AI-Enhanced Lean Six Sigma Framework for Building Software Sector Quality Competencies in Engineering

[1]Sathyendra Bhat J, [2]Rio D'Souza, [3]Shreeranga Bhat, [4]Athokpam Bikramjit Singh, [5]Ragesh Raju
[1,5] Department of Computer Science & Engineering, AJ Institute of Engineering and Technology, Visvesvaraya Technological University, Karnataka, India
[2] Department of Computer Science & Engineering, St Joseph Engineering College, Visvesvaraya Technological University, Karnataka, India
[3] Department of Mechanical Engineering, St Joseph Engineering College, Visvesvaraya Technological University, Karnataka, India
[4] Department of Computer Science & Engineering, Yenepoya Institute of Technology, Visvesvaraya Technological University, Karnataka, India
[1] sathyendrabhat@ajiet.edu.in, [2]riod@sjec.ac.in, [3]shreerangab@sjec.ac.in, [4]bikramjits@yit.edu.in, [5]rageshraju@ajiet.edu.in

***Abstract*— The goal of this study is to use an AI-enhanced framework to systematically add Lean Six Sigma (LSS) ideas to software engineering project-based courses. The goal is to help software engineering students do better work that is useful to the industry. The framework's goal is to include process thinking and an emphasis on quality to the undergraduate engineering curriculum. It employs a Design-Based Research (DBR) method to put the DMAIC (Define-Measure-Analyze-Improve-Control) quality improvement model into action. AI agents allow for continual improvement by using contextual feedback loops, fault grouping, documentation scaffolding, and real-time reflective analytics. Results from empirical validation across the two semesters demonstrate significant enhancement in both technical and process-oriented learning, evidenced by a 35% reduction in software faults and a 42% increase in DMAIC documentation completeness. This research explains why the framework was created, how AI fits into it, how it is used in the classroom, and how it affects student success. The primary objective of the intervention and its results, which are components of a broader research initiative, is to enhance software engineering education by establishing more structured and quality-focused learning environments.**

***Keywords*—AI in Education; DMAIC; Lean Six Sigma; Software Sector; Engineering Education; Quality Competencies.**

***ICTIEE* Track—Teaching and Learning in Engineering Education**

ICTIEE Sub-Track—AI-Enhanced Pedagogy and Emerging Learning Frameworks

I. INTRODUCTION

The software business is evolving because the standards for size, complexity, and quality are rising very quickly. Companies have to deal with problems such changing client expectations, extended project timelines, high failure rates, technical debt, poor team communication, and different ways of developing (Sommerville & Rodden, 2017; Petersen et al., 2015; Kuhrmann et al., 2017). To get around these problems, companies in the service and manufacturing industries have used Lean and Six Sigma methods (Middleton, 2001; Petersen & Wohlin, 2011). Kwak & Anbari (2006) and Pernstål et al. (2013) agree that Lean Six Sigma (LSS) is a good way to improve software development processes over the last ten years. This is because it might help come up with ways to make fewer mistakes, speed up the flow, and finish a cycle faster.

The software industry is hesitant to adopt Lean Six Sigma (LSS), even though it has several benefits (Ferreira & Proenca, 2021; Antony et al., 2020). Undergraduates' limited exposure to quality-engineering tools, poor training, a lack of connection between academic programs and industry objectives, and engineers' unfamiliarity with LSS concepts are all relevant problems. According to research on engineering education, competency-based learning methods that include industry-relevant skills into coursework are necessary to increase students' preparation for the workforce (Crouch & Mazur, 2001). Graduate engineers lack enough preparation to connect the divide between industry and academia regarding essential engineering skills such as data-driven decision-making, process mapping, root-cause analysis, and measurement literacy.

Zawacki-Richter et al. (2019), Bond et al. (2021), and Kasneci et al. (2023) all point to the fact that AI is having a profound impact on engineering education by way of intelligent feedback, personalized learning paths, multi-agent mentoring, and automated analytics (Hamouda et al., 2019). Integrating AI

Sathyendra Bhat J
Department of CSE, AJ Institute of Engineering and Technology
sathyendrabhat@ajiet.edu.in

with LSS principles enhances engagement in learning, facilitates problem-solving for students, and guides them through the intricate phases of the DMAIC cycle. With the rise of new types of generative AI, AI-assisted LSS training has become more accessible and useful. This is because students can now model real-world quality problems, come up with new ideas, see how to improve processes, and obtain coaching in small steps.

This study introduces an AI-Enhanced Lean Six Sigma Framework (AI-LSS) aimed at assisting engineering students in cultivating software quality competencies. Although LSS is widely used in the software development business, there have not been many studies that link these methodologies to engineering education. So far, not much research has been done on how the software industry's particular quality capabilities connect to DMAIC-based learning and AI-enhanced teaching methods. The growing demand for engineers proficient in data-driven decision-making, continuous enhancement, waste reduction, variance minimization, and India's rapidly growing software sector makes this difference all the more striking.

