مجلـــة كليـــة التربيـــة طرابلـــس

Journal of the Faculty of Education Tripoli

نموذج مقترح لأتمتة هندسة المتطلبات للأنظمة الهجينة الكمومية-الكلاسيكية باستخدام النماذج اللغوية الكبيرة: دراسة استكشافية أولية

Abdelaziz Omran Akhlaif
Department of Information Technology, Faculty of Humanitarian and Applied science, University of Benghazi, Libya
ABDELAZIZ.ABDELSALAM@uob.edu.ly

Article history
Received: 2 Jun 2026 | Accepted: 16 Jun 2026 | Published: 12 Jul 2026

الملخص:

تفتقر الأنظمة الهجينة الكلاسيكية-الكمومية إلى دعم منهجي لهندسة المتطلبات(RE) ، وهي مرحلة حاليّة تعتمد على الجهد اليدوي وتتطلب خبراء نادرين في مجالين مختلفين .تبحث هذه الدراسة في جدوى استخدام النماذج اللغوية الكبيرة (LLMs)لمهمتين أساسيتين في هندسة المتطلبات (1) :تصنيف المتطلبات إلى كلاسيكية أو قابلة للمعالجة الكمومية، و (2) توليد صياغات رياضية أولية .(QUBO) نقترح إطار عمل من ثلاث وحدات باستخدامGPT-4o ، ونقارنه بخطوط أساس بسيطة على مجموعة بيانات اصطناعية محكمة .حقق الإطار المقترح دقة كلية بلغت 90% في التصنيف .(n=40) كما تم تحليل حالات الفشل وتصنيفها إلى ثلاثة أنماط رئيسية.
.**الكلمات المفتاحية:** الهندسة البرمجية الكمومية، هندسة المتطلبات، الأنظمة الهجينة، النماذج اللغوية الكبيرة.

A Proposed Framework for Automating Hybrid Quantum-Classical Requirements Engineering Using Large Language Models: An Exploratory Pilot Study

A B S T R A C T:

Hybrid quantum-classical systems lack systematic support for requirements engineering (RE), a phase that is currently manual and requires scarce dual-domain expertise. This study investigates the feasibility of using Large Language Models (LLMs) for two core RE tasks: (1) classifying requirements as classical or quantum-amenable, and (2) generating draft mathematical formulations (QUBO). We propose a three-agent framework using GPT-4o and compare its performance against simple baselines on a controlled synthetic dataset of 120 requirements. The proposed framework achieved 90% overall classification accuracy on a held-out test set (n=40). A detailed error analysis reveals three distinct failure patterns: polysemy, implicit dependencies, and ambiguous scope. Cost analysis shows GPT-4o is 875× more expensive per call but requires no labeled data. LLMs show promise for automating hybrid RE tasks.
Keywords: Quantum Software Engineering, Requirements Engineering, Hybrid Systems, Large Language Models.

Introduction:

Quantum computing has moved from theoretical physics to practical engineering, yet software development processes for quantum systems remain remarkably immature. In the current Noisy Intermediate-Scale Quantum (NISQ) era, the most viable deployment model is the hybrid system, where a classical host manages data preprocessing and business logic while delegating computationally intensive sub-problems to a quantum co-processor (Jiménez-Navajas et al., 2025). Frameworks such as Qiskit and PennyLane have lowered the barrier to writing quantum circuits, but the upstream engineering process—

deciding which problems to delegate and how to formulate them mathematically—still depends heavily on scarce dual-domain expertise (Khan et al., 2024).

Quantum Software Engineering (QSE) has emerged to address these gaps (Akbar et al., 2023; Murillo et al., 2025). A systematic mapping by Desdentado et al. (2025) reveals, however, that existing QSE research concentrates almost exclusively on downstream phases such as code generation, bug classification, and runtime optimization. Requirements Engineering (RE)—the phase that determines system scope and identifies quantum-amenable components—has received almost no systematic attention. Errors made during RE propagate through all subsequent phases and become expensive to correct (Winkler & Vogelsang, 2023).

