Towards Educator-Driven Tutor Authoring: Generative AI Approaches for Creating Intelligent Tutor Interfaces

Tommaso Calo
Politecnico Di Torino
Torino, Italy, ITA
tommaso.calo@polito.it

Christopher MacLellan
Georgia Institute of Technology
Atlanta, Georgia, USA
cmaclellan3@gatech.edu

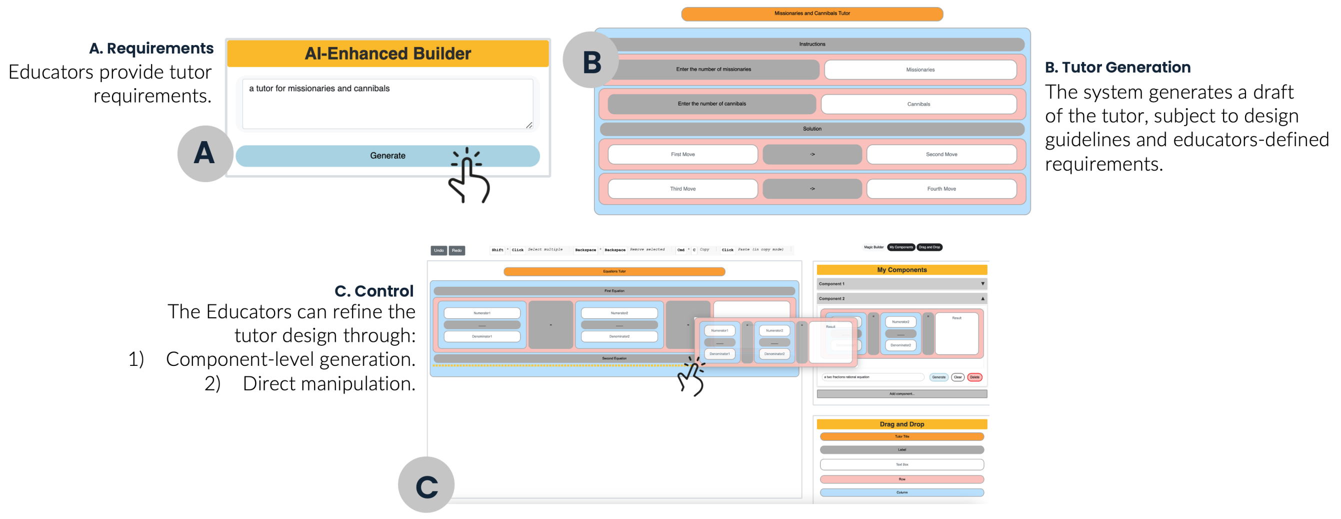
A. Requirements
Educators provide tutor requirements.
AI-Enhanced Builder
a tutor for missionaries and cannibals
Generate
A
B
Missionaries and Cannibals Tutor
Instructions
Enter the number of missionaries
Missionaries
Enter the number of cannibals
Cannibals
Solution
First Move
Second Move
Third Move
Fourth Move
B. Tutor Generation
The system generates a draft of the tutor, subject to design guidelines and educators-defined requirements.
C. Control
The Educators can refine the tutor design through:
1) Component-level generation.
2) Direct manipulation.
My Components
Drag and Drop
C

Figure 1: Illustration of our AI-Enhanced Tutor Builder System: Educators provide input requirements (A) which inform the automated generation of a draft tutor interface (B), followed by the educator's hands-on refinement through component generation and direct manipulation (C), visualizing the integration of generative design and educator-driven customization.

ABSTRACT

Intelligent Tutoring Systems (ITSs) have shown great potential in delivering personalized and adaptive education, but their widespread adoption has been hindered by the need for specialized programming and design skills. Existing approaches overcome the programming limitations with no-code authoring through drag and drop, however they assume that educators possess the necessary skills to design effective and engaging tutor interfaces. To address this assumption we introduce generative AI capabilities to assist educators in creating tutor interfaces that meet their needs while adhering to design principles. Our approach leverages Large Language Models (LLMs) and prompt engineering to generate tutor layout and contents based on high-level requirements provided by educators as inputs. However, to allow them to actively participate in the design process, rather than relying entirely on AI-generated solutions, we allow generation both at the entire interface level and at the individual component level. The former provides educators with a complete interface that can be refined using direct manipulation, while the latter offers the ability to create specific elements to be added to the tutor interface. A small-scale comparison shows the potential of our approach to enhance the efficiency of tutor interface design. Moving forward, we raise critical questions for assisting educators with generative AI capabilities to create personalized, effective, and engaging tutors, ultimately enhancing their adoption.