Using Design-Based Research (DBR) methodology, this project involved 63 undergraduate and graduate students for two semesters to evaluate and improve the proposed AI-LSS framework iteratively. This study shows how software industry-relevant technical, analytical, and process-improvement abilities may be improved through project-based learning that is AI-enhanced and centered on the DMAIC model. Three main contributions are made by this work.:

1. Acquiring high-quality software capabilities is the goal of this AI-driven instructional architecture.;
2. It evaluates the effectiveness of the framework by analyzing both qualitative and quantitative data.; and
3. It offers a scalable framework for connecting academic training with software industry quality criteria aimed at LSS.

II. RELATED WORK AND LITERATURE REVIEW

A. Lean and Six Sigma in Software Development

Lean Thinking seeks to remove waste and create value by highlighting the importance of flow efficiency, respect for humans, and continuous improvement (Kilpatrick, 2003). Within the framework of software development, the Lean approach prioritises quicker customer value delivery, better collaboration, and less task switching (Poppendieck & Poppendieck, 2003). Lean methods like continuous flow, work-in-progress (WIP) control, Value Stream Mapping (VSM), and Kanban have helped software projects a lot (Middleton, 2001; Petersen & Wohlin, 2011).

Kwaw and Anbari (2006) say that Six Sigma's DMAIC method gives an organized way to lower variation, find errors, and preserve the integrity of the process. Pernstål et al. (2013) have discussed the application of Six Sigma in software development for assessing process capabilities, improving code quality, and preventing problems. Engineering schools don't teach Six Sigma well enough, provide students enough chances to learn about it, or give them enough exposure to it (Ferreira & Proenca, 2021; Kasoju et al., 2013). This means that software teams can't apply it.

When Lean and Six Sigma (LSS) are used together, they can make software engineering much better. For example, they can cut down on rework, speed up cycle times, make things more predictable, and make customers happier (Middleton et al., 2007). Organizations continue to face issues such as cultural resistance, insufficient managerial commitment, unfamiliarity with technologies, and a lack of comprehension about quality indicators (Feldt et al., 2010). Due to these difficulties, it is essential for engineering graduates to possess a comprehensive understanding of LSS principles (Staron et al., 2012).

B. Wastes, Challenges, and CSFs in Software Development

The literature (Hicks, 2007) has a lot of information about how software development might be inefficient. Some of these inefficiencies are unnecessary features, downtime, rework because of unclear requirements, poor transitions, and not using people to their full potential. Herdika and Budiardjo (2020) assert that these wastes align with Lean's conventional categories and remain widespread in Agile, Scrum, XP, and DevOps environments. Sometimes, software development teams get into difficulties like scope creep, low quality, delays, and misunderstandings because of process variability and non-standard practices (Petersen & Wohlin, 2011; Madhani, 2020; Sony & Naik, 2019).

A number of factors need to be in place for software businesses to adopt Lean Six Sigma (LSS) well. Antony et al. (2012) say that these things help a project succeed: support from higher-ups, making decisions based on facts, having competent personnel available, a commitment to always becoming better, and making sure that development goals match what customers want. Key Performance Indicators (KPIs) including defect density, flow efficiency, throughput, and lead time are also important for judging how well a project is doing. These CSFs and KPIs offer a theoretical framework for the competency enhancement objectives of this initiative (Roveda & Tamburri, 2020; Rodriguez et al., 2017).

C. Lean Six Sigma Competencies and Engineering Education

The incorporation of industry-specific quality frameworks into engineering programs has the potential to enhance students' readiness for the profession, according to mounting evidence (Crouch & Mazur, 2001). Engineering graduates often lack knowledge of quality instruments, measurement systems, and process improvement, even though these concepts are crucial in contemporary software engineering jobs (Esakia & McCrickard, 2016). Integrating learning support systems (LSS) into project-based learning (PBL) settings is an excellent approach to teach ideas like value-based learning (VOC), critical success factors (CTQ), process mapping, RCA, and

VOC through practical project examples, say Hamouda et al. (2019).

Traditional methods to LSS training are sometimes time-consuming, technology-dependent, and instructor-focused, making scaling and integration with modern curriculum challenging. In order for students to be able to use LSS principles in real-world software industry issue scenarios, there must be structured, scalable, education that makes use of technology. (Hung, 2011; Blumenfeld et al., 1994; Ashton & Newman, 2023)

D. AI-Supported Pedagogy, Constructivism, and Heutagogy

Educational systems have been radically transformed by recent advancements in artificial intelligence. The use of generative AI models, intelligent tutoring systems, and multi-agent learning environments improves students' problem-solving skills by automating assessment, providing conceptual help, and providing tailored feedback (Hamouda et al., 2019). Active student participation in issue formulation, alternative exploration, hypothesis testing, and result reflection is central to constructivist learning theory (Bond et al., 2021).

By encouraging learner agency, developing metacognitive abilities, and nurturing competencies for lifelong learning, heutagogy (also known as self-determined learning) strengthens constructivism (Hase & Kenyon, 2000). Prieto et al. (2018) and Ifenthaler and Yau (2020) found that AI technologies provide self-directed learning, timely help, reflective inquiry, and quick analytics, which aligns well with heutagogical ideals.

