Revista EDaPECI
EDUCAÇÃO A DISTÂNCIA E PRÁTICAS EDUCATIVAS COMUNICACIONAIS E INTERCULTURAIS

ISSN: 2176-171X

Revista EDaPECI
São Cristóvão (SE)
v.26. n. 1, p. 232-244
jan./abr. 2026

DOI: http://dx.doi.org/10.29276/redapeci.2026.26.123392.232-244

Teaching software requirements engineering with event storming and generative AI: experience report

---

O ensino da engenharia de requisitos de software com event storming e IA generativa: relato de experiência

---

La enseñanza de la ingeniería de requisitos de software con tormentas de eventos e inteligencia artificial generativa: informe de experiencia

Marcos Cesar Barbosa dos Santos[1]
Shexmo Richarlison Ribeiro dos Santos[2]
Fabio Gomes Rocha[3]

Abstract. *This article presents an experience report with an exploratory approach, combining results measurement methods applied in the educational context of Software Engineering. The study aims to investigate the responsible use of Large Language Models (LLMs) in generating software requirements based on artifacts produced by Event Storming workshops. It aims to evaluate whether combining methodologies with AI-based tools can improve student autonomy and conceptual understanding. The research was conducted with final-year undergraduate students in a Systems Analysis and Development program. The activity was structured into three phases: training and project planning using Lean Inception, requirement generation using LLMs based on Event Storming outputs, and individual evaluation. We collected data through a Likert-scale questionnaire. The findings indicate that even students with no experience in Event Storming were able to use the technique to generate inputs for LLMs. The results showed a high utilization of the requirements generated by AI in conjunction with Event Storming. The combined use of Event Storming and LLMs has proven promising for teaching requirements engineering, fostering critical thinking, and promoting student autonomy. We observed in the study that students who had greater ease in meeting the requirements were more critical of the quality.*

Keywords: *Active Learning. Event Storming. Large Language Models. Requirements Engineering. Software Engineering.*

Resumo: Este artigo apresenta um relato de experiência com abordagem exploratória, combinando métodos de mensuração de resultados aplicados no contexto educacional de Engenharia de Software. O estudo visa investigar o uso responsável de Large Language Models (LLMs) na geração de requisitos com artefatos produzidos por Event Storming. O objetivo é avaliar se a combinação dessa metodologia com IA pode melhorar a autonomia e compreensão dos alunos. A pesquisa foi conduzida com alunos de graduação de uma disciplina de Análise e Desenvolvimento de Sistemas. A atividade foi estruturada em três fases: treinamento e planejamento do projeto utilizando Lean Inception, geração de requisitos utilizando LLMs com os resultados do Event Storming e avaliação individual. Os dados foram coletados por meio de questionário em escala Likert. Os resultados indicam que alunos inexperientes em Event Storming conseguiram utilizar a técnica para gerar requisitos com LLMs. Os resultados demonstraram um aproveitamento dos requisitos gerados em conjunto com o Event Storming. O uso combinado de Event Storming e LLMs mostrou-se promissor para o ensino de engenharia de requisitos, fomentando a crítica e promovendo autonomia dos alunos. Observamos que os alunos com mais facilidade em gerar requisitos eram mais críticos em relação à qualidade.

Palavras-chave: Aprendizado ativo. Event etorming. Large language models. Engenharia de requisitos. Engenharia de software.

1 Mestrando no Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe (UFS), marcos.cesar.se@gmail.com.
2 Doutorando em Ciência da Computação pela Universidade Federal da Bahia (UFBA), shexmor@gmail.com.
3 Doutor em Educação, Professor no Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Sergipe (UFS), Coordenador de pesquisa e Inovação na NTT Data, gomesrocha@gmail.com.

Resumen: *Este artículo presenta un reporte de experiencia con enfoque exploratorio, combinando métodos cuantitativos y cualitativos aplicados al contexto educativo de la Ingeniería de Software. Investigar el uso responsable de Large Language Models (LLMs) en la generación de requisitos a partir de artefactos producidos mediante Event Storming. Se busca analizar si esta combinación mejora la comprensión conceptual y la autonomía de los estudiantes. El estudio se realizó con estudiantes de pregrado en Análisis y Desarrollo de Sistemas. La actividad se dividió en tres fases: formación y planificación con Lean Inception, generación de requisitos con LLMs a partir de Event Storming, y evaluación individual mediante cuestionario con ítems en escala Likert. Los estudiantes, aun sin experiencia previa, aplicaron Event Storming para generar insumos que alimentaron eficazmente los LLMs. La mayoría consideró adecuados los requisitos generados, aunque algunos requerían ajustes. Estudiantes con mayor dominio en redacción de requisitos fueron más críticos en sus evaluaciones. El uso combinado de Event Storming y LLMs mostró ser prometedor para la enseñanza de ingeniería de requisitos, favoreciendo el pensamiento crítico y la autonomía estudiantil en el proceso de aprendizaje.*

Palabras clave: *Aprendizaje activo. Event storming. Large language models. Ingeniería de requisitos. Ingeniería de software.*

1 INTRODUCTION

Software Engineering education has undergone continuous methodological revisions in response to the profound transformations brought about by digital transformation in educational and technological contexts. The increasing sophistication of information technologies and global integration presents new challenges and demands in the training of professionals capable of working in complex, dynamic, and highly collaborative environments (Rocha, Sabino, Acipreste, 2015). In this scenario, aligning teaching practices with contemporary demands becomes essential, narrowing the gap between academic learning and professional practice while promoting the active construction of knowledge (Rocha, 2019).

