A Lightweight Industry-Oriented Framework for Teaching Requirements Engineering Using Low-Code Platforms and Flipped Classroom Pedagogy*

Walter Koch[1,*,†], Amit Kumar Mishra[2,†] Dietmar Rafolt[3,†] and Odo Benda[1,†]

[1] AIT Angewandte Informationstechnik ForschungsGmbH, Klosterwiesgasse 32, 8010 Graz, Austria
[2] University West, Sweden; AGH University, Poland
[3] Medical University Vienna, Austria

Abstract

Recent advances in microcomputing and Artificial Intelligence enable the de-ployment of low-cost, low-energy on-premise infrastructures based on Single Board Computers (SBCs). Such environments can host Low-Code Application Platforms (LCAPs) that support collaboration between domain experts and IT professionals while remaining independent of external cloud resources. In these settings, Requirements Engineering (RE) plays a central role, as requirements ar-tefacts are often directly translated into executable system behaviour. This paper presents a lightweight, standards-informed framework for teaching Requirements Engineering in an industry-oriented context. The framework revis-its established RE practices, including structured User Requirements as defined in the ESA Software Engineering Standard, and integrates them with contemporary low-code development practices. Teaching is organised using a flipped classroom methodology combined with project-based learning. Students work in small teams to elicit, structure, validate, and refine requirements for information sys-tems deployed on SBC-based infrastructures. The paper outlines the framework, its pedagogical rationale, and its application in a university course aligned with industrial practice. Initial experience suggests that the approach supports a practice-oriented understanding of Requirements Engineering in low-code environments.

Keywords

Flipped Classroom, Software Development Life Cycle, User Requirements, Single Board Computer, Integrated Project Support Environment, Alvey Programme, Low Code Application Platform, Large Language Model

1. Introduction

Low-Code Application Platforms (LCAPs) [1,2] are increasingly adopted in industrial settings to support Rapid Application Development (RAD) [3] of information systems and collaboration between domain experts and IT professionals. A defining characteristic of such platforms is the tight coupling between requirements artefacts and executable configurations. Consequently, deficiencies in requirements specification tend to manifest immediately in system behaviour, increasing the importance of systematic Requirements Engineering (RE) [4–6].

In software engineering curricula, RE is traditionally introduced through lectures, modelling notations, and documentation-focused assignments. While these approaches effectively convey terminology, standards, and specification techniques, prior work in RE and software engineering education reports challenges in helping students experience authentic stakeholder interaction and understand the down-stream impact of requirements decisions on running systems [7]. This separation between requirements artefacts and executable behaviour becomes particularly salient

* Joint Proceedings of REFSQ-2026 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2026. Poznan, Poland, March 23-26, 2026
[1,*] Corresponding author.
[†] These authors contributed equally.
kochw@ait.co.at (W. Koch); akmishra@ieee.org (A.K.Mishra); dietmar.rafolt@meduniwien.ac.at (D. Rafolt); bendao@ait.co.at (O. Benda)
0000-0002-0609-8649 (W. Koch); 0000-0001-6631-1539 (A.K. Mishra); 0000-0001-8098-6583 (D. Rafolt); 0000-0002-9480-7475 (O. Benda)

in low-code contexts, where requirements are operationalised early and modifications propagate rapidly through executable artefacts.

Recent advances in microcomputing and artificial intelligence enable realistic, low-cost, on-premise infrastructures based on Single Board Computers (SBCs). Such infrastructures reflect industrial constraints related to data sovereignty, security, and operational independence, making them suitable environments for teaching industry-relevant RE practices. This paper presents a lightweight educational framework for teaching Requirements Engineering that integrates standards-informed RE concepts with low-code development and flipped classroom pedagogy [8–10]. The contribution is educational and methodological: it illustrates how established RE principles can be operationalised in contemporary learning environments aligned with industrial practice.

2. Background and Motivation

Low-code platforms are increasingly deployed in industry to enable rapid prototyping, process automation, and collaborative development between domain experts and IT staff. Their effectiveness relies heavily on the quality of requirements artefacts, since workflows, business rules, and data models are often implemented directly from requirements. Consequently, poor specification can lead to misalignment between business needs and deployed systems.