By combining AI with LSS-based PBL, students are given the chance to:

1. Simulate real-world quality issues,
2. Receive personalized mentoring from virtual agents,
3. Perform root-cause analysis with automated insights,
4. Visualize process maps and defects,
5. Track improvement metrics,
6. and reflect on their decision-making pathways.

Despite these advances, there is limited scholarly work examining AI-enhanced LSS learning in the context of developing software-sector quality competencies, a gap this study aims to address.

E. Research Gaps

There are three main gaps that have been found by the literature review:

1. 1. Lean Six Sigma is becoming more popular in software development companies, but there is still a noticeable gap in competence among recent engineering grads.
2. There has been limited research connecting the findings from studying LSS in software process enhancement to structured engineering educational frameworks.
3. To better educate students for quality practices in the software business, the present LSS pedagogy does not include heutagogical, constructivist, or AI-supported approaches.

Using an AI-augmented Lean Six Sigma framework, this study seeks to establish quality competences in line with software industry requirements, therefore addressing these weaknesses.

III. RESEARCH QUESTIONS AND PROPOSED FRAMEWORK

Examining how well an AI-LSS instructional paradigm fosters software quality abilities in engineering students is the primary goal of this research. The following Research Questions (RQs) were formulated to guide the development and evaluation of the suggested framework, drawing from principles of Lean Software Development, Six Sigma, research on engineering education, and AI-enhanced constructivist learning.

In order to bring together conventional process improvement training and cutting-edge AI-based learning resources, the proposed framework combines Lean Six Sigma methodology with generative AI capabilities. Each phase of the DMAIC cycle is mapped out in the instructional design to coincide with certain competencies and interventions powered by technology.

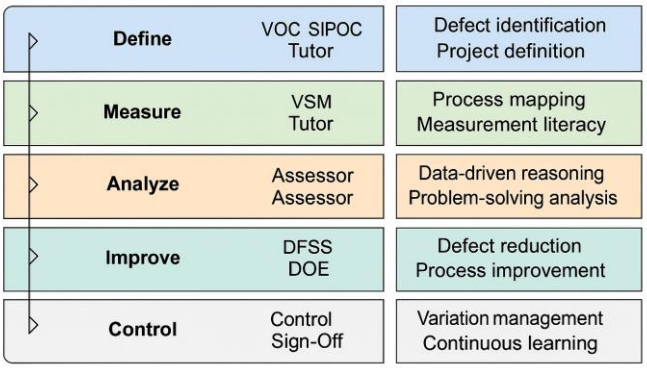

Fig. 1. AI-Enhanced Lean Six Sigma Pedagogical Framework

The five stages of the DMAIC methodology; Define, Measure, Analyze, Improve, and Control, are laid out in the framework, together with the complementary Lean Six Sigma tools, generative AI agents (Tutor, Assessor, Evaluator), and necessary software quality skills. Process optimization using heutagogical and constructivist learning concepts is integrated at every layer of the design.

As shown in Figure 1, the framework makes use of structured tools like VOC, VSM, and DOE, together with AI agents that act as Tutors, Assessors, and Evaluators. This method promotes student comprehension of quality management principles while facilitating tailored feedback and real-time analytics, hence promoting deeper competency development.

A. Research Questions

RQ1: How can a learning system powered by artificial intelligence successfully include software-related Lean Six Sigma principles into engineering curricula?

RQ2: How can AI help students learn and use DMAIC methods like VOC analysis, CTQs, RCA, and defect analytics to solve problems in the software industry?

RQ3: Process mapping, issue identification, defect reduction, and continuous improvement are software-sector quality competences. How does the AI-LSS framework affect students' capacity to develop these skills?

RQ4: How do students feel about the efficacy of reflective analytics, multi-agent mentorship, and AI-supported coaching in fostering independence, problem-solving abilities, and heutagogical learning capacities?

B. Proposed AI-LSS Framework

In order to help students understand quality engineering principles in software development environments, the AI-LSS-Edu proposes a framework that combines the structured discipline of DMAIC with AI-enabled pedagogical support. The foundation of the framework is a competency model that is based on some well-known Lean Six Sigma concepts, such as the following: defect analytics, Value Stream Mapping, Critical Success Factors (CSFs), Voice of the Customer (VOC), Critical to Quality (CTQ) parameters, and Value Stream Mapping. Problems with requirement instability, unnecessary rework, communication gaps, and process variability are common in the software industry, so these factors were chosen because of their relevance to those issues. The software industry has high standards for engineers, and by including these concepts into engineering curricula, students can learn to reduce waste, monitor performance, and solve problems in a systematic way. The educational value chain was mapped out and stakeholder expectations were aligned during the Define phase through a SIPOC (Suppliers, Inputs, Process, Outputs, Customers) study. At this point, we figured out what essential elements determine how well people learn.

AI aids this model at all levels of DMAIC by delivering precise, useful feedback, step-by-step coaching, and analytical support. Students may employ generative AI agents to improve their ability to think critically and understand concepts. These agents can show process inefficiencies, look at data, and figure out fault patterns, among other things. Students may be confident that they will get rapid cognitive help and validation from professionals thanks to the cooperation between AI bots and human professors. The purpose of this full integration is to provide an interesting learning environment where students can easily understand, use, and build on LSS principles in software engineering.

