted that, while ChatGPT was helpful, they felt it could only be used for parts of the design process, especially at the start when faced with an empty slate. They got their design kickstarted through ChatGPT's ability to quickly generate many ideas that could be discussed and taken forward into the design. One team noted that getting started, was "*Usually the hardest part*" (T_{100}), and so it helped them get over the initial hump faced when faced with a new project.

Teams felt ChatGPT was less beneficial once the design became larger and slightly more complex. None of the teams felt it was good enough to complete the entire design without the team leading the process.

Arguably, the underlying reason teams considered the ChatGPT generated designs as merely starting points was due to inaccuracies that they spotted in the responses (discussed in Section 5.3). Many teams recognized a review was necessary as ChatGPT was not perfect, and that its responses should be taken as suggestions, treated with skepticism, and that "*Its suggestions should be critically evaluated.*" (T_{141}) For some, realizing ChatGPT was fallible was a learning point, "*At first, I unfortunately made the mistake of taking its responses too seriously; that is, I thought that whatever it cooked up was going to be in the final design. This caused some panic in me because I did not get ChatGPT to give me satisfying results*" (T_{146}).

5.4.2 Collaboration. Despite the issues, teams felt ChatGPT acted as a valued team member able to support and coach the team. It supported teams in many different ways, including helping to spark new ideas when a team would riff off the ideas provided by GenAI, supplying alternative perspectives to those identified by the team, unblocking the team when stuck, or "*Jumpstarting the sessions when we reached lulls or obstacles*" (T_{183}), or even providing a "*Platform to bounce ideas off.*" (T153) By reviewing its responses, it can also help the team better understand the problem, "*Including another perspective in our brainstorming process helped to identify other edge cases and further our understanding of the problem at hand.*" (T_{130}).

One team mentioned that, because it was viewed as being "*An impartial platform for idea growth and confirmation*" (T_{122}), it felt that using ChatGPT improved team dynamics. This is an interesting perspective given students' teamwork struggles when working on team projects [15].

5.4.3 The Role of Human Designers. Overwhelmingly, the teams felt human designers needed to lead the design, rather than being wholly dependent on ChatGPT to generate the design. They felt humans were essential for the critical thinking and decision-making required when designing, viewing those as tasks that ChatGPT could not perform due to the lack of contextual understanding and intuition human designers have. As noted earlier, teams needed to sense-check ChatGPT's suggestions to ensure the design was not becoming overly complicated, straying away from the essence of the problem, or just making sense.

Some teams noted that ChatGPT-generated responses lack creativity, so much so that some felt it was important that designers should rely on their own skills and creativity to generate an initial design before using ChatGPT, as there is a risk that "*Designers could potentially limit their understanding of the problem by relying on them too much.*" (T_{111}) That said, we observed a good number of cases where ChatGPT responses sparked human ideas, suggesting blending human ideas with ChatGPT-generated ideas can be important to imbue creativity in design.

5.4.4 Productivity. Similar to studies on coding [34], teams noted a perceived boost to their productivity by using ChatGPT and felt they were able to complete their designs more quickly. One team even felt its use "*Definitely cutting down our whole process in half.*"(T_{132}) Productivity improved for several reasons, including the speed at which ChatGPT generates ideas and the automation of "busy" work through its ability to generate structured formats such as PlantUML that could be quickly improved upon through iteration based on feedback from the team.

One team observed that cost-benefit trade-offs had to be made when deciding whether to persist in refining the model via ChatGPT or stop using it and take the design into a tool for manual updates. While mentioned only by this team in their reflection, this clearly is an important skill students have to practice and learn.

5.4.5 Helpful Learning Experience. Some students appreciated the opportunity to use GenAI in their assignments, noting that it was an interesting and eye-opening experience and that it helped to highlight the potential of "*AI-assisted design*" (T_{180}), more so as its use is still banned in other classes. One team even noted that, for all the team members, this was the first assignment at the University that they were "*Granted permission to use AI without being penalized for plagiarism or academic dishonesty.*" (T_{181}) Perhaps because of this lack of experience, some teams struggled with finding prompts that gave reasonable answers and felt disappointed with ChatGPT. Others, however, used the class as an opportunity to learn how to author prompts and also how to best incorporate ChatGPT into their design process:

> "*In the beginning, we simply input the prompt and had it generate the project for us. However, most of the output was overwhelming ... As we progressed ..., we realized ChatGPT was a good way for us in creating starter points for our designs, such as our UML diagram and pseudocode.*" (T_{151})

6 Lessons Learned

This experience report on the use of GenAI within a software design education class contributes to the growing body of research on using GenAI for software development education, with a specific focus on software design. While (UML) modeling was part of the exercise, our treatment of design was broader and incorporated a focus on expert design behaviors. In this context, we share below our lessons learned for educators interested in repeating our approach to introducing GenAI into software design classes.