According to Ausubel (1963, 1968), meaningful learning occurs when new knowledge is non-arbitrarily linked to existing cognitive structures, allowing for its critical and lasting assimilation. This perspective gains even more relevance within the digital culture, where easy access to information does not necessarily lead to deep learning (Masini & Moreira, 2023; Novak, 2010). As a result, it becomes necessary to adopt approaches that foster autonomy, critical thinking, and collaborative learning.

Active pedagogical practices such as Test-Driven Development (TDD) and Behavior-Driven Development (BDD) have been successfully integrated into Software Engineering education due to their ability to immerse students in real-world problems and agile development practices. In the same vein, Event Storming has emerged as a collaborative domain modeling technique that supports collective understanding of problems and enables the visual and interactive identification of requirements (Rocha, Misra, & Soares, 2023). Within this context, Lean Inception is introduced as a complementary tool, offering a structured approach for defining Minimum Viable Products (MVPs). Its application in educational settings enables students to experience strategic and technical alignment dynamics, in line with agile and user-centered practices.

Simultaneously, the advancement of Large Language Models (LLMs), such as those developed by OpenAI and Google, opens new possibilities for supporting the teaching and learning process. These technologies have increasingly been used to assist in the automated generation of software artifacts, including functional and non-functional requirements, representing a promising opportunity to integrate pedagogical strategies with AI-based tools.

This study aims to investigate the critical and responsible use of LLMs in Software Engineering education, focusing on the extraction of requirements from artifacts produced by students during Event Storming sessions. The central objective is to analyze whether combining these technologies with collaborative methodologies can enhance the learning process by fostering deeper conceptual understanding and greater student autonomy.

Based on this premise, the article addresses the following main research question: How

do Software Engineering students evaluate the use of LLMs for generating requirements based on artifacts produced through Event Storming?

In addition, the study explores three secondary research questions:

RQ1: How do students assess the quality of functional and non-functional requirements generated by LLMs from inputs produced through Event Storming?

RQ2: Does the use of Event Storming contribute to the creation of more effective prompts for requirement generation by LLMs?

RQ3: Does the combined use of Event Storming and LLMs foster the development of students' critical autonomy in the evaluation of requirements?

By addressing these questions, this study contributes to the ongoing discussion on the role of generative AI in software engineering education, proposing innovative pedagogical practices that combine collaboration, creativity, and technology to foster more meaningful and context-aware learning.

2 REFERENCIAL

The activity carried out by the students involved requirements documentation. As part of the process, they were required to use appropriate techniques and methodologies aligned with agile development, such as Lean Inception to define the project scope and Event Storming to detail the requirements.

2.1 LEAN INCEPTION

Lean Inception is a workshop-based methodology developed by Paulo Caroli, designed to define a Minimum Viable Product (MVP). Its primary goal is to generate a product that is aligned with the business team, the technical team, and the end user, thereby reducing waste (Caroli, 2018).

A Lean Inception session is conducted initially over approximately one week, with its stages occurring sequentially. Each stage focuses on a specific aspect of the project and is described as follows:

Purpose Alignment – This initial phase aims to discover the overarching objective of the product from the participants' perspectives. It is the first moment when expectations are aligned, though definitions remain broad and exploratory.

Product Vision – Still within the alignment stage, more specific objectives are defined or discarded, progressively consolidating a shared vision of the product.

Personas – Using a Design Thinking approach (Siricharoen, 2021), personas are created to represent potential users, taking into account their diverse goals and limitations.

User Journey – The user's actions are illustrated using Post-itss notes, making it possible to identify key interaction points between users and the system.

Functionality Brainstorm – Based on the personas and identified goals, system functionalities begin to take shape and are defined.

Complexity, Value, and UX Evaluation – At this stage, participants assess each functionality in terms of technical effort, business value, and user (or persona) satisfaction.

Functionality Sequencer – Functionalities are prioritized based on business value, user satisfaction, and complexity. The Functionality Sequencer is the point at which the content of each delivery version is defined.

MVP Canvas – A visual and structured summary that outlines the potential of the proposed product.

2.2 EVENT STORMING

Event Storming, created by Alberto Brandolini, is a requirements-discovery technique that utilizes color-coded Post-itss notes, each with a specific role. As participants advance through the workshop, requirements are explored in depth, avoiding superficial analysis.

Like Lean Inception, Event Storming adopts a workshop format and relies on Post-itss notes, but it is shorter in duration and assigns fixed semantics to every note type (Brandolini, 2013). The facilitator initially guides the discussion toward the system's domain events.

Events represent identifiable milestones within the system—specific points where a change has occurred. Events are written on orange Post-itss notes using an "object + past tense verb" naming convention. For example: *Product added*, *Order placed*, *Report sent*.