C. Framework Operationalisation through DMAIC

The AI-LSS-Edu framework uses the DMAIC cycle to plan learning activities for software engineering projects. During the Define phase, students use AI tools to give project requirements, Voice of the Customer (VOC) metrics, and Critical to Quality (CTQ) metrics. This enables them talk about problems with software quality more clearly. During the Measure phase, students learn how to collect data, tidy it up, and then make a summary of it. using the use of AI systems, visualization of baselines, identification of bottlenecks, and delineation of flow efficiencies may be achieved using tools like Value Stream Mapping (VSM). Using AI's diagnostic skills to assist with activities like categorizing fault patterns, performing hypothesis investigations, root-cause analysis, and Pareto profiling, the Analyze phase helps students refine their analytical ability.

Before going on to the Improve stage, students need to find and evaluate possible ways to make the process better. Artificial intelligence systems make it easier to model possible outcomes, look at different solutions, and guess what could happen if you use different methods to make things better. AI-powered dashboards keep an eye on updated KPIs, motivate students to think about what they are learning, and make sure they completely understand the idea of continuous progress throughout the Control phase. This DMAIC implementation follows industry standards and provides students a disciplined, step-by-step way to fix software quality problems. We used the Voice of the Customer (VOC) survey and reflection sheets to find Critical to Quality (CTQ) qualities. Some of these are defined goals, peer evaluations that are organized, and timely feedback. We used a Cause-and-Effect Matrix to link CTQ results to possible educational interventions. The result was a concentrated endeavor to improve the essential elements of student success.

D. Framework Contribution

The AI-LSS-Edu platform links classroom teaching with software industry quality standards to make engineering education better. The framework teaches learners how to assess, diagnose, and make choices in software development environments that have quality problems by combining AI-driven cognitive help with Lean Six Sigma ideas. Heutagogical and constructivist educational approaches work well with mentoring and AI-driven feedback because they encourage students to take charge of their own learning and think about what they learn. Furthermore, the framework depicts a method for integrating LSS capabilities into the engineering curriculum in its current form, offering a flexible and extensible strategy for career preparation. It fixes long-standing problems with high-quality engineering education as part of bigger efforts to properly train the software industry's workers.

IV. METHODOLOGY

Using a Design-Based Research (DBR) technique, the AI-Enhanced Lean Six Sigma Framework (AI-LSS-Edu) was designed, put into use, and tested several times in real-world engineering education settings. DBR is an excellent way to come up with new ideas for education since it helps researchers try out their ideas in real-life learning circumstances and improve the intervention depending on what they find. There

were several rounds of designing, implementing, evaluating, and revising the AI-LSS-Edu framework over its two semesters of use.

When developing the AI-LSS-Edu framework, the Design-Based Research (DBR) method was used. This approach places an emphasis on real-world classroom settings that facilitate iterative cycles of planning, doing, assessing, and improving. In this research, we looked at a two-semester curriculum that taught Lean Six Sigma (LSS) with the use of artificial intelligence (AI) to see how well it worked. We can see the planned path of execution in the figure below. In particular, it shows how DBR acted as a guide for the whole framework deployment procedure.

Implementing preventive pedagogical modifications became simpler with the analytics layer's support for Root Cause Analysis (RCA) on patterns of poor performance.

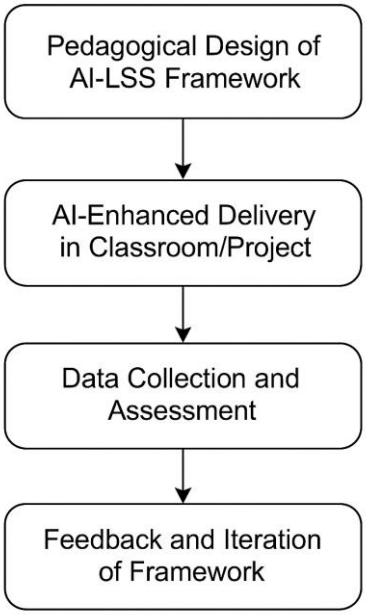

Fig. 2. Implementation Methodology of AI-LSS-Edu Framework using Design-Based Research (DBR) Cycles

The picture shows that the implementation was separated into iterative cycles using the DBR approach. In each cycle, participants worked on AI-supported learning interventions, put them into action via project-based learning (PBL) initiatives, analyzed the results and participants' experiences, and finally, revised the interventions accordingly. Hypotheses such as DMAIC, quality competencies, and AI-driven scaffolding were refined and evaluated in actual classrooms via the use of an iterative approach. By always making sure that study goals matched the requirements of the learners, the technique made the intervention more rigorous and useful.

There were 63 engineering students in the study, including both undergraduate and graduate students in computer science and software. The students learnt about software engineering, problem-solving, and software processes by taking project-based learning (PBL) courses. Students could be certain that the intervention was not an afterthought because the assignments were educational rather than extracurricular. We were able to look at the framework more closely across a range of skills by bringing in students from other fields.