Earlier software engineering research explored similar dependencies. Rapid Appli-cation Development (RAD) and fourth-generation languages (4GL) [11] emphasised tight integration between requirements and executable artefacts. Initiatives such as the Alvey Programme [12] envisioned “Information System Factories” where structured requirements drove automated system creation. Although these visions - based on Integrated Project Support Environments (IPSE) [13] - were limited by technology at the time, they provide conceptual guidance for contemporary RE education.

Software engineering standards, such as the ESA Software Engineering Standard [14], provide a structured approach to defining User Requirements. Key components include operational context, user needs and goals, constraints and assumptions, and acceptance criteria. These concepts remain highly relevant for teaching RE in an industrially inspired low-code context. They allow students to experience require-ments as structured, traceable, and actionable artefacts, rather than abstract documents.

3. Challenges in Teaching Industry-Relevant Requirements Engineering

The output of the User Requirements phase plays a pivotal role in low-code application platforms (LCAPs), serving as the foundation for executable components. For instance, an AI-assisted dialogue such as “create an object entry form based on Spectrum 3.2 museum standard in Aubit 4GL” generated a complete, functional data entry procedure for museum objects. Using the Spectrum 3.2 [15] domain ontology as a knowledge base, the Aubit 4GL-IDE [16] transformed structured requirements directly into a working application, illustrating how precise, standardized artefacts can drive rapid and accurate low-code development. Despite the potential for automation, teaching Requirements Engineering (RE) remains challenging due to the abstract nature of requirements artefacts and limited exposure to real-world constraints. Students often understand theoretical concepts such as stakeholder needs, non-functional requirements, and traceability, but struggle to apply them in executable contexts. This gap is especially evident in low-code environments, where poorly defined requirements can quickly lead to rigid or misaligned systems. Bridging this gap requires embedding RE activities within realistic development contexts. Project-based learning allows students to engage in end-to-end projects, observing how requirements decisions shape system behavior and evolve iteratively. The challenges (Table 1 contains the mapping the proposed Framework) are:

Challenge 1: Limited experiential learning in traditional RE education. Traditional lecture-based approaches often focus on terminology and documentation, offering few opportunities for students to observe the effects of requirements decisions on actual systems [7].

Challenge 2: Gap between RE concepts and operational system behaviour in low-code contexts. In low-code platforms, requirements are operationalised early and changes propagate immediately, but students often do not have access to fully executable systems during coursework.

Challenge 3: Industrial realism and infrastructure constraints. Most professional information system development occurs in large cloud-based environments (e.g., SAP, Salesforce), which are not accessible or practical for classroom settings. Students rarely experience system deployment on real hardware.

Challenge 4: Integration of pedagogy and practice. Aligning flipped classroom methodologies with hands-on development and RE principles is non-trivial. Students need structured guidance to connect theoretical knowledge with practical tasks.

Table 1
Mapping Educational Challenges to the Proposed Framework

Challenge	Framework Solution
Limited experiential learning	Low-code platforms allow students to manipulate running systems and see immediate effects of requirements changes.
Gap between RE concepts and system behaviour	Each exercise operationalises standards-informed RE principles on executable systems deployed on SBCs.
Industrial realism and infrastructure constraints	Students use low-cost, energy-efficient on-premise Single Board Computers, simulating real-world constraints and professional-grade development environments.
Integration of pedagogy and practice	Flipped classroom methodology guides students step-by-step, linking RE concepts to hands-on development, with iterative feedback loops.

Blended learning further supports this process by combining online modules for foundational knowledge with in-person workshops and collaborative exercises, enabling students to link theory with practice. Together, these strategies make requirements decisions tangible, preparing students to navigate the complexities of agile and low-code development while fostering industry-ready competencies.

4. Lightweight Framework for Teaching Requirements Engineering

The framework integrates three core components: standards-informed RE concepts (ESA User Requirements), low-code development platforms deployed on SBCs, and flipped classroom pedagogy with project-based learning.

The teaching loop (outlined in Figure 3) contains following main topics:

Context and stakeholder analysis: Students identify stakeholders, operational scenarios, and system boundaries. Emphasises real-world constraints such as security, limited resources, and regulatory requirements.