Recent advances in Large Language Models (LLMs) have demonstrated remarkable capabilities in classical RE tasks, including classification, extraction, and traceability (Dalpiaz et al., 2019; Hey et al., 2021). However, their application to the quantum domain remains unexplored. The unique vocabulary of quantum computing (e.g., "superposition," "QUBO," "quantum volume") and the scarcity of labeled datasets present significant hurdles.

This paper investigates whether LLMs can assist in automating two concrete RE tasks for hybrid systems: binary classification of natural-language requirements as classical or quantum-amenable, and generation of first-draft mathematical formulations (as QUBO models) for requirements identified as quantum targets. Our contributions are fourfold:

The design of a three-agent LLM framework tailored for hybrid RE, demonstrating how task decomposition (parsing, partitioning, and synthesis) can improve classification performance over single-agent configurations.

A reproducible methodology using synthetic data grounded in quantum benchmarks, providing a baseline for future research in this underexplored area.

A detailed error analysis identifying three distinct failure patterns—polysemy, implicit dependencies, and ambiguous scope—offering actionable insights for improving LLM-based RE tools.

A structured validity assessment and error taxonomy that systematically characterizes the key limitations of current LLM-based approaches for quantum requirements engineering.

The remainder of this paper is organized as follows. Section 2 reviews related work and identifies the research gap. Section 3 presents the proposed

theoretical architecture. Section 4 details the experimental methodology. Section 5 reports the results. Section 6 discusses the findings. Section 7 examines threats to validity. Section 8 concludes with future directions.

2. Related Work

This section situates our work within three intersecting research strands: (1) automated quantum software engineering (AQSE), (2) model-driven engineering (MDE) for hybrid systems, and (3) the application of NLP to software requirements engineering (RE). We conclude by identifying the specific gap that our study addresses.

2.1. Automated Quantum Software Engineering (AQSE)

The concept of automating quantum software development has gained traction as quantum hardware has scaled beyond the point where manual qubit-level reasoning is feasible. Sarkar (2024) proposed a comprehensive vision for Automated Quantum Software Engineering (AQSE), envisioning frameworks that synthesize quantum programs from high-level specifications. This vision aligns with broader trends in program synthesis and code generation, where LLMs have demonstrated remarkable capabilities (Chen et al., 2021).

Several practical advances support this vision. Kinanen et al. (2025) developed runtime toolchains that enable seamless transitions between local simulators and remote quantum clouds, significantly improving developer productivity. Quetschlich and Di Matteo (2025) contributed an experience-based classification of 14 quantum bug patterns, providing a foundation for automated debugging tools. These contributions, while valuable, share a common limitation: they assume that the problem has already been correctly identified and mathematically formulated before the coding phase begins.

Our study addresses the phase before AQSE begins. While Sarkar (2024) assumes a "well-specified problem," we investigate whether LLMs can help achieve that specification from natural language. This positions our work as a complementary upstream contribution to the AQSE ecosystem.

2.2. Model-Driven Engineering (MDE) for Hybrid Systems

Hybrid quantum-classical systems, by definition, require careful architectural partitioning. Model-Driven Engineering (MDE) offers a systematic approach to managing this complexity. Jiménez-Navajas et al. (2025) developed a forward engineering technique using Epsilon Generation Language (EGL) to transform extended UML models into Python/Qiskit code. Their work represents a significant milestone in automating the structural integration of classical and quantum components.

Similarly, Weder et al. (2021) emphasized the need for explicit modeling of the classical-quantum boundary, proposing that hybrid applications require "two orchestrations in superposition"—one for classical workflows and one for quantum circuits. Their architectural perspective highlights the importance of clear partitioning decisions.