L@S '24, July 18–20, 2024, Atlanta, GA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0633-2/24/07.
https://doi.org/10.1145/3657604.3664694

CCS CONCEPTS

• **Human-centered computing → User interface programming**; • **Applied computing → E-learning**; **Interactive learning environments**.

KEYWORDS

Human-centered computing, Intelligent tutoring systems, UI/UX, Intelligent-User-Interfaces

ACM Reference Format:
Tommaso Calo and Christopher MacLellan. 2024. Towards Educator-Driven Tutor Authoring: Generative AI Approaches for Creating Intelligent Tutor Interfaces. 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/10.1145/3657604.3664694

1 INTRODUCTION

The rapid advancements in artificial intelligence is revolutionizing various sectors, and education is no exception. It is transforming the way we teach and learn, enabling personalized learning experiences that adapts to the unique needs and abilities of each student. Among the most promising applications of AI in education are Intelligent Tutoring Systems (ITSs), which have garnered significant attention from researchers and educators alike.

Intelligent Tutoring Systems (ITSs) provide personalized and adaptive education to students by supplying practice problems and sequences tailored to their expertise as well as support and feedback during problem solving. Research shows that ITSs are effective in raising learning outcomes [7]. By tracking each learner's knowledge and skills, ITSs can customize problem sequences to improve learning efficiency, allowing tutors to scale to serve many students simultaneously and address needs around providing supplemental instruction large classes [18].

However, several limitations prohibit the widespread use and applicability of ITSs. One significant limitation is that tutor development often requires specialized programming and tutor design knowledge [13]. This prohibits non-technical educators from building their own customized tutors [5], restricting creation to experts like scholars and software developers.

The recently introduced Apprentice Tutor Builder (ATB) [16] enables educators to construct their own intelligent tutors. It proposes a drag-and-drop interface for assembling tutor layouts row and column formats, allowing them to interactively develop the underlying tutoring model with demonstrations through validating responses and supplying labels. While ATB mitigates the the challenges of tutor model authoring, *it implicitly assumes educators can also effectively design interfaces without specific training or support*. In contrast, research shows that interface and instructional design requires specialized skills [14]. An obvious question is therefore **how can we help educators in the UI design process to help them produce better tutors?**.

To address this, we propose to enhance the Apprentice Tutor Builder with the capabilities of Generative AI [1]. We create a prompt engineered to generate layouts and content directly from the requirements provided by the educators. This prompt incorporates design constraints to allow training of tutor, design principles, and examples to ensure the generated interfaces are effective and visually appealing. However, in order to avoid taking control away from educators in interface design, we enable generation of either the **whole interface** or just of **specific components**. The former serves as a starting point for the interface, and it can be further modified to meet educators' needs. The latter facilitates the creation of specific and reusable components and ensures educators for educators to place them within the final interface. With this dual approach, illustrated in Figure 1, we aim to balance the high automation provided by generative AI with high control for targeted customization. To our knowledge, this method represents a novel application in the field of tutor authoring; indeed, While AI has been leveraged to automate the creation of the underlying models in intelligent tutoring systems [9], its potential to assist end-users, particularly educators, in designing the user interfaces of these systems has remained unexplored. Through a preliminary comparison, we show the potential of our method to accelerate the process of designing tutor interfaces, reducing the time and effort required to create both simple and complex interfaces. However, further research is needed to assess the quality and completeness of the AI-generated interfaces and their impact on the overall effectiveness of the tutoring system. We also raise questions about the optimal integration of generative interface tools into educators' workflows, ensuring the system meets real-world tutoring requirements. These questions will be addressed in a future user study, where we will further develop our approach. We aim to collect detailed feedback from educators on the usability and impact of the Generative AI assistance on their final design process and the resulting interfaces. The final goal is to empower educators to create personalized and effective tutor interfaces by making the tutor design process more efficient and engaging with the assistance of generative AI. The resulting improvements in the usability of intelligent tutor interfaces could ultimately lead to wider adoption of adaptive education among students.

1.1 Related Work

The field of intelligent tutoring systems (ITSs) has witnessed significant advancements in recent years, with researchers and developers striving to create adaptive learning environments that cater to the unique needs of each student [7]. Despite the proven effectiveness of ITSs in enhancing learning outcomes, their widespread adoption has been hindered by the complexity involved in their development, which often requires specialized programming and design skills [13]. In an effort to democratize the creation of ITSs and empower educators to take an active role in their development, various authoring tools and platforms have emerged.