#1: Students appreciated GenAI being incorporated into their design class. It helped them get started, generate ideas, validate their solutions, and speed up their process. In their reflections, the teams recognized these benefits, but equally highlighted some of the issues that they encountered, such as GenAI's unreliability and tendency to introduce complexity. Students considered GenAI a tool that can supplement human design skills, which they firmly believed remain at the core of the design process by deciding when and what to incorporate into the design through thoughtful prompting and critical review of the responses. This perspective is similar to findings in other design areas (e.g., [24]). For educators, then, the lesson is that in our teaching we must introduce GenAI as a tool, similar to the way we introduce other software design tools. GenAI should be explained, practiced with, and discussed in terms of what value it can bring and where its limitations are – so that students become familiar with how to best work with this new kind of tool that undoubtedly they will use in their future careers.

#2: Asking for reflections enhances pedagogy. As already observed, asking the teams to reflect on their experiences helped reinforce the advantages and disadvantages of using GenAI for software design. Beyond, however, some teams utilized advanced prompting strategies, which presents an opportunity for knowledge sharing within the class. In addition to asking students for their written reflections, then, we recommend educators incorporate opportunities for open discussion in class, so teams can share and benefit from each other's approaches and experiences.

#3: Expert design practices can help anchor GenAI usage. The expert practices we teach in our class (see Table 1) serve as a crucial framework for the course overall, guiding students in how they approach their design projects. Ideally, their use of GenAI fits in with this approach. We indeed witnessed some of that, both in terms of a limited amount of explicit prompting (as discussed prior) and in how the students reviewed the responses from GenAI (e.g., testing if they felt it addressed the essence, ignoring additional cruft that complicated solutions). As such, we feel that the benefits of teaching the expert design practices can extend into GenAI usage, but that explicit encouragement and especially training (see next insight) are an essential component of achieving the full potential.

We note that our approach of introducing GenAI after teaching (and practicing) some foundational expert practices in lectures and in a first design studio without GenAI helped the students contextualize the subsequent use of GenAI. We recommend a similar sequencing to others wishing to introduce GenAI: teach some basic design first so students become sensitive to the topic before introducing GenAI.

#4: Prompt engineering skills require attention. Today, prompt engineering has become a core practice for students to learn [14]. We provided some training to help students start on prompting, though it was not exhaustive, as for this first time we were curious how students would engage. It is thus perhaps not surprising that some teams struggled to get reasonable responses from GenAI due to inadequate prompts. Moreover, some teams followed the taught approaches rigidly and did not explore alternative prompting strategies. Contrastingly, we saw some highly novel prompts in both the strategy to coax a design from GenAI and in undertaking expert practices such as validating the design either directly or by asking GenAI to self-reflect or to simulate the execution of the pseudocode. Further training before using GenAI could help inexperienced students in prompt engineering. Yet, there is a trade-off, as too much training could lead to more copycat usage rather than student-initiated exploration and reflection. We recommend educators consider how much training is required, which could vary depending on the students' prior experience.

7 Conclusion and Future Work

This paper provides an experience report on introducing GenAI (specifically ChatGPT) into an undergraduate software design class. Students were required to use GenAI in a team-based assignment to produce a design for a stated problem. By analyzing the conversation logs as well as the students' reflections on their experiences using ChatGPT, we observed that students found GenAI helpful in several ways, including generating ideas, getting them started, improving their teamwork, and reducing the time it took to complete their design. However, they all noted the necessity for humans to lead the design process and not rely entirely on GenAI for reasons such as its inability to understand the context or its tendency to overcomplicate the solution. As such, we believe our approach of introducing GenAI after some fundamental design practices have been taught and asking students to reflect on their experiences has helped students learn about the strengths and weaknesses of GenAI for software design.

In terms of our future work, research is needed to explore some of the observations further. In particular, we did not determine whether GenAI makes a difference in learning outcomes as the focus was on students learning more about its capabilities for supporting design. Accordingly, defining appropriate measures of learning, examining how the use of GenAI affects success in future courses and careers, and novel tooling to assist learning outcomes should all be considered further. Exploring alternative approaches to the one outlined here is also needed. For example, breaking up the single assignment into smaller, more targeted assignments where each focuses on different aspects (e.g., idea generation, design verification) may lead to improved and more focused use of GenAI toward student learning.