Once the events are mapped, the next step is to create the commands that trigger these events. For each event, a corresponding command is written on a blue Post-it note and connected to the event with a line. The command follows a naming pattern based on the verb in the infinitive form, followed by the object, maintaining alignment with the event. For instance, the command for the *"Report sent"* event is *"Send report".*

Commands, in turn, can be initiated by specific users, who are identified using yellow Post-its notes, indicating a type of authorization-based restriction. Some commands, however, are not triggered by users but by other events; in such cases, a connection is established between the triggering event and the command it initiates.

Problems are any obstacles identified at any point during the workshop. Both the text and the placement are open - examples: *Licensing limitations, contractual issues, unstable external API dependencies.*

Aggregators are used to group and define a familiar context for a set of events. These pastel-yellow Post-its as aggregators may represent microservices.

Table 1: Color-code from event storming

Color	Meaning	Recommendation
Orange	Events: Moments in the flow where it is possible to determine that something relevant has occurred	Short phrases using past-tense verbs. Object + verb.
Blue	Commands: Triggers that cause events to happen	Use the same names as the events, but with the verb in the infinitive form
Yellow	Users: Assigned to commands, users are the actors who trigger them	Name of the external system or user role
Green	Interface: The means through which the user interacts with the command	Possible menu or screen through which the user gains access
Lilac	Policy: Conditions that must be met to continue specific flows	Name of the policy or condition
Pink	Problems: Any point in the Big Picture	Free text
Pastel Yellow	Aggregator: Groups together related elements in the Event Storming map	Free text

Source: Developed by the author (2025).

2.3 RELATED WORKS

Event Storming is already used as a system modeling tool and can be extended (Grimm, Rubart, 2021), similarly to UML and its stereotypes, but in a more flexible manner. It also offers a simpler learning curve compared to notations such as BPMN (Copei *et al.*, 2022).

The use of LLMs as a support tool in the requirements phase brings productivity gains, particularly in the stages of documentation and ideation (Ruiz & Panadero, 2024). However, the risk of hallucination remains present, making human intervention essential in the requirements engineering process (Belzner, Gabor, & Wirsing, 2023).

Although the use of LLMs offers several advantages, it also presents certain risks, such as the difficulty in handling contextual details and the potential biases introduced by the algorithm itself (Arora, Grundy, & Abdelrazek, 2024). Data security is another concern, especially when using publicly accessible LLMs, as sensitive process information and business rules may be stored within their databases.

Despite promising results, the use of LLMs as support tools in the Software Engineering process requires human supervision, curation, and even revision (Belzner, Gabor, & Wirsing, 2023). Challenges related to responsibility and ethics are also present when employing such tools in professional environments.

3 METHODOLOGICAL PROCEDURES

This study employs an exploratory approach, combining both qualitative and quantitative characteristics. It was conducted with final-year students from the Systems Analysis and Development graduation at a private higher education institution in the state of Sergipe, Brazil. As part of their evaluation, the students were assigned the task of developing a software project from the planning phase, focusing on requirements, through to the delivery of a stable version in the form of a Minimum Viable Product (MVP).

3.1 PARTICIPANTS

The sample consisted of 17 students regularly enrolled in the course "Integrative Topics". All students carried out the proposed activities in groups; however, they completed the final evaluation questionnaire individually at the end of the experience.

3.2 PROCEDURES

The proposed activity was structured into three main stages, developed continuously and under the supervision of the instructor throughout the course: (1) training and project planning, (2) automated requirement generation using an LLM, and (3) individual evaluation.

3.2.1 TRAINING AND PROJECT PLANNING

Students participated in introductory sessions on Lean Inception, which were used to define the initial project scope and identify the Minimum Viable Product (MVP). Next, the students were introduced to and trained in the Event Storming technique, focusing on building the system's Big Picture based on the functionalities identified during the Lean Inception stage.

3.2.2 AUTOMATED REQUIREMENT GENERATION WITH LLM

With the Big Picture completed, the students utilized a Large Language Model (LLM) with specific instructions regarding Event Storming and requirements engineering. Interaction occurred through a structured prompt containing a summary of the project and the corresponding Big Pictures. Based on this, the model generated functional and non-functional requirements, as well as product quality recommendations.

3.2.3 INDIVIDUAL EVALUATION

After obtaining the LLM outputs, each student assessed the quality of the generated requirements and completed an individual questionnaire consisting of closed-ended items on a Likert scale (e.g., level of adequacy, perceived difficulty, intention to reuse). The questions addressed their experience using the Event Storming technique and included self-assessments of their ability to elaborate requirements with and without AI assistance. The items used in the questionnaire are presented in Table 02.

Table 02 – Questionnaire Administered to Students

	Question
Q1	Is this your first time using Event Storming as a requirement engineering technique?
Q2	Have you previously used LLMs (such as ChatGPT, Gemini, Copilot, etc.) to generate software requirements?
Q3	How would you rate your ability to write functional requirements for your software project?
Q4	How would you rate your ability to write non-functional requirements for your software project?
Q5	Overall, how would you evaluate the functional requirements generated by the LLM?
Q6	Overall, how would you evaluate the non-functional requirements generated by the LLM?
Q7	Were the recommendations for ensuring product quality appropriate for your project?
Q8	Would you use LLMs again to generate functional and non-functional requirements?