One prominent example is the Cognitive Tutor Authoring Tools (CTAT) [5], which provide a suite of tools for designing and deploying cognitive tutors. CTAT allows educators to create tutors by demonstrating problem-solving steps and specifying pedagogical rules. Similarly, the Authoring Software Platform for Intelligent Resources in Education (ASPIRE) [12] enables domain experts to create constraint-based tutors by defining the domain model and problem-solving strategies. Another notable system is SimStudent [10, 11] and the related Apprentice Learner System [8], which employ machine learning techniques to model student learning and support tutor authoring.

These tools primarily focus on the authoring of the domain model and pedagogical strategies, assuming that educators possess the necessary skills to design effective user interfaces. Our approach, in contrast, specifically addresses the challenges of assisting educators in creating tutor's user interfaces.

Generative user interfaces for end-user design have garnered significant attention in recent years, with various approaches being explored to automate or assist in the design process [15]. One of the main line of research focuses on the unsupervised generation of user interfaces, where the system automatically creates interfaces based on a set of predefined rules or learned patterns. For example, Neural Design Network [6] uses deep learning to generate user interface layouts from a given set of UI components. Closer to our work, Huang. et al. [3] introduced a method to generate layouts from textual descriptions using transformers models.

Interface Type	Time (s)			Keystrokes		
	Classical	AI-Enhanced	Reduction	Classical	AI-Enhanced	Reduction
Simple	187	**143**	**-23%**	184	**126**	**-31%**
Complex	372	**116**	**-68%**	141	**74**	**-47%**

Table 1: Comparison of time and keystrokes required for building tutor interfaces: Classical vs. AI-Enhanced

Another approach involves the controlled generation of sub-elements, where the system assists designers by generating specific components or suggestions based on user input. Stylette [4] is a system that allows end-users to customize web elements based on natural language instructions, while Rewire [17] suggests alternative layouts for a given user interface based on design heuristics and user feedback.

With respect to the above methods, our approach differs in two key aspects. First, it focuses on the application domain of tutor design. Second, it integrates both unsupervised generation at the interface level and controlled generation of components. This is in contrast to previous works, focused on the refinement of specific UI components [4], generation of layouts without human intervention [6], or layout generation without AI support [19].

2 AI-ENHANCED TUTOR INTERFACE BUILDER

Our method for enhancing the ATB with generative AI capabilities involves three key components: a Domain Specific Language (DSL) for communicating with the LLM, prompt engineering to guide the generation process, and two levels of interaction for flexibility and control.

We define a compact DSL to represent tutor interface layouts, which includes fundamental elements such as `title[value]` for specifying the tutor's title, `row` and `column` for representing horizontal and vertical arrangements of elements, `label[value]` for defining text labels, and `input[placeholder]` for defining input fields with optional placeholder text. These elements can be combined to generate complex tutors. The DSL representation enables efficient communication with the LLM and ensures that the generated HTML output adheres to the desired template, preventing inconsistencies and deviations from the intended layout and aesthetics that may arise if a tutor's HTML interface is generated directly from the LLM.

To guide the genera[illegible]del in creating appropriate tutor interfaces, we engineered a prompt consisting of different sections. The **System Description** provides an overview of the desired tutor interface, emphasizing the importance of a clear problem statement and a step-by-step resolution pathway to align with the educational objectives. The **Format Explanation** explains the DSL format used to represent the tutor interface layout, enabling the model to generate layouts that conform to the specified format. The **Design Instructions** specify design principles, such as separating input elements, arranging elements within rows and columns, and avoiding interactive buttons within the layout. The **Task Instruction** instructs the model to transform a detailed description of the tutor into the compact DSL representation, ensuring that the model understands its primary objective and generates the desired output format. Finally, a set of representative **Examples** serves as a reference for the model on how to apply the above principles in different contexts.

We introduce two levels of interface generation to fit into different design phases. First, **Interface Generation** supports creating a complete tutor interface layout based on the provided detailed description. It serves as a starting point for tutor designing. This allows the user to start from a generated user interface aligned with the design principles specified in the prompt. We argue that the refined user interface will be more likely to follow these principles. In addition, interface generation could be particularly useful for users who prefer a more automated approach or have limited time for customization. On the other hand, **Component Generation** outputs designs for specific and reusable interface components; e.g., widgets for equations or other forms and flows. This capability aims to let users create interfaces that closely match their specific intent at a lower scope, while still benefiting from the efficiency and consistency provided by the generative model.

Our AI-enhanced tutor interface builder is implemented on top of the existing ATB interface, leveraging an HTML/Javascript front-end and a Flask backend. We utilize GPT-4 [1] as the LLM engine to power the generation process. The interface and component generator are integrated into ATB's user interface as toolbar widgets.