Acknowledgments

van der Hoek kindly acknowledges support by the National Science Foundation under grants CCF-2210812 and 2326489.

References

[1] Erfan Al-Hossami, Razvan Bunescu, Justin Smith, and Ryan Teehan. 2024. Can Language Models Employ the Socratic Method? Experiments with Code Debugging. In *Proceedings of the 55th ACM Technical Symposium on Computer Science*

Education V. 1 (New York, NY, USA, 2024-03-07) *(SIGCSE 2024)*. Association for Computing Machinery, 53–59. https://doi.org/10.1145/3626252.3630799

[2] Eman Abdullah AlOmar and Mohamed Wiem Mkaouer. 2024. Cultivating Software Quality Improvement in the Classroom: An Experience with ChatGPT. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)* (Würzburg, Germany). IEEE, IEEE, 1–10. https://doi.org/10.1109/CSEET62301.2024.10663028

[3] Alex Baker and André van der Hoek. 2009. An experience report on the design and delivery of two new software design courses. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA) *(SIGCSE '09)*. Association for Computing Machinery, New York, NY, USA, 519–523. https://doi.org/10.1145/1508865.1509045

[4] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. 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* (New York, NY, USA, 2023-03-03) *(SIGCSE 2023)*. Association for Computing Machinery, 500–506. https://doi.org/10.1145/3545945.3569759

[5] Christopher N. Bull and Jon Whittle. 2014. Supporting Reflective Practice in Software Engineering Education through a Studio-Based Approach. *IEEE Software* 31, 4 (2014), 44–50. https://doi.org/10.1109/MS.2014.52

[6] Doga Cambaz and Xiaoling Zhang. 2024. Use of AI-driven Code Generation Models in Teaching and Learning Programming: A Systematic Literature Review. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (New York, NY, USA, 2024-03-07) *(SIGCSE 2024)*. Association for Computing Machinery, 172–178. https://doi.org/10.1145/3626252.3630958

[7] Kua Chen, Yujing Yang, Boqi Chen, José Antonio Hernández López, Gunter Mussbacher, and Dániel Varró. 2023. Automated Domain Modeling with Large Language Models: A Comparative Study. In *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)* (Västerås, Sweden, 2023-10). IEEE, 162–172. https://doi.org/10.1109/MODELS58315.2023.00037

[8] Li-Yuan Chiou, Peng-Kai Hung, Rung-Huei Liang, and Chun-Teng Wang. 2023. Designing with AI: An Exploration of Co-Ideation with Image Generators. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (New York, NY, USA, 2023-07-10) *(DIS '23)*. Association for Computing Machinery, 1941–1954. https://doi.org/10.1145/3563657.3596001

[9] Victoria Clarke and Virginia Braun. 2017. Thematic analysis. *The Journal of Positive Psychology* 12, 3 (2017), 297–298. https://doi.org/10.1080/17439760.2016.1262613

[10] Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. 2023. On the Assessment of Generative AI in Modeling Tasks: An Experience Report with ChatGPT and UML. *Software and Systems Modeling* 22, 3 (2023), 781–793. https://doi.org/10.1007/s10270-023-01105-5

[11] Javier Cámara, Javier Troya, Julio Montes-Torres, and Francisco J. Jaime. 2024. Generative AI in the Software Modeling Classroom: An Experience Report with ChatGPT and UML. *IEEE Software* 41, 6 (2024), 1–10. https://doi.org/10.1109/MS.2024.3385309

[12] Nicole Davila, Igor Wiese, Igor Steinmacher, Lucas Lucio da Silva, André Kawamoto, Gilson José Peres Favaro, and Ingrid Nunes. 2024. An industry case study on adoption of ai-based programming assistants. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*. 92–102.

[13] Gabriele De Vito, Fabio Palomba, Carmine Gravino, Sergio Di Martino, and Filomena Ferrucci. 2023. ECHO: An Approach to Enhance Use Case Quality Exploiting Large Language Models. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (Durres, Albania, 2023-09). IEEE, 53–60. https://doi.org/10.1109/SEAA60479.2023.00017

[14] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (New York, NY, USA, 2024-03-07) *(SIGCSE 2024)*. Association for Computing Machinery, 296–302.

[15] Helen Donelan and Karen Kear. 2024. Online group projects in higher education: persistent challenges and implications for practice. *Journal of computing in higher education* 36, 2 (2024), 435–468.