User requirements elicitation: Students gather and document user needs through interviews, workshops, and scenario descriptions. ESA-inspired artefacts guide the creation of structured requirements.

Structuring and validation: Requirements are organised into functional, non-functional, and acceptance criteria artefacts. Students validate requirements with peers and instructor guidance, mimicking stakeholder feedback loops in industry.

Low-code implementation and feedback: Requirements are implemented in a low-code environment on SBCs. Students observe the impact of their requirements on system behaviour, iteratively refining artefacts.

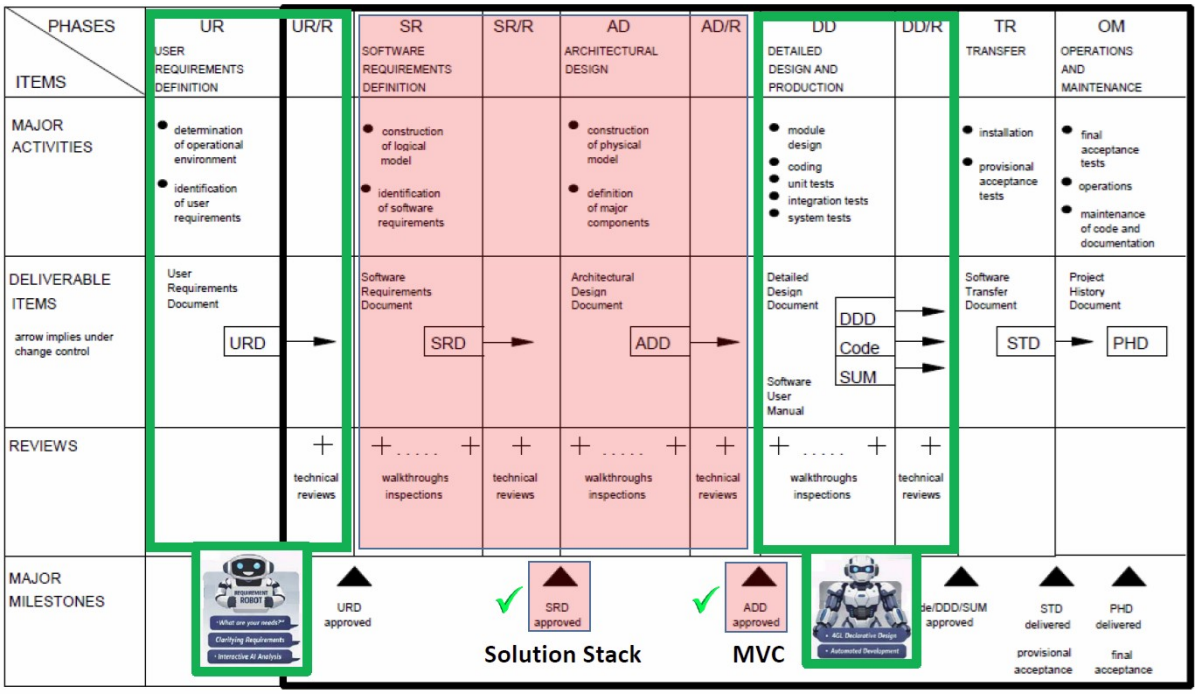

Figure 1: The System Development Life Cycle (SDLC)

Figure 1 presents a segment of an early Software Development Life Cycle (SDLC) model developed by the European Space Agency (ESA) [14], highlighting the phases most relevant to [17] low-code development of Information Systems: User Requirements (UR) and Detailed Design (DD), which includes Low-Code Production. The Software Requirements Definition and Architectural Design phases are not required, as in Information System production based on Single Board Computers, these aspects are already defined by the chosen Solution Stack and the Model-View-Controller (MVC) architectural pattern [18]. In the UR and Production phases, AI 'robots' act as digital collaborators, bringing intelligence and automation to the heart of the Smart "Information System Factory". "These processes leverage an open-source Large Language Model, enhanced with domain-specific knowledge - including ontologies, user requirements standards (e.g., ESA, IREB, the International Requirements Engineering Board), programming documentation, and source code.