3 PRELIMINARY EVALUATION

To evaluate the efficiency of the AI-enhanced Apprentice Tutor Interface Builder, we conducted a small-scale comparison with a previous version. We compared the performances of four team members against the reported performances of high-expertise individuals from the ATB paper. Although comparing our team's performance with that of the high-expertise ATB participants is not a strictly controlled comparison, we believe their high level of expertise significantly influences their performance, making it a suitable basis for preliminary comparison. Furthermore, to support our findings and mitigate some of the issues associated with comparing different participant groups, we employed the Keystroke-Level Model [2]. This model evaluates the efficiency of the interface by measuring the minimum number of keystrokes required to complete tasks, providing a quantitative measure of user interaction efficiency.

Evaluation Setup. Users were asked to design the same two interfaces used in a prior ATB evaluation [16]: a simple interface for a sequential problem and and a more complex interface for an arithmetic equation solver, both illustrated in Figure 2. We recorded

[1]version gpt-4-0613

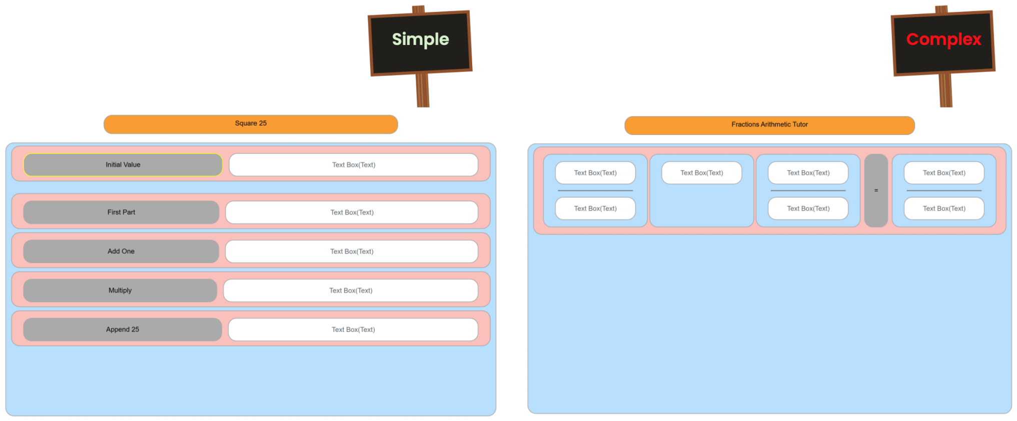

Figure 2: This image illustrates the two interfaces used in the evaluation: On the left, the 'Simple' interface, designed for sequential problem-solving tasks, offers a user-friendly layout with simple input fields. On the right, the 'Complex' interface is tailored for an arithmetic equation solver, featuring a more advanced layout with multiple input fields and operational functions to handle equations.

the time taken for each task to compare the efficiency of the two approaches.

Results. As shown in Table 1, using AI assistance consistently reduced the time required to build both simple and complex. Interestingly, the efficiency gain was more pronounced in the case of the complex interface, with a 68% reduction in time compared to the classical approach. This can be attributed to the fact that the complex interface, tailored for an arithmetic equation solver, required a more advanced layout with multiple input fields and operational functions. In this scenario, the AI assistance likely played a more significant role as users could leverage it to generate the equation components and layout elements more efficiently. On the other hand, the efficiency gain for the simple interface was lower but substantial, at 23%. This lower gain can be explained by the need for users to manually type all the labels in the simple interface, a process that was not necessary for the complex interface. This manual input likely offset some of the efficiency gains provided by the generative AI capabilities. Furthermore, although the reduction in absolute keystrokes is not as substantial as the time savings, it still supports the overall improvements in efficiency.

Limitations. While this small-scale comparison provides promising indications of generative AI potential to improve tutor's interface design efficiency, particularly for complex tutors, further large-scale studies with diverse participants, tasks, and detailed feedback are necessary to validate these findings and gain deeper insights into the tool's impact on user satisfaction and the design process.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we presented an approach to improve the design of intelligent tutors' with Generative AI. The comparison demonstrates the potential of the approach to significantly increase efficiency of interface designing, particularly for complex interfaces. To develop further our approach, we aim to address the following research questions:

- **RQ1:** How can generative interface tools best integrate into educators' tutor building efforts?
- **RQ2:** What is the optimal configuration to balance design freedom and control for educators?
- **RQ3:** To what extent can automatic design tools meet the diverse requirements of real-world tutoring contexts?