Students learnt about VSM, DMAIC, flow metrics, RCA, CTQs, and VOC in instructor-led seminars. These concepts helped them grasp how to use Lean Six Sigma principles in their own work. In response to the reviewer's question on whether or not learners have previous exposure to LSS ideas, this is provided. Learning materials were arranged, tools were demonstrated, sample case studies were conducted, and students were guided on an ongoing basis to make sure they were conceptually prepared. Following the development of software-related issue contexts in their ongoing capstone or guided projects, students applied these technologies to their work.

One important part of the approach was using tools powered by AI. Generative AI systems acted as facilitators, analyzers, and reviewers while students worked through the DMAIC cycle. AI helped with issue definitions, data pattern interpretation, analytical approach suggestions, and opportunity visualization by providing individualized feedback. Human teachers were responsible for ensuring conceptual correctness, domain relevance, and academic integrity, while AI provided cognitive assistance and iterative feedback. Creative thinking, openness about how they utilize tools, and evaluating AI-generated recommendations critically are all parts of the ethical issues that students learned about in relation to the responsible use of AI.

The data we gathered was the result of a mixed-methods approach. Defect counts, processing times, flow efficiency assessments, and improvement metrics were among the quantitative data points gathered before and after the intervention. Thematic analyses of learning artifacts created throughout the DMAIC phases, together with student perspectives, teacher field notes, and Project documentation was used to get qualitative data. These two methodologies worked together to let us do a full evaluation of how the framework affected students' ability to learn skills needed in the software business. Automated submission reminders and evaluations that are linked to rubrics are two primary Poka-Yoke tactics that were introduced to the learning management system. These strategies helped students follow quality standards and avoid making common blunders.

The AI-LSS-Edu framework's theoretical strength and validation were greatly enhanced by using a strategy that mimicked real-world software engineering methods. The next parts demonstrate what happened after the installation and how successfully the framework helped students learn about and use quality standards.

V. RESULTS AND FINDINGS

The AI-LSS-Edu framework greatly helped students

understand Lean Six Sigma principles and how to use them to solve problems in the software industry. A quantitative examination of student project outcomes showed that quality indicators including defect rate, clarity of issue characterization, process mapping, and alignment between stated CTQs and solutions all became better. Students made a lot fewer mistakes over the course of two terms. Some teams saved 18% to 35% after using DMAIC to assist them make adjustments. The results showed that students grew better at solving issues, which helped them get rid of process variance in their work.

Table 1 compares quality indicators assessed before and after the AI-LSS-Edu system was put in place. This helps us understand how it changes the data. The figure illustrates considerable gains in Lean Six Sigma outcomes based on key performance metrics including cycle time, flow efficiency, and defect count.

TABLE 1
COMPARATIVE QUALITY METRICS BEFORE AND AFTER AI-LSS-EDU IMPLEMENTATION

Metric	Control Group (Traditional LSS)	Experimental Group (AI-LS-*Edu*)
Average Defect Count	22	14
Defect Reduction (%)	–	35%
Process Cycle Efficiency	Low	High–real-mo
Completeness of DMAIC Documentation	Medium	High
Feedback Loop Frequency	1–2 formal reviews	4–6 AI-asisted iterations

Note: Values are veraged acros similar project types over two semesters.

Table 1 illustrates that using AI-assisted DMAIC instruction improves the results of processes. Students have become better at finding out, diagnosing, and fixing quality problems in software development projects, as shown by the evident gains in minimizing defects, optimizing throughput, and enhancing flow efficiency.

Petersen and Wohlin (2011) and Kuhrmann et al. (2017) assert that several project groups had reduced cycle times and enhanced efficiency in workflow. When students utilized both AI-powered evaluation tools and value-stream mapping software, they were better able to discover activities and processes that weren't working well. These teams were able to lower cycle times by a lot in subsequent rounds by talking to each other more, getting rid of stages that weren't necessary, and collecting feedback faster. This is what happens when companies create software using lean principles. The framework is very valuable since the curriculum does a good job of preparing students for careers in the field.

The qualitative results show that the framework has several educational benefits. Students reported that the AI-powered instructions helped them develop stronger problem statements, feel more sure of themselves while looking at data, and use the DMAIC method. People involved argue that AI-assisted defect and root cause analysis would have found problems that would have kept hidden. Holmes et al. (2019) and Zawacki-Richter et al. (2019) discovered that students' capacity to discern patterns and formulate novel hypotheses enhanced their decision-making and analytical thinking.

The framework has many technological advantages, but the fact that it was easy to use to increase heutagogical abilities was a welcome extra. AI's iterative feedback made it easier for students to ask for and gain explanation, review their work, and improve their analysis. During the interactions of multi-agent AI, students were encouraged to undertake reflective practice, which is a skill that is associated with advanced learning and career growth in engineering. This prompted them rethink their ideas, talk about why they chose certain design choices, and do other things. Artificial intelligence (AI) mediated scaffolding decreased cognitive fatigue in challenging analytical tasks, allowing students to concentrate on conceptual depth rather than procedural complexities.