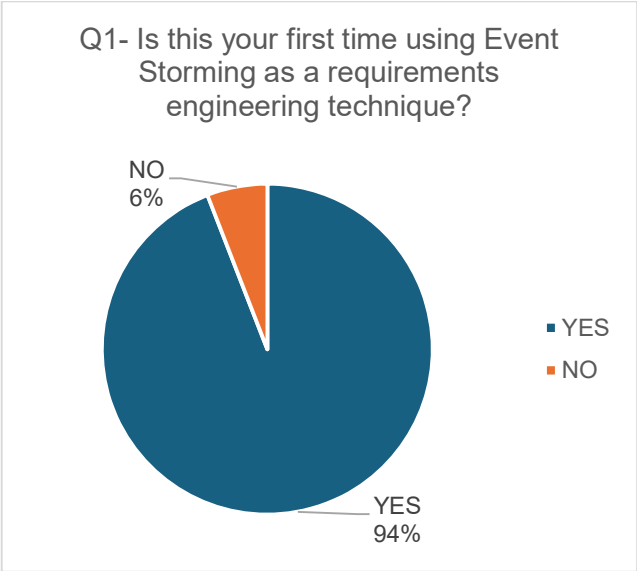
Source: Developed by the author (2025).

The data collected were primarily analyzed using descriptive statistics. This was followed by cross-analysis of selected variables and a fundamental qualitative analysis of students' reported perceptions, categorizing aspects such as usefulness, limitations, critical autonomy, and intention for future use.

4 RESULTS AND DISCUSSION

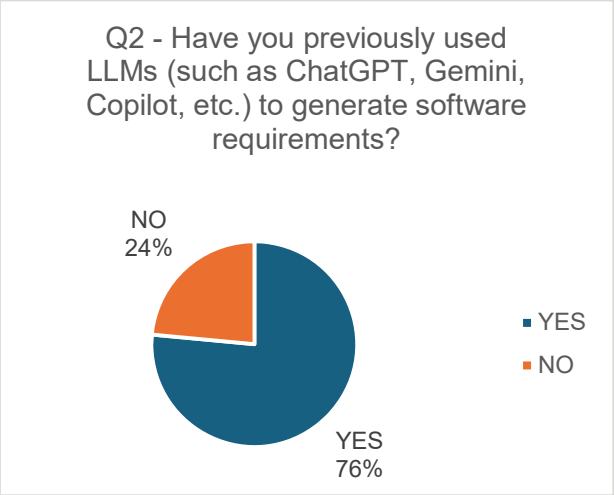
Based on the descriptive analysis of the responses from the individual evaluation questionnaires, the data were grouped, and several initial observations emerged:

Figure 1- Q1- Is this your first time using Event Storming as a requirement engineering technique?



Source: Developed by the author (2025).

Figure 2 – Q2 Have you previously used LLMs (such as ChatGPT, Gemini, Copilot, etc.) to generate software requirements?



Source: Developed by the author (2025).

a) Student experience: Only one student reported having used the Event Storming technique before the experiment, although most students had previously worked with LLMs for working with software requirements.

b)Self-assessed competence: Most students reported difficulties in writing requirements. Functional requirements were easier to develop than non-functional ones.

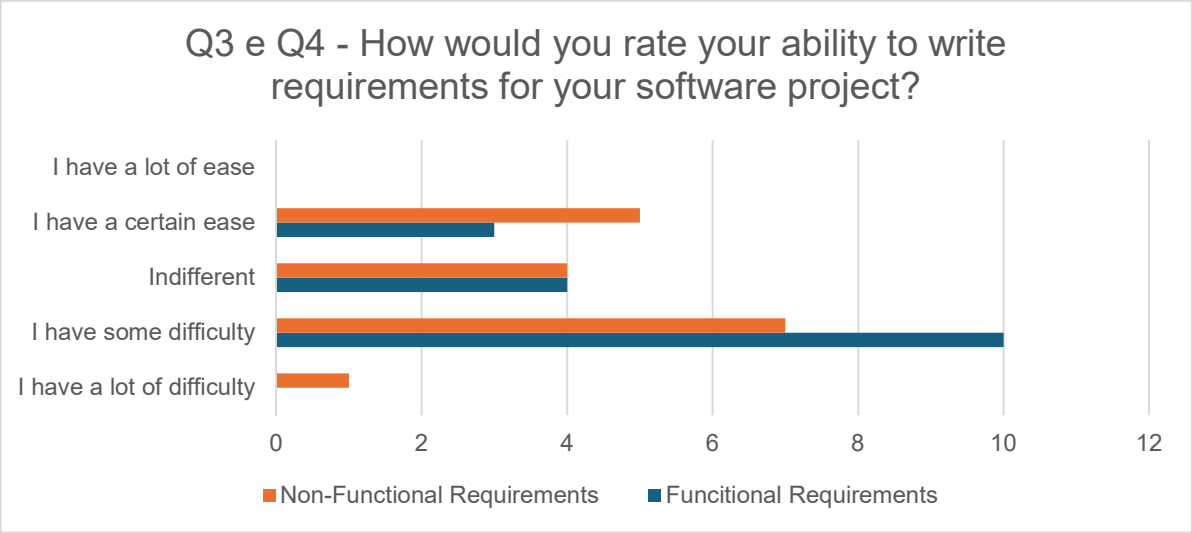
c)Evaluation of LLM-generated functional and non-functional requirements: The majority of students considered the functional and non-functional requirements generated by the LLM to be adequate. However, they noted the need for adjustments.

d) Evaluation of quality recommendations: Although 42% of students found the quality-related recommendations to be helpful, no student expressed negative criticism regarding this aspect.

e) Reusing LLMs for requirements engineering: Only 35.3% of students indicated they would consider using an LLM again for requirements generation, while 11.8% reported being indifferent.