Key educational benefits include learning RE in a hands-on, iterative manner aligned with industrial practice, emphasizing the consequences of incomplete or ambiguous requirements, and facilitating understanding of traceability from requirements to implementation.

Students iteratively perform context and stakeholder analysis, requirements elicitation, structuring and validation, and low-code implementation. The framework is supported by SBC infrastructure and flipped classroom tools, linking standards-informed RE artefacts to executable low-code systems. Fig, shows the central teaching loop of the four phases with inputs from ESA User Requirements and supporting elements including Mediathread [19] and SBCs. Output is validated requirements and functioning systems.

Mediathread supports the three phases of the flipped classroom methodology:

Pre-Class Phase: In the Pre-Class phase, students are introduced to theoretical concepts and practical tools through curated multimedia content. Topics include e.g. the ESA User Requirement Concept, how to generate a Domain Ontology, fine-tuning Large Language Models and RAD methodologies. Mediathread is used to host videos, readings, and interactive media. Students annotate content, pose questions, and engage in asynchronous discussions. This phase ensures that students arrive in class with a shared baseline of knowledge.

In-Class Phase: The In-Class phase focuses on hands-on activities and collaborative problem solving. Instructors act as coaches, providing guidance, feedback, and just-in-time explanations. Students work in small teams to design, implement, and test system components. RAD principles are emphasized by encouraging rapid prototyping, frequent testing, and incremental improvements. Short development cycles allow students to quickly observe the effects of design decisions.

Post-Class Phase: In the Post-Class phase, students reflect on their work, document their designs, and refine their implementations. Deliverables may include short reports, code repositories, system diagrams, and demonstration videos. Peer feedback and self-assessment are used to reinforce learning outcomes.