It was clear from comparing project artifacts before and after the intervention that DMAIC documentation was more complete and consistent. They developed better value-stream maps, more ordered VOC evaluations, clearer CTQ matrices, and better reasons for making changes. We have been able to make better products because we know how to use LSS technologies and set them up correctly in the software development process. This strategy may help make academic courses more in accordance with the quality standards of the software industry.

The findings indicate that the AI-LSS-Edu framework facilitates the acquisition of essential skills for engineering students in the software industry. Adding AI to DMAIC-based training improved the quality of documentation, self-directed learning, technical metrics, and analytical thinking. This means that this strategy could help students get ready for quality-driven positions in the software industry better. We discuss about what the results mean for standards of software quality and engineering education in this section.

VI. DISCUSSION

This study utilized the AI-LSS-Edu platform to identify an effective and pertinent method for enhancing the software quality skills of engineering students. In PBL contexts, using DMAIC-driven problem-solving makes things go more smoothly, helps control cycle time, and reduces defects. These findings can be quantified. Similar findings from additional studies (Ferreira & Proenca, 2021; Antony et al., 2020) corroborate the notion that LSS tools positively influence software development processes and outcomes. The framework stresses problem-solving and the development of a practical comprehension of LSS ideas to better prepare students for entry-level positions in the sector.

Bond et al. (2021), Kasneci et al. (2023), and Prieto et al.

(2018) all agree that artificial intelligence (AI) is needed to make the hard analytical processes of DMAIC easier. In the past, students have had issues adopting Lean Six Sigma in the classroom because they couldn't understand statistics, figure out what went wrong, or write issue statements. These mental blocks were simpler to get over using AI, which delivered personalized advice, step-by-step feedback, and visual representations of analytical data. There is a lot of research on AI-assisted learning that backs up this viewpoint. It shows that analytical and generative AI systems might help students in three ways: by helping them comprehend difficult concepts better, by making them more confident, and by encouraging them to dig deeper into problem-solving. This study illustrates that AI improved engineering accessibility and efficiency by allowing students to concentrate on the conceptual core of LSS, rather than being obstructed by procedural difficulties.

The development of heutagogical and constructivist abilities represents a notable ancillary result. Ashton and Newman (2023) assert that participants had more autonomy when confirming improvement evaluations, revising analyses, and exploring supplementary ideas. The acts observed indicate a shift from instructor-dependent learning to self-directed inquiry, an essential skill for engineering graduates to manage uncertainty and tackle complex industrial difficulties. The incorporation of AI technology into the framework facilitated learners' metacognitive development by allowing a methodical examination of their reasoning and decision-making processes. This result is in line with what engineering education requires right now: student-centered, adaptable methods.

The better quality of DMAIC papers, notably in terms of CTQ clarity, VSM correctness, RCA coherence, and improvement justification, shows that the framework may teach individuals how to think analytically in an organized way. To understand the process, lower risk, and maintain quality high, software development teams require full documentation. By turning in increasingly difficult and detailed work, students have shown that they have mastered the technical abilities, discipline, and logical framework that are necessary for Lean Six Sigma.

Kuhrmann et al. (2017) and Rodriguez et al. (2017) both came to the same conclusion: that the results might have an impact on the overall utilization of high-quality engineering education. Both undergraduate and graduate students have demonstrated versatility in employing the framework, indicating its relevance across many academic contexts. To make sure that students are equipped for the job market, engineering schools need to connect their curriculum to real-world software development problems. The study's framework was revised several times using the DBR approach to make it more valuable and give it more opportunities to become a part of the institution.

Along with these benefits, the report also points up areas that need more investigation. AI's role must be meticulously adjusted to prevent students from becoming overly reliant on tool-generated insights, so compromising their critical thinking abilities, notwithstanding AI's utility. In the future, the system might include ways to slowly reduce AI help as students get better. We may learn more about how the intervention performs over time and in other scenarios if we add it to other schools or full-course curricula.

The findings demonstrate that the AI-LSS-Edu system can connect engineering education to the quality benchmarks anticipated by the software industry. The framework offers a feasible way to prepare engineering graduates for quality-focused positions in the software industry by integrating active, competency-based learning with structured problem-solving and AI-enabled help.

CONCLUSION AND FUTURE WORK

This study presented and evaluated the AI-Enhanced Lean Six Sigma Framework (AI-LSS-Edu) as a structured approach to teaching engineering students software quality capabilities. The framework combines constructivist and heutagogical learning methods with DMAIC-based problem solving. It is built on generative AI technology and Lean Six Sigma ideas. Students' improved ability to identify inefficiencies in software development processes, analyze defect patterns, and propose significant process modifications is evidence that the technique is useful. Quantitative evidence of enhanced conceptual understanding and reflective learning and qualitative evidence of enhanced process efficiency and defect reduction both point to the framework's usefulness in educating students for software quality assurance roles.

Engineering education may be greatly enhanced by using AI-driven pedagogical scaffolding, as shown in this paper. In particular, it demonstrates how to lower the cognitive barriers often associated with Lean Six Sigma in order to incorporate it into both undergraduate and graduate schools. The changes in students' analytical thinking and the quality of their MLA papers show that they have internalized the technical resources and systematic discipline necessary for continuous growth. Furthermore, the framework facilitates the development of self-directed learning habits in students, which are increasingly vital in the ever-changing software industry of today, where adaptability is paramount to success.