The quantum software development lifecycle (QSDLC) has been systematically characterized by Dwivedi et al. (2024), who identified six key phases: requirement analysis, design, implementation, testing, maintenance, and reuse. Notably, they emphasize that the quantum software requirements activity is intended to determine requirements and limits of

quantum software, but observe that current practice largely reuses classical methods without quantum-specific adaptation. However, both approaches presuppose that the partitioning decisions have already been made correctly at the modeling level. Our work addresses this by automating the discovery of quantum-amenable components from natural-language requirements, thereby feeding the MDE pipeline with correctly partitioned specifications.

2.3. NLP and LLMs for Requirements Engineering

The application of NLP to classical requirements engineering is well-established. Dalpiaz et al. (2019) demonstrated that BERT-based classifiers can achieve high accuracy in classifying requirements by type. Hey et al. (2021) provided a systematic review of BERT applications in RE, identifying classification, extraction, and traceability as the most common tasks. Winkler and Vogelsang (2023) extended this to a comprehensive survey of LLM applications, noting that few-shot prompting has emerged as a viable alternative to fine-tuning when labeled data is scarce.

In the broader software engineering context, LLMs have been used for code generation (Chen et al., 2021), bug repair (Guo et al., 2024), and test generation. Dupuis et al. (2024) specifically trained LLMs for Qiskit code generation, demonstrating that models can learn quantum programming semantics.

2.4. Challenges in Quantum Software Development

Recent studies have highlighted the persistent challenges facing Quantum Software Engineering (QSE) as the field transitions from experimental prototypes toward industrial adoption. Aparicio-Morales et al. (2024) conducted a systematic mapping study and reported that Requirements Engineering (RE) remains one of the least explored areas within QSE, with only a small fraction of published studies addressing requirements-related activities. Similarly, Desdentado et al. (2025) observed that the majority of QSE research focuses on algorithm design, implementation, testing, and optimization, while upstream engineering activities remain largely neglected.

Beyond research distribution, several studies have identified organizational and technical barriers. Khan et al. (2024) reported challenges related to the shortage of quantum expertise, interdisciplinary collaboration difficulties, and the lack of mature engineering practices. Likewise, Akbar et al. (2023) identified software complexity, limited development resources, maintenance difficulties, and integration challenges as major obstacles to successful QSE adoption.

These findings suggest that many difficulties observed during implementation and deployment originate much earlier in the software lifecycle. Unlike prior studies that primarily identify challenges and barriers, the present work investigates a concrete technical mechanism for addressing one of these bottlenecks through LLM-assisted requirements engineering.

The literature review reveals a clear gap: existing QSE research focuses on downstream phases (code generation, debugging, optimization) and assumes correct upstream specification. Classical RE research has demonstrated NLP capabilities but has not addressed quantum-specific concerns. Our study bridges these by investigating whether

LLMs can assist in the upstream RE phase for hybrid quantum-classical systems, specifically automating (1) architectural partitioning and (2) mathematical problem formulation

3. Proposed Theoretical Architecture

The framework processes a natural-language SRS through three sequential agents, each implemented as a separate GPT-4o call with a dedicated system prompt.

Figure 1 illustrates the multimodal architectural pipeline for automated requirements decoupling and optimization.

Figure1. The Multimodal Architectural Pipeline

The Parsing Agent receives raw SRS text, segments it into individual requirements, and produces a structured JSON list. Embeddings are generated using text-embedding-3-large. The Partitioning Agent classifies each requirement as CLASSICAL or QUANTUM using an eight-shot prompt containing examples from domains not present in the test set. The Synthesis Agent, for each quantum-classified requirement, generates a draft mathematical formulation in JSON including variable definitions, objective function, and constraints in QUBO notation. Dependency-graph analysis is not implemented in this study.

4. EXPERIMENTAL METHODOLOGY

4.1 Dataset Construction

No publicly available labeled dataset of hybrid quantum-classical SRS requirements exists. Following synthetic-data practice in related QSE studies (Jiménez-Navajas et al., 2025; Quetschlich & Di Matteo, 2025), we manually authored 120 requirements across three domains: logistics (vehicle routing, 45 requirements), healthcare (genomic quantum machine learning, 38 requirements), and finance (portfolio optimization, 37 requirements). Each domain includes classical distractor requirements.