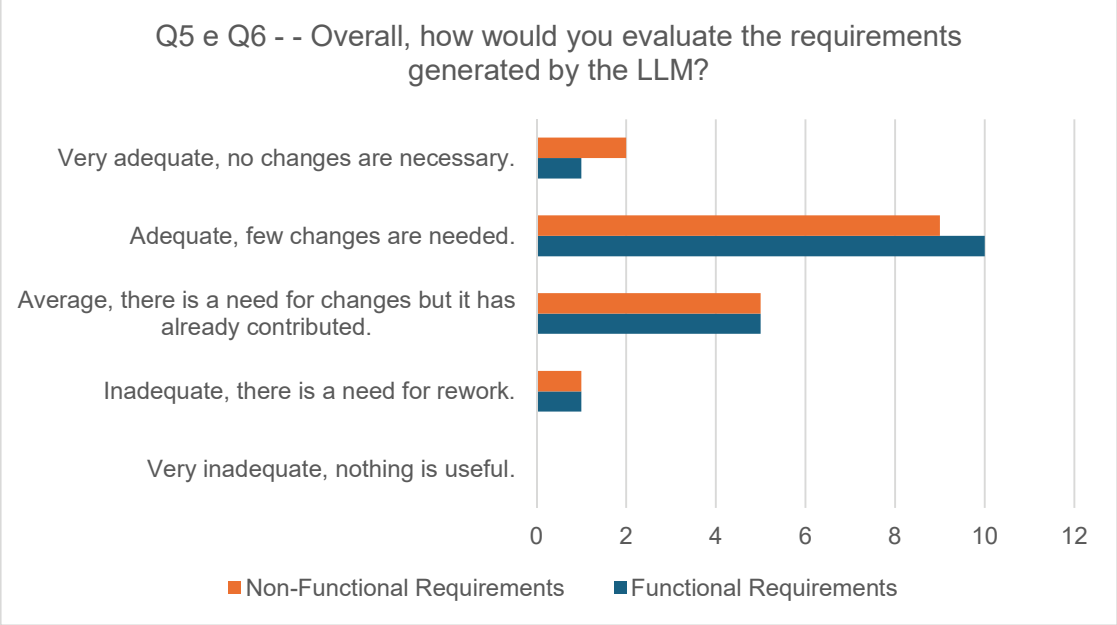
The questionnaire data generated several charts for analysis. When students were initially asked to develop user stories freely, it left room for a mechanical and uncritical use of artificial intelligence tools. However, by explicitly incorporating the LLM into the activity, students were encouraged to evaluate the results critically. This critical engagement is reflected in the responses, as illustrated in Figures 4 and 5. The findings are even more relevant given that students were prompted to reflect on their ability to develop both functional and non-functional requirements, as shown in Figure 3.

Figure 3 – How would you rate your ability to write requirements for your software project?



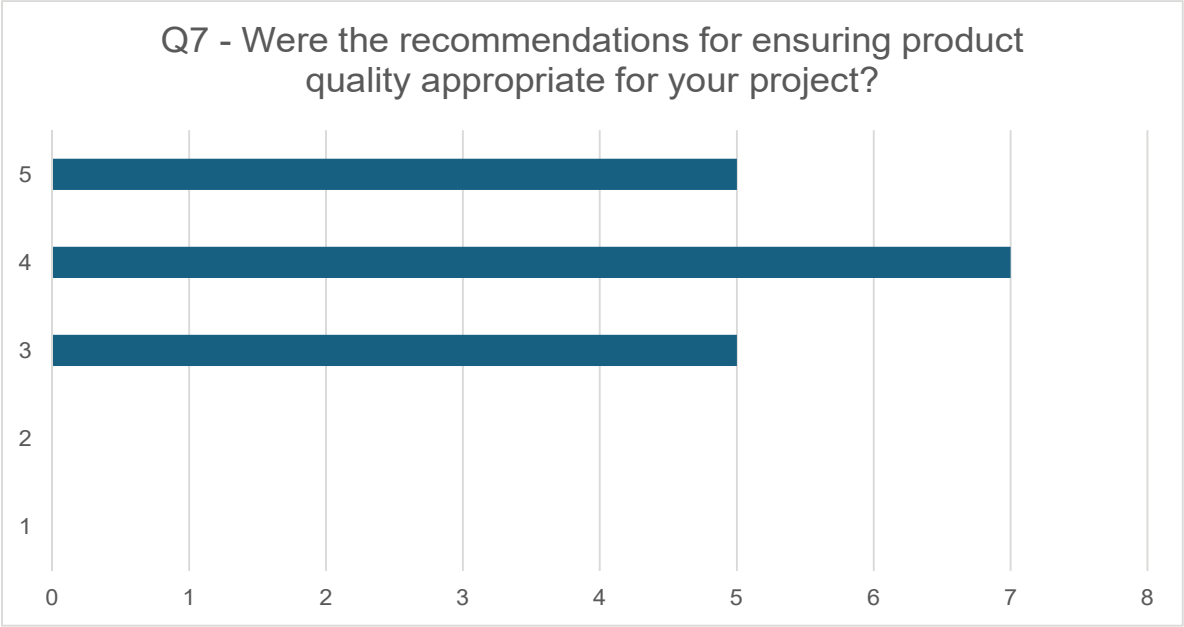
Source: Developed by the author (2025).