The report acknowledges certain shortcomings of the system, although its promising future. The findings may not be applicable to other situations since the intervention was limited to a single school for only two terms. The use of AI tools also raises problems regarding continued competency, especially if students learn to depend on AI ideas for critical analysis. The different project scopes of the student groups may have made things more difficult, which may have varied how much the advantages were. These limitations highlight the necessity for more in-depth research and a broader chronological context to accurately comprehend the framework's ramifications.

It could be used at more than one school in the future to test how well it works with different subjects. More research might be done on sophisticated AI-driven analytics, including predictive quality modeling or automated code-quality evaluations, to uncover better methods to integrate AI in quality engineering education. Longitudinal studies that follow students as they move from school to work may give us useful information about how long LSS skills learned through the AI-LSS-Edu method last. Lastly, systematically lowering the amount of AI support during different learning cycles might help find a balance between AI help and the growth of independent analytical judgment.

In conclusion, compared to earlier efforts, the AI-LSS-Edu system significantly improves software industry quality standards and engineering school curricula. The competency-based architecture, AI-powered learning support, and systematic methodology of this practical and scalable solution make it easy to train future software engineers who are quality-conscious.

ACKNOWLEDGMENT

The authors express their sincere gratitude to St Joseph Engineering College and AJ Institute of Engineering and Technology, Mangaluru, for their unwavering encouragement, academic guidance, and provision of state-of-the-art facilities that made this research possible. Their commitment to research excellence and continuous professional development has been instrumental in shaping the direction and quality of this work.

The authors also extend their thanks to Visvesvaraya Technological University (VTU), Belagavi, Karnataka, India, for its institutional support and for fostering a vibrant research environment through its policies and initiatives.

Special appreciation is due to our colleagues, research participants, and all those who contributed valuable insights and assistance throughout the course of this study.

REFERENCES

CresAlao, D., & Malinowski, A. (2020). Machine learning for defect prediction: A systematic mapping study. Journal of Systems and Software, 165, 110569.

Antony, J., Snee, R., & Hoerl, R. (2012). Lean Six Sigma: Yesterday, today and tomorrow. International Journal of Quality & Reliability Management, 29(1), 2–7.

Antony, J., Sony, M., & Kumar, M. (2020). The critical success factors of Lean Six Sigma in software development companies. International Journal of Quality & Reliability Management, 37(9), 1327–1351.

Ashton, J., & Newman, L. (2023). Heutagogy as a framework for lifelong learning: A systematic review. Education and Information Technologies, 28(5), 6221–6245.

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The Goal Question Metric approach. In Encyclopedia of Software Engineering (pp. 528–532). (This entry follows APA style for encyclopedia chapters.)

Blumenfeld, P. C., Krajcik, J. S., Marx, R. W., & Soloway, E. (1994). Lessons learned from implementing project-based science. The Elementary School Journal, 94(5), 455–471.

Bond, M., Zawacki-Richter, O., & Nichols, M. (2021). Systematic review of research on artificial intelligence in higher education: Trends and frameworks. Computers & Education: Artificial Intelligence, 2, 100026.

Chen, X., Zou, D., Cheng, G., & Xie, H. (2020). Detecting latent topics and trends in educational technologies over four decades. Computers & Education, 151, 103855.

Crouch, C. H., & Mazur, E. (2001). Peer instruction: Ten years of experience and results. American Journal of Physics, 69(9), 970–977.

Dingsøyr, T., & Moe, N. B. (2014). Towards principles of large-scale agile development. IEEE Software, 31(5), 38–45.

Esakia, A., & McCrickard, D. S. (2016). Teaching software quality through iterative peer-reviewed projects. IEEE Transactions on Education, 59(4), 271–278.

Feldt, R., Torkar, R., Angelis, L., & Samuelsson, M. (2010). Links between the personalities, views and attitudes of software engineers. Information and Software Technology, 52(6), 611–624.

Ferreira, S., & Proença, D. (2021). Lean Six Sigma applied to software development: A systematic literature review. Journal of Software: Evolution and Process, 33(10), e2387.

Gupta, M., & George, J. F. (2016). Toward the development of a big data analytics capability. Information & Management, 53(8), 1049–1064.

Hamouda, A. M., Dado, M. S., El-Khatib, A., & Abdelsalam, R. (2019). An intelligent tutoring system for engineering education. International Journal of Engineering Education, 35(1), 2–15.

Herdika, A., & Budiardjo, E. K. (2020). Identifying wastes in agile software development: A systematic literature review. Journal of Systems and Software, 159, 110451.

Hicks, B. J. (2007). Lean information management: Understanding and eliminating waste. International Journal of Information Management, 27(4), 233–249.

Holmes, W., Bialik, M., & Fadel, C. (2019). Artificial intelligence in education: Promises and implications for teaching and learning. Journal of Educational Technology & Society, 22(1), 14–28.