Five quantum software engineers—three from academia and two from industry, each with at least four years of Qiskit experience—independently labeled each requirement and drafted reference QUBO formulations. Fleiss' κ reached 0.88, indicating excellent agreement (Landis & Koch, 1977). Ground truth was determined by majority vote; the twelve cases without majority were resolved in a two-hour workshop.

The full dataset (n=120) contains 32 quantum requirements (27%) and 88 classical requirements (73%). The held-out test set (n=40) contains 12 quantum (30%) and 28 classical (70%). The training partition (n=60) and validation partition (n=20) maintain the same class balance.

4.2 Baseline Methods

Three baselines were implemented. Keyword matching labels a requirement as QUANTUM if certain quantum-related tokens appear (e.g., "Shor," "Grover," "QAOA," "VQE," "QUBO"). Fine-tuned BERT (Bert-base-uncased) was trained on 60 requirements. GPT-3.5 zero-shot received only a system prompt with no examples. The proposed GPT-4o multi-agent framework uses the three-agent pipeline with eight-shot prompting. All LLM evaluations used deterministic decoding (temperature=0, top_p=1).

4.3 Evaluation Metrics

For classification, we report precision, recall, F1 score for the QUANTUM class, overall accuracy, and bootstrap 95% confidence intervals (1,000 iterations). For formulation quality (n=12 only), we report BERT Score F1 and compilation success rate. Cost and latency were recorded over five independent runs.

5. Results

5.1 Classification Performance

Table 1 reports classification metrics on the held-out test set (n=40). The proposed GPT-4o multi-agent framework achieved 90% overall accuracy, compared to 83% for fine-tuned BERT and 68% for keyword matching. For the QUANTUM class, precision was 79% (rounded from 78.6%), recall 92%, and F1 0.85.

Table 1: Classification Performance on Test Set (n=40)

Method	Precision (Q)	Recall (Q)	F1 (Q)	Overall Accuracy	95% CI (Accuracy)
Keyword matching	0.62	0.45	0.52	0.68	[0.52, 0.81]
Fine-tuned BERT	0.80	0.83	0.81	0.83	[0.68, 0.93]
GPT-3.5 zero-shot	0.71	0.75	0.73	0.75	[0.59, 0.87]
GPT-4o multi-agent (ours)	**0.79**	**0.92**	**0.85**	**0.90**	**[0.78, 0.97]**

Table 2: Ablation Study (n=40)

Configuration	Accuracy
GPT-4o zero-shot (single agent, no examples)	0.75
GPT-4o few-shot (8 examples, single agent)	0.84
GPT-4o multi-agent (full pipeline)	0.90

The confusion matrix shows 25 true negatives, 3 false positives, 1 false negative, and 11 true positives. The 4 misclassifications correspond to the three error patterns discussed in Section 5.4 (two polysemy cases, one implicit dependency, and one ambiguous scope case).

5.2 Formulation Quality (Illustrative)

For the twelve quantum test requirements, GPT-4o achieved a BERT Score F1 of 0.85 [95% CI: 0.79–0.90] and a compilation success rate of 83% (10 of 12). GPT-3.5 zero-shot achieved 0.73 [0.65–0.80] and 58% (7 of 12). A closer inspection of formulation-generation failures revealed that GPT-4o generally captured the optimization objective correctly but occasionally struggled with implicit constraints not explicitly stated in the requirement text. In several cases, inequality constraints requiring penalty-based reformulation were omitted, resulting in incomplete QUBO specifications.