Figure 4- Overall, how would you evaluate the requirements generated by the LLM?



Source: Developed by the author (2025).

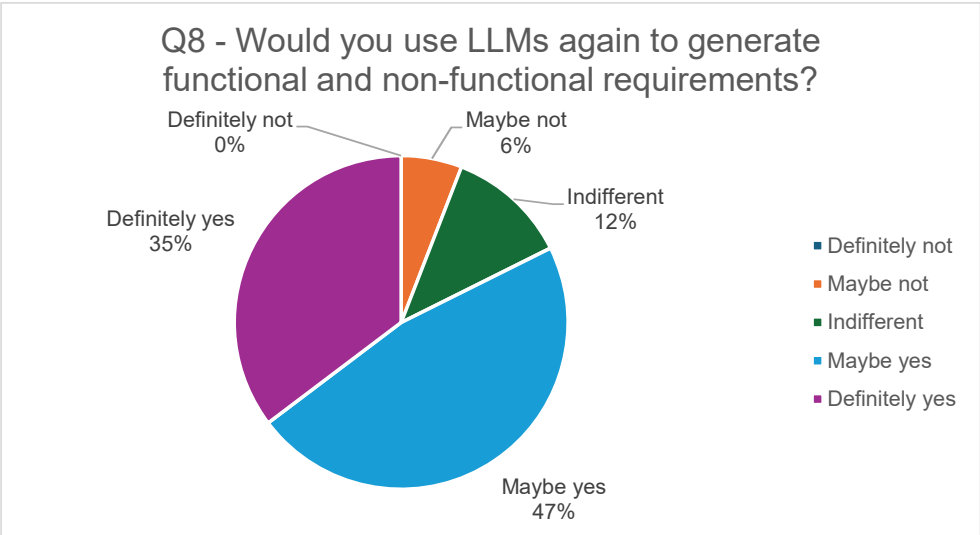
Figure 5- Were the recommendations for ensuring product quality appropriate for your project?



Source: Developed by the author (2025).

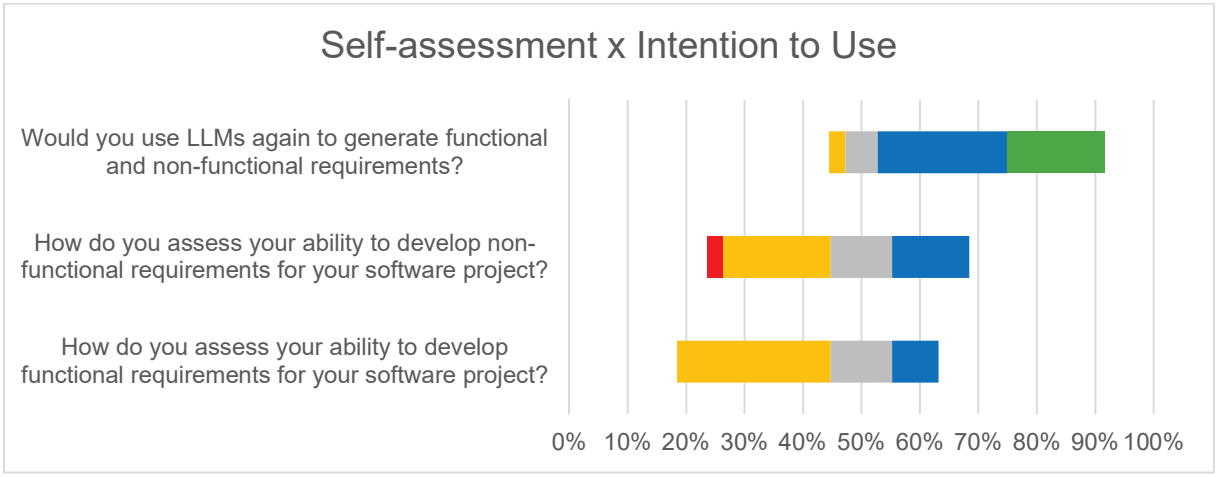
When comparing the intention to use LLMs for generating functional and non-functional requirements, it is evident that among the participants, the greater the difficulty in developing such requirements independently, the higher the interest in continuing to use LLMs as a support tool—suggesting that AI offers compensatory value in educational contexts. Similarly, when comparing the perceived quality of the generated requirements to reuse LLMs, a relationship emerges between the perceived usefulness of the outputs and the willingness to rely on LLMs for future requirement generation. An important observation arises when comparing students' ability to formulate requirements with their evaluation of the LLM outputs: students who report less difficulty in writing requirements tend to be more critical when assessing the AI-generated responses.