[16] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. 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)* (Melbourne, Australia, 2023-05). IEEE, 31–53. https://doi.org/10.1109/ICSE-FoSE59343.2023.00008

[17] Alessio Ferrari, Sallam Abualhaijal, and Chetan Arora. 2024. Model Generation with LLMs: From Requirements to UML Sequence Diagrams. In *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*. IEEE, 291–300. https://doi.org/10.1109/REW61692.2024.00044

[18] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024), 1–79.

[19] Victoria Jackson, Susannah Liu, and André van der Hoek. 2025. *Supplementary Data*. https://doi.org/10.5061/dryad.rr4xgxdjg

[20] Jasmin Jahić and Ashkan Sami. 2024. State of Practice: LLMs in Software Engineering and Software Architecture. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)* (Hyderabad, India, 2024-06). IEEE, 311–318. https://doi.org/10.1109/ICSA-C63560.2024.00059

[21] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. 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* (New York, NY, USA, 2023-04-19) *(CHI '23)*. Association for Computing Machinery, 1–23.

[22] Vassilka D. Kirova, Cyril S. Ku, Joseph R. Laracy, and Thomas J. Marlowe. 2024. Software Engineering Education Must Adapt and Evolve for an LLM Environment. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (New York, NY, USA, 2024) *(Sigcse 2024)*. Association for Computing Machinery, 666–672. https://doi.org/10.1145/3626252.3630927

[23] Sam Lau and Philip Guo. 2023. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 106–121. https://doi.org/10.1145/3568813.3600138

[24] Jie Li, Hancheng Cao, Laura Lin, Youyang Hou, Ruihao Zhu, and Abdallah El Ali. 2024. User Experience Design Professionals' Perceptions of Generative Artificial Intelligence. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2024-05-11) *(CHI '24)*. Association for Computing Machinery, 1–18. https://doi.org/10.1145/3613904.3642114

[25] 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* (New York, NY, USA, 2024-03-07) *(SIGCSE 2024)*. Association for Computing Machinery, 750–756. https://doi.org/10.1145/3626252.3630938

[26] Tim Menzies, Brittany Johnson, David L. Roberts, and Lauren Alvarez. 2023. The Engineering Mindset Is an Ethical Mindset (We Just Don't Teach It That Way… Yet). *IEEE Software* 40, 2 (2023), 103–110. https://doi.org/10.1109/MS.2022.3227597

[27] Ha Nguyen and Vicki Allan. 2024. Using GPT-4 to Provide Tiered, Formative Code Feedback. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (New York, NY, USA, 2024-03-07) *(SIGCSE 2024)*. Association for Computing Machinery, 958–964. https://doi.org/10.1145/3626252.3630960

[28] OMG. 2024. *Welcome To UML Web Site!* https://www.uml.org/

[29] Marian Petre and André van der Hoek. 2016. *Software Design Decoded: 66 Ways Experts Think*. MIT Press.

[30] Cigdem Sengul, Rumyana Neykova, and Giuseppe Destefanis. 2024. Software Engineering Education in the Era of Conversational AI: Current Trends and Future Directions. *Frontiers in Artificial Intelligence* 7 (2024). https://doi.org/10.3389/frai.2024.1436350

[31] Macy Takaffoli, Sijia Li, and Ville Mäkelä. 2024. Generative AI in User Experience Design and Research: How Do UX Practitioners, Teams, and Companies Use GenAI in Industry?. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (New York, NY, USA, 2024-07-01) *(DIS '24)*. Association for Computing Machinery, 1579–1593. https://doi.org/10.1145/3643834.3660720

[32] PlantUML Team. 2024. *PlantUML Web Server*. https://plantuml.com

[33] Jakob Tholander and Martin Jonsson. 2023. Design Ideation with AI - Sketching, Thinking and Talking with Generative Machine Learning Models. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (New York, NY, USA, 2023-07-10) *(DIS '23)*. Association for Computing Machinery, 1930–1940.

[34] Thomas Weber, Maximilian Brandmaier, Albrecht Schmidt, and Sven Mayer. 2024. Significant Productivity Gains through Programming with Large Language Models. *Proc. ACM Hum.-Comput. Interact.* 8 (2024), 256:1–256:29. Issue EICS.

[35] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. 2023. The Effect of Generative Artificial Intelligence (AI)-Based Tool Use on Students' Computational Thinking Skills, Programming Self-Efficacy and Motivation. *Computers and Education: Artificial Intelligence* 4 (2023), 100147. https://doi.org/10.1016/j.caeai.2023.100147