To address these questions, we plan to conduct a study involving educators to assess our approach. This study will provide insights into how generative interface tools can best meet the needs of educators. Moreover, understanding how well these tools can accommodate the diverse requirements of real-world tutoring contexts is crucial for their successful adoption. By actively involving educators in the development process, we can identify interaction patterns that align our solutions with their needs, ultimately enhancing the quality and accessibility of intelligent tutoring systems.

Finally, we believe that our approach could be extended and applied to a wider range of tutor-building tools, such as CTAT [5]. By investigating the generalizability of our approach, we aim to unlock new ways of integrating generative AI in supporting educator-driven tutor design. Moreover, the concepts and methods presented in this paper could also find applications in other areas of end-user design, such as website builders or game development tools, opening up new possibilities for empowering non-expert users to create engaging and effective digital experiences.

5 ACKNOWLEDGEMENT

This project is supported by National Science Foundation under Grant No. 2247790 and Grant No. 2112532. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1980. The keystroke-level model for user performance time with interactive systems. *Commun. ACM* 23, 7 (jul 1980), 396–410. https://doi.org/10.1145/358886.358895

[3] Forrest Huang, Gang Li, Xin Zhou, John F Canny, and Yang Li. 2021. Creating User Interface Mock-ups from High-Level Text Descriptions with Deep-Learning Models. *arXiv preprint arXiv:2110.07775* (2021).

[4] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans,LA,USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 5, 17 pages. https://doi.org/10.1145/3491102.3501931

[5] K.R. Koedinger, V. Aleven, N. Heffernan, B. McLaren, and M. Hockenberry. 2004. Opening the Door to Non-programmers: Authoring Intelligent Tutor Behavior by Demonstration. In *Intelligent Tutoring Systems*, J.C. Lester, R.M. Vicari, and F. Paraguaçu (Eds.).

[6] Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B. Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. 2020. Neural Design Network: Graphic Layout Generation with Constraints. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 491–506.

[7] Wenting Ma, Olusola O Adesope, John C Nesbit, and Qing Liu. 2014. Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of Educational Psychology* 106, 4 (2014), 901.

[8] C.J. MacLellan and K.R. Koedinger. 2022. Domain-General Tutor Authoring with Apprentice Learner Models. *Int J Artif Intell Educ* 32, 1 (2022), 76–117. https://doi.org/10.1007/s40593-020-00214-2

[9] Christopher J. MacLellan, Erik Harpstead, Rony Patel, and Kenneth R. Koedinger. 2016. The Apprentice Learner Architecture: Closing the Loop between Learning Theory and Educational Data. In *Proceedings of the International Conference on Educational Data Mining (EDM)*. International Educational Data Mining Society, Raleigh, NC. https://doi.org/10.13140/RG.2.2.28609.35684

[10] Christopher J MacLellan, Kenneth R Koedinger, and Noboru Matsuda. 2014. Authoring tutors with SimStudent: An evaluation of efficiency and model quality. In *Intelligent Tutoring Systems: 12th International Conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. Proceedings 12*. Springer, 551–560.

[11] Noboru Matsuda, William W Cohen, Jonathan Sewall, and Kenneth R Koedinger. 2007. SimStudent: a machine learning agent that models student learning for tutor authoring. In *Proceedings of the 2007 conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work*. IOS Press, 599–601.

[12] A. Mitrovic, B. Martin, P. Suraweera, K. Zakharov, N. Milik, J. Holland, and N. McGuigan. 2009. ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education* 19, 2 (2009), 155–188.

[13] Tom Murray. 2003. An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software* (2003), 491–544.

[14] Jakob Nielsen. 2023. Bridging the Designer-User Gap. https://www.nngroup.com/articles/bridging-the-designer-user-gap/. Accessed: 2024-02-26.

[15] Jakob Nielsen. 2023. Generative UI and Outcome-Oriented Design. https://www.nngroup.com/articles/generative-ui/. Accessed: 2024-04-10.

[16] Glen Smith, Adit Gupta, and Christopher MacLellan. 2024. Apprentice Tutor Builder: A Platform For Users to Create and Personalize Intelligent Tutors. arXiv:2404.07883 [cs.HC]

[17] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Andrew J Ko. 2018. Rewire: Interface design assistance from examples. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[18] Kurt Vanlehn. 2006. The Behavior of Tutoring Systems. *Int. J. Artif. Intell. Ed.* 16, 3 (aug 2006), 227–265.

[19] Marcus Woo. 2020. The Rise of No/Low Code Software Development—No Experience Needed? *Engineering* 6 (07 2020). https://doi.org/10.1016/j.eng.2020.07.007