Figure 2 shows the one Flipped Classroom Cycle in BPMN-Notation [20],

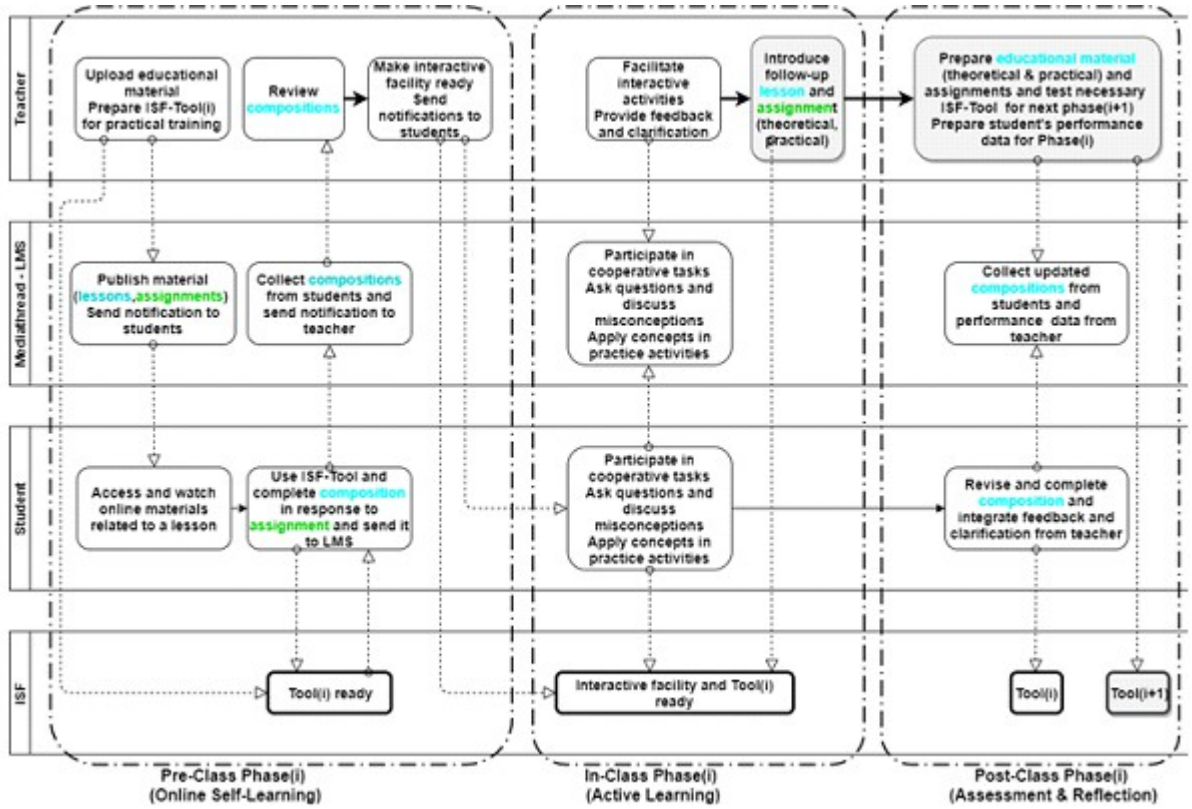

Figure 2: Flipped Classroom Cycle (BPMN)

The participants (roles) in the Flipped Classroom teaching methodology are: Teacher, Student, and a Learning Management System (LMS) [21]. In Figure 2 three "Lanes" (BPMN-Notation) are assigned to these roles. There are also three Phases (grouped by dotted-dashed bold line) in the Workflow: Pre-Class Phase (Online Self-Learning), In-Class Phase (Active Learning), and Post-Class Phase (Assessment and Reflection). The three Phases constitute a Cycle.

5. Application in an Industry-Oriented Course

The framework is designed for courses with small student groups (4–5 members) working on information systems and data management applications across various domains. Projects are

constrained by on-premises deployment and realistic operational requirements. The Flipped Classroom methodology using Mediathread was previously tested in a university course about Knowledge Management [22].

Instructor observations and student feedback indicate increased engagement and an improved understanding of the used tools. In particular, students reported that the course structure encouraged independent preparation, critical reflection, and active discussion of different Knowledge Management topics. While no controlled empirical evaluation has been conducted, initial experience suggests the flipped classroom approach and project based framework is feasible and aligns well with industry-oriented education.

Students emphasized that learning materials were made available in advance and could be accessed flexibly, allowing them to acquire theoretical knowledge independently before applying it in practical assignments. Classroom time was then used for presentations, peer discussion, and instructor feedback.

According to student reflections, this setting fostered cooperative learning, transparency of individual reasoning processes, and structured documentation of evolving understanding. The continuous feedback during assignment work and presentations helped students recognize inconsistencies, unclear assumptions, and missing information in their own and others' contribution - an experience closely related to real-world requirements engineering practice.

Overall, the students perceived the course format as supportive of knowledge ex-change, reflection, and skill development, reinforcing the suitability of the framework for education in domain-specific contexts. While no controlled empirical evaluation has been conducted, initial experience suggests the framework is feasible and aligns well with industry-oriented RE education.

6. Discussion and Conclusion

The framework bridges theoretical RE education and industrial practice. By embedding RE artefacts in an executable context, students experience how decisions affect system behavior - a key aspect of industry readiness. Limitations include reliance on small group sizes and active instructor facilitation, which may restrict scalability. Hardware setup (SBCs) requires maintenance, which may constrain large cohorts.

The framework is transferable to different LCAPs or institutional settings. ESA-inspired artefacts are flexible and can be scaled to larger industrial projects. Iterative, project-based learning with real deployment helps students internalise RE principles more effectively than lecture-only approaches.

In conclusion, this paper presented a lightweight, standards-informed framework for teaching Requirements Engineering in an industry-oriented low-code context. By integrating ESA User Requirements, flipped classroom pedagogy, and hands-on de-ployment, the framework supports practice-oriented RE education. Future work will focus on systematic evaluation, scaling to larger cohorts, and exploring AI-supported requirements analysis.

Acknowledgements

The authors took into the account the comments and suggestions of the reviewers as much as possible. The Introduction Section has to some part reworked and a new bibliographic citation has been added. The challenges in Section 3 have been explicitly listed and later on mapped how those challenges are adressed by the proposed framework. In Section 4 a Business Process Diagramm has been provided explaining the teaching activities and linking the phases clearly to them. The Flipped Classroom approach was used for teaching another topic and the students did a report which is summarized and described in Section 5. It demonstrates that this framework will be particularly valuable for small and medium-sized enterprises in educating small groups of students within the STEM domain.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT-4 in order to: Grammar and spelling check. After using this tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] P. Mellor, "Low-Code Platforms for Rapid Development: Industrial Adoption and Chal-lenges," Journal of Systems and Software, vol. 180, 2021.