Figure 06- Would you use LLMs again to generate functional and non-functional requirements?



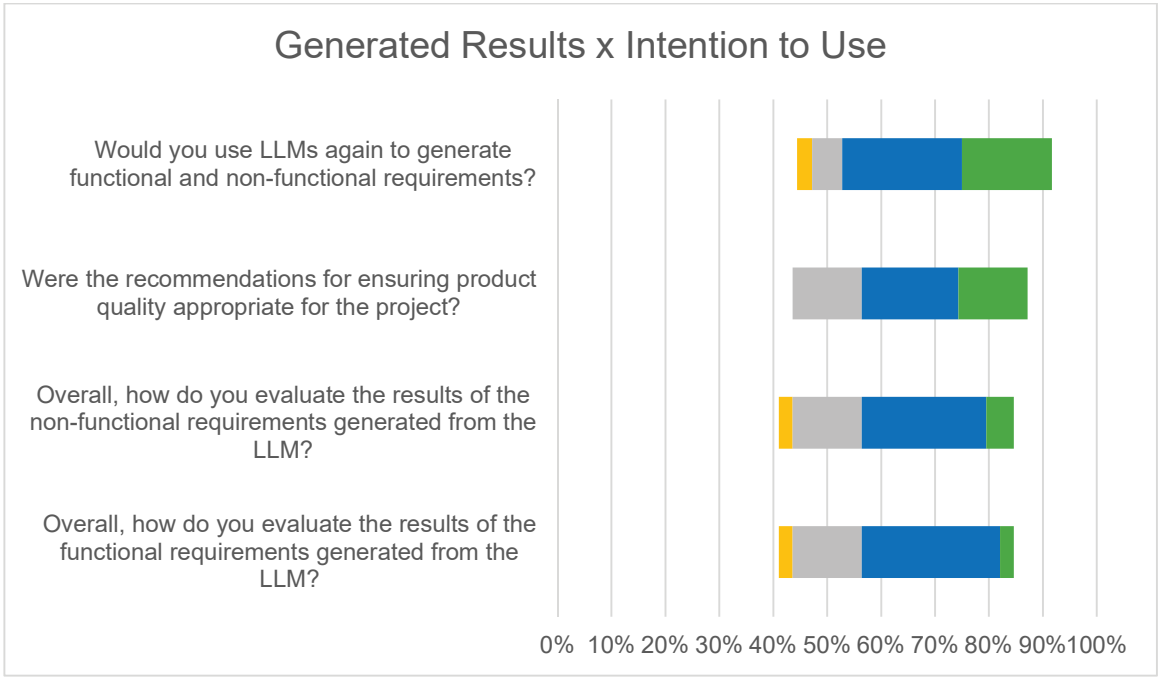
Source: Developed by the author (2025).

Figure 7 – Self-assessment x Intention to Use



Source: Developed by the author (2025).

Figure 8 – Generated Results vs Intention to Use



Source: Developed by the author (2025).

Table 3 – Relationship between Functional Capacity and Assessment of the results obtained

Functional Capacity	LLM Assessment Average	Standard Deviation
I have a lot of difficulty	3.5	0.6
I have some difficulty	3.2	0.7
Indifferent	3.0	0.5
I have some ease	2.7	0.6
I have ease	2.2	0.4

Source: Developed by the author (2025).

RQ1 – How do students assess the quality of the functional and non-functional requirements generated by LLMs based on inputs produced through Event Storming?

Two questions from the individual evaluation questionnaire (Questions 5 and 6) addressed the quality of the requirements generated. Regarding functional requirements, 58% of students stated that the results were adequate, though some adjustments were necessary. Additionally, 29.4% considered the results average but potentially useful for requirements analysis.

For non-functional requirements, the results were similar: 52.9% found them adequate with some caveats, while 29.4% considered the outcomes average but usable.

Given these findings, students generally viewed the requirements generated by the LLM as positive and valuable.

RQ2 – Does the use of Event Storming contribute to the creation of more effective prompts for requirement generation by LLMs?

Even without prior experience with Event Storming, most students achieved satisfactory results. The use of prompts, combined with the Big Picture, suggests that visual representations of events and commands can be effectively employed to support the generation of requirements by the LLM.

RQ3 – Does the combined use of Event Storming and LLMs foster the development of students' critical autonomy in requirements evaluation?

The experiment was designed to include a built-in evaluation phase of the generated results. Instead of simply copying the outputs produced by the LLM, students were asked to compare them to the requirements they had constructed themselves. This process enabled students to evaluate the quality of the AI-generated outputs based on their ability to formulate requirements, prompting them to identify which parts needed improvement. This approach facilitates the development of critical autonomy in evaluating software requirements.

5 CONCLUSION

The results of the study indicate that integrating the Event Storming technique with the use of Large Language Models (LLMs) shows significant potential for the software requirements elicitation process, particularly in educational contexts such as the classroom. Although most students had no prior experience with Event Storming, the presentation of the Big Picture provided effective input for the context of their projects, supporting the generation of both functional and non-functional requirements through the LLM. These outputs were, in the majority of cases, evaluated as adequate.