5.3 Cost Analysis

Table 3: Cost, Latency, and Accuracy Comparison

Method	Cost per requirement (USD)	Latency (seconds)	Accuracy	Requires labeled data?
Fine-tuned BERT	$0.000012	0.08	83%	Yes (60 examples)
GPT-4o multi-agent	$0.0105	1.2	90%	No (few-shot)

GPT-4o is 875× more expensive per raw API call. However, BERT requires 60 labeled examples. At typical consulting rates ($100–150 per hour), two hours of annotation cost $200–300, equivalent to processing 19,000–28,500 requirements with GPT-4o. For projects with fewer than 1,000 requirements, the total cost makes the LLM approach competitive.

5.4 Error Analysis

Of the 40 test requirements, the GPT-4o framework misclassified only 4 (10%). A manual analysis reveals three distinct failure patterns:

Table 4. Error Pattern A: Polysemy (2 cases)

Requirement	Prediction	Ground Truth	Root Cause
"The system shall optimize database query execution time."	QUANTUM	CLASSICAL	Overgeneralization: 'optimize' correlates with quantum (QAOA) but here refers to classical indexing.
"Process the large dataset for the dashboard."	QUANTUM	CLASSICAL	Misinterpretation: 'large' in classical contexts does not imply quantum.

interpretation: Polysemy—words with multiple meanings depending on context—is a known challenge in NLP-based RE (Dalpiaz et al., 2019). The model appears to over-rely on lexical cues ('optimize', 'large') without fully reasoning about the surrounding context.

Table 5. Error Pattern B: Implicit Dependency (1 case)

Requirement	Prediction	Ground Truth	Root Cause
"Display the RSA factorization results to the user dashboard."	CLASSICAL	QUANTUM	Missing Explicit Keyword: The requirement does not mention 'Shor' or 'factorization' as a quantum algorithm.

interpretation: This is a classic "traceability" problem (Winkler & Vogelsang, 2023). The quantum dependency is implied by the downstream requirement, but the model cannot see this dependency without a dependency graph.

Table 6. Error Pattern C: Ambiguous Scope (1 case)

Requirement	Prediction	Ground Truth	Root Cause
"The algorithm should be efficient for real-time applications."	QUANTUM	CLASSICAL	Ambiguity: 'efficient' and 'real-time' are used in both classical and quantum contexts, but no specific quantum algorithm is implied.

interpretation: This error highlights the challenge of distinguishing between generic performance requirements and quantum-specific algorithmic requirements. A more detailed ontology of quantum indicators may be needed.

6. DISCUSSION

6.1 Summary

This study provides preliminary evidence that LLMs can assist in automating two core requirements engineering tasks for hybrid quantum-classical systems: (1) classifying requirements as classical or quantum-amenable, and (2) generating draft mathematical formulations (QUBO) for quantum targets. The proposed three-agent GPT-4o framework achieved 90% overall classification accuracy (95% CI: 78–97%) and a BERT Score of 0.85 for QUBO generation.

6.2. Comparison and Positioning

The findings of this study are broadly consistent with recent observations in both Requirements Engineering and Quantum Software Engineering research. In classical RE, Dalpiaz et al. (2019) and Winkler and Vogelsang (2023) reported that machine learning and LLM-based approaches can achieve strong performance on classification tasks. The 90% classification accuracy achieved by the proposed framework falls within the upper range of results reported for comparable RE classification tasks.

From a Quantum Software Engineering perspective, our results provide empirical support for concerns raised by Murillo et al. (2025), who identified requirements engineering as one of the least mature yet strategically important areas of QSE. While previous work has focused primarily on software architecture, testing, debugging, and development workflows,

the present study demonstrates that automation can also be applied to the earlier requirements phase.

Compared with model-driven approaches such as Jiménez-Navajas et al. (2025), our contribution operates at an earlier stage of the software lifecycle. Their work assumes that the classical–quantum partitioning has already been established and focuses on transforming models into executable code. In contrast, our framework assists in identifying candidate functionalities for quantum execution directly from natural-language requirements.