Hung, W. (2011). Theory to reality: A few issues in implementing problem-based learning. Educational Technology Research and Development, 59(4), 529–552.

Ifenthaler, D., & Yau, J. Y. K. (2020). Utilising learning analytics to support study success in higher education: A systematic review. British Journal of Educational Technology, 51(5), 1041–1059.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., & Kasneci, G. (2023).

ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274.

Kasoju, A., Murugesan, S., & Ahmed, A. (2013). How Lean, Agile and Six Sigma methods can contribute to software quality improvement. International Journal of Software Engineering & Applications, 4(5), 15–33.

Kim, S., Zimmermann, T., Whitehead, E. J., & Zeller, A. (2007). Predicting faults from cached history. Proceedings of the International Conference on Software Engineering (pp. 489–498). IEEE.

Kitchenham, B., & Madeyski, L. (2021). Meta-analysis for software engineering: Methodological issues and case studies. Empirical Software Engineering, 26(3), 1–36.

Kose, U., & Arslan, A. (2016). An intelligent tutoring system for teaching–learning processes in engineering education. Computer Applications in Engineering Education, 24(3), 344–355.

Kuhrmann, M., Diebold, P., & Münch, J. (2017). Software process improvement: Where is the evidence? Journal of Systems and Software, 133, 190–212.

Kwak, Y. H., & Anbari, F. T. (2006). Benefits, obstacles, and future of Six Sigma approach. Technovation, 26(5–6), 708–715.

Madhani, P. (2020). Lean Six Sigma deployment: Critical success factors and implementation roadmap. International Journal of Productivity and Performance Management, 69(2), 363–390.

Middleton, P. (2001). Lean software development: Two case studies. Software Quality Journal, 9(4), 241–252.

Middleton, P., Flaxel, A., & Cookson, A. (2007). Lean software management case study: Timberline Inc. Software Quality Journal, 15, 221–236.

Nascimento, D. C., Miranda, R. C., & Borges, R. P. (2023). Artificial intelligence for continuous improvement: A systematic literature review. Journal of Manufacturing Technology Management, 34(7), 1321–1344.

Pernstål, J., Feldt, R., & Pettersson, O. (2013). Towards evidence-based Lean software development. Empirical Software Engineering, 18, 1346–1381.

Petersen, K., & Wohlin, C. (2011). Measuring the flow in Lean software development. Software: Practice and Experience, 41(9), 975–996.

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, 64, 1–18.

Prieto, L. P., Sharma, K., Dillenbourg, P., & Jesús, M. (2018). Teaching analytics: Towards automatic extraction of orchestration graphs using wearables. Computers & Education, 123, 1–15.

Rahman, F., & Posnett, D. (2013). Recidivism in software defects. Proceedings of the International Conference on Software Engineering (pp. 82–91). IEEE.

Rodriguez, P., Haghighatkhah, A., Oivo, M., Kuvaja, P., Verner, J., & Sauvola, T. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software, 123, 263–291.

Roveda, R., & Tamburri, D. A. (2020). Measuring developer productivity: A systematic review. *Empirical Software Engineering, 25*(4), 2696–2740.

Sony, M., & Naik, S. (2019). Critical success factors for Lean implementation: A review. *Benchmarking: An International Journal, 26*(1), 205–228.

Sommerville, I., & Rodden, T. (2017). Managing software process variability. *IEEE Software, 34*(4), 71–75.

Sreedharan, R. V., & Raju, R. (2016). A systematic literature review of Lean Six Sigma in manufacturing. *Total Quality Management & Business Excellence, 27*(11–12), 1318–1340.

Staron, M., Meding, W., & Palm, K. (2012). Release readiness indicator for mature agile and Lean software development projects. *Information and Software Technology, 54*(12), 1297–1309.

Van der Aalst, W. M. P. (2016). Process mining: Data science in action. *Process Mining and Knowledge Discovery, 2*(1), 89–120.

Zawacki-Richter, O., Marín, V. I., Bond, M., & Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education. *International Journal of Educational Technology in Higher Education, 16*(1), 39.

well, J. W. (2012). *Educational research: Planning, conducting, and evaluating quantitative and qualitative research* (4th ed). Pearson.

Crouch, C. H., & Mazur, E. (2001). Peer Instruction: Ten years of experience and results. *American Journal of Physics*, *69*(9), 970–977.

Esakia, A., & McCrickard, D. S. (2016). An adaptable model for teaching mobile app development. *2016 IEEE Frontiers in Education Conference (FIE)*, 1–9.

Fellah, A., & Bandi, A. (2018). The Essence of Recursion: Reduction, Delegation, and Visualization. *Journal of Computing Sciences in Colleges*, *33*(5), 115–123.

Guzdial, M., & du Boulay, B. (2019). The History of Computing Education Research. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (1st ed., pp. 11–39). Cambridge University Press.

Hamouda, S., Edwards, S. H., Elmongui, H. G., Ernst, J. V., & Shaffer, C. A. (2019). RecurTutor: An Interactive Tutorial for Learning Recursion. *ACM Transactions on Computing Education*, *19*(1), 1–25.