This finding suggests that Event Storming has the potential to serve as a form of visual documentation, contributing to the construction of more effective prompts and helping to minimize generic or inaccurate responses from the AI. From a pedagogical perspective, the activity stimulated students' critical autonomy—they did not passively accept the generated outputs but instead analyzed and reflected on their applicability, limitations, and points for revision.

This combined approach proved effective in promoting critical thinking about the use of generative AI tools, reinforcing their pedagogical value in Software Engineering education.

5.1 LESSONS LEARNED

The widespread adoption of LLMs has led to their broad—and at times indiscriminate—use. The accuracy of the information they generate can be highly questionable, which we believe should serve as a starting point for educational approaches that employ LLMs as support tools in knowledge construction.

While this experiment yielded promising results, a longer time frame for producing and critically evaluating the outputs would be necessary to explore scenarios that more closely resemble real-world contexts, such as changes in requirements throughout the development process.

Upon analyzing the questionnaire data, we observed that expanding the instrument to include more subjective items would have provided valuable insights, helping to guide future research directions.

5.2 FUTURE WORKS

The requirements generated during the experiment were generally well received by the students, despite their respective caveats. This suggests a possible reduction in hallucination rates, which could be further investigated in a more in-depth study. Repeating the experiment with other student samples is a valid and recommended course of action.

REFERENCES

ARORA, Chetan; GRUNDY, John; ABDELRAZEK, Mohamed. **Advancing requirements engineering through generative AI**: Assessing the role of llms. In: GENERATIVE AI FOR EFFECTIVE SOFTWARE DEVELOPMENT. Cham: Springer Nature Switzerland, 2024. p. 129-148.

ARVIDSSON, Simon; AXELL, Johan. **Prompt engineering guidelines for LLMs in Requirements Engineering**. 2023.

AUSUBEL, David P. **The psychology of meaningful verbal learning**. 1963.

AUSUBEL, David P.; HANESIAN, H. **Educational Psychology**: a cognitive view. New Cork. Holt, Reine Hart and Wilson, 1968.

BELZNER, Lenz; GABOR, Thomas; WIRSING, Martin. **Large language model assisted software engineering**: prospects, challenges, and a case study. In: INTERNATIONAL CONFERENCE ON BRIDGING THE GAP BETWEEN AI AND REALITY. Cham: Springer Nature Switzerland, 2023. p. 355-374.

BRANDOLINI, Alberto. Introducing event storming. blog, **Ziobrando's Lair**, v. 18, 2013.

CAROLI, P. **Lean inception**: how to align people and build the right product. Editora Caroli, 2018.

COPEI, Sebastian *et al*. Supporting Requirements Engineering and Development with Event Modeling-an Overview. **Modellierung**,, p. 145-154, 2022.

GRIMM, Valentin; RUBART, Jessica. **Modelling Gamified Systems with Event Storming Augmented by Spatial Hypertext**. In: PROCEEDINGS OF THE 4TH WORKSHOP ON HUMAN FACTORS IN HYPERTEXT. 2021. p. 3-10.

JACOB, Christo; KERRIGAN, Páraic; BASTOS, Marco. The chat-chamber effect: trusting the AI hallucination. **Big Data & Society**, v. 12, n. 1, p. 20539517241306345, 2025.

MASINI, Elcie F. Salzano; MOREIRA, Marcos Antonio. **Aprendizagem significativa**: condições para ocorrência e lacunas que levam a comprometimentos. Vetor Editora, 2023.

MATAS RUIZ, Guillermo; YAGÜE PANADERO, Agustín. **streamlining product inception and backlog management with large language models**. Available at SSRN 5036454.

NOVAK, Silvestre. **Educação a distância e racionalidade comunicativa**: a construção do entendimento na comunidade virtual de aprendizagem. 2010. Tese de Doutorado. Universidade Federal do Rio Grande do Sul.

ROCHA, Fabio G. et al. **Agile teaching practices**: using TDD and BDD in software development teaching. In: PROCEEDINGS OF THE XXXIII BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING. 2019. p. 279-288.

ROCHA, Fábio Gomes; MISRA, Sanjay; SOARES, Michel S. Guidelines for future agile methodologies and architecture reconciliation for software-intensive systems. **Electronics**, v. 12, n. 7, p. 1582, 2023.

ROCHA, Fabio Gomes; SABINO, Rosimeri Ferraz; ACIPRESTE, Ronald Henrique Leal. A metodologia Scrum como mobilizadora da práti-

ca pedagógica: um olhar sobre a engenharia de software. In: FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE (CBSOFT '15), 8. **Anais...** Belo Horizonte: Sociedade Brasileira de Computação, p. 12-13, 2015.

SIRICHAROEN, Waralak Vongdoiwang. Using empathy mapping in design thinking process for personas discovering. In: INTERNATIONAL CONFERENCE ON CONTEXT-AWARE SYSTEMS AND APPLICATIONS. Cham: Springer International Publishing, 2020. p. 182-191.

Recebido em 25 de julho de 2025
Aceito em 01 de novembro de 2025