[2] AIT Forschungsgesellschaft mbH. AIT-Low Code Application Platform [PDF]. AIT. 2025. https://ait.co.at/wp/wp-content/uploads/2025/11/aubit_02.pdf

[3] K. Beck et al., Rapid Application Development and Iterative Software Engineering, Addi-son-Wesley, 1998.

[4] A. Sutcliffe, Requirements Engineering: Fundamentals, Principles, and Techniques, Spring-er, 2012.

[5] M. Jackson, Software Requirements & Specifications: A Lexicon of Practice, Princi-ples, and Prejudices, ACM Press, 1995.

[6] J. W. Satzinger, R. B. Jackson, and S. D. Burd, Systems Analysis and Design in a Chang-ing World, 7th ed., Cengage Learning, 2016.

[7] G. Regev, D. C. Gause, and A. Wegmann, Experiential learning approach for requirements engineering education. Requirements Engineering, 14(3), 189–203. 2009.

[8] M. Lage, G. Platt, and M. Treglia, Inverting the Classroom: A Gateway to Creating an In-clusive Learning Environment, Journal of Economic Education, vol. 31, no. 1, pp. 30–43, 2000.

[9] J. L. Bishop and M. A. Verleger, The flipped classroom: A survey of the research. ASEE National Conference Proceedings (2013).

[10] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, and M. P. Wenderoth, Active learning increases student performance in science, engineering, and mathematiccs. PNAS, 111(23), 8410–8415 (2014).

[11] A. Alzahrani, 4GL Code Generation: A Systematic Review, International Journal of Ad-vanced Computer Science and Applications, vol. 6, 2020.

[12] D. Talbot, R. W. Witty. Alvey Programme: Software Engineering: Strategy. Alvey Direc-torate, London (1983).

[13] A. W. Brown, A Critical Review of the Current State of IPSE Technology (CMU/SEI-91-TR-29, ESD-TR-91-29). Software Engineering Institute, Carnegie Mellon Universi-ty (1991).

[14] P. H. Feiler, B. E. Randell, and T. Kelly, The ESA Software Engineering Standard PSS-05-0: User Requirements Specification, European Space Agency, 2005. [Online]. Available: http://microelectronics.esa.int/vhdl/pss/PSS-05-0.pdf

[15] Collections Trust. SPECTRUM: The UK museum collections management standard (4th ed.). [Standard detailing 21 procedures]. Collections Trust 2011. https://collectionstrust.org.uk/resource/spectrum/

[16] Aubit Computing Ltd. Aubit 4GL manual 2025. https://aubit4gl.sourceforge.net/aubit4gldoc/manual/html/aubit_4gl_manual__home_page.html

[17] N. B. Ruparelia, Software development lifecycle models. In SIGSOFT Software En-gineering Notes, vol. 35, no. 3 (May 2010), pp. 8–13 (2010).

[18] W. Koch, R. Ortiz, O. Benda, and A. K. Mishra, The information system factory – Return to simplicity. In Proceedings of the 5th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME 2025) 2025.

[19] C. Phillipson, MediaThread: A collaborative multimedia analysis environment. Co-lumbia Center for New Media Teaching and Learning, https://www.cni.org/wp-content/uploads/2011/08/cni_next_phillipson.pdf (2011).

[20] Object Management Group, Business Process Model and Notation (BPMN 2.0.2), https://www.omg.org/spec/BPMN/2.0.2, last accessed 2025/11/12
[21] M. Ebner, S. Schön, J. Alcober, R. Bertonasco, A. Herczak-Ciara, C. Hoppe, F. M. da Silva, Usage of (federated) Learning Management Systems in European University Alliances. Ubiquity Proceedings, 4(1), Article 37. 2024
[22] EuroMACHS Consortium. Europe, Digital Media, Arts and Cultural Heritage Studies (EuroMACHS): programme overview and best practice (Erasmus Programme). 2010.