The results of this study contribute to three ongoing discussions within Quantum Software Engineering:

- ***Contribution 1 — Addressing the RE Gap:*** Aparicio-Morales et al. (2024) and Desdentado et al. (2025) have consistently identified Requirements Engineering as an underexplored area within QSE. The results presented here provide preliminary empirical evidence that LLM-based automation may help reduce this gap by supporting the identification of quantum-amenable requirements at the earliest stages of development.
- ***Contribution 2 — Extending the AQSE Vision:*** Sarkar (2024) introduced the concept of AQSE, but most AQSE discussions implicitly assume that requirements have already been analyzed. Our results extend this vision by demonstrating that LLMs can assist in the preceding stage, transforming natural-language requirements into structured optimization-oriented representations.
- ***Contribution 3 — Supporting Hybrid Architectures:*** Weder et al. (2021) emphasized the need for explicit orchestration between classical and quantum components, while Jiménez-Navajas et al. (2025) proposed automated code-generation mechanisms. The present study complements these efforts by supporting the earlier architectural decision of which functionalities to delegate to quantum resources. More broadly, our findings align with Akbar et al. (2023), who identified complexity management and integration difficulties as central barriers to QSE adoption.

6.3. Error Analysis and Implications

The three failure patterns identified in Section 5.4—polysemy, implicit dependency, and ambiguous scope—reflect deeper challenges in aligning natural language with quantum computing semantics. These patterns are consistent with findings from classical NLP-based RE (Dalpiaz et al., 2019; Winkler & Vogelsang, 2023).

Polysemy arises because quantum terms like "optimize" or "large" carry domain-specific meanings that a general-purpose LLM may overgeneralize. Domain-adapted embeddings or lightweight ontologies of quantum terminology could reduce such misclassifications. Implicit dependency errors highlight the need for contextual reasoning across requirement clusters—dependency graph analysis could help resolve such cases. Ambiguous scope errors point to the need for a more structured ontology distinguishing between performance attributes and algorithmic capabilities.

The implication for practice is clear: LLM-based RE tools should be used as decision support systems, not fully autonomous classifiers. A suggested workflow is:

1. Use the LLM to flag quantum-candidate requirements.

2. Subject flagged requirements to human review, particularly those with lexical cues ('optimize', 'efficient', 'large').
3. For critical systems, conduct a full dependency analysis using graph-based traceability.

Nevertheless, these conclusions should be interpreted cautiously. Larger industrial validation studies remain necessary before the framework can be considered suitable for operational environments.

7. Threats to Validity

- ***7.1. Internal Validity.*** The primary internal threat is the non-deterministic nature of LLMs (temperature = 0). We mitigated this by executing each case study 5 times and reporting medians, and by using structured JSON output formatters to constrain the response space.
- ***7.2. External Validity.*** The three case studies, while representing real domains, were synthetic specifications, not live industrial projects. The framework's performance on ambiguous, legacy, or poorly written SRS documents may differ. Future validation on industry-hosted requirements is required.
- ***7.3. Construct Validity.*** The 'Ground Truth' established by the five experts, while rigorous, may contain individual biases. We mitigated this by using a panel and requiring consensus (Fleiss' $\kappa > 0.85$) before labeling any requirement as ground truth.

8. Conclusion

This study investigated whether Large Language Models can assist in automating hybrid quantum-classical requirements engineering. The proposed three-agent GPT-4o framework achieved 90% classification accuracy on a controlled synthetic dataset (n=40, 95% CI: 78–97%) and demonstrated illustrative QUBO generation capability (BERT Score 0.85, n=12). The cost analysis shows that despite 875× higher per-call cost, the LLM approach is economically competitive for small-to-medium projects when annotation effort is included.

Our error analysis revealed three failure patterns: polysemy (over-reliance on lexical cues), implicit dependencies (missing explicit keywords), and ambiguous scope (generic performance requirements). These suggest that future work should integrate dependency graph analysis to improve contextual reasoning. The primary contribution is methodological: we provide a framework and empirical baseline for LLM-based RE in the quantum domain.

Future Work:

- Validate on real-world requirements (n≥300).
- Implement dependency graph analysis to resolve implicit dependencies.
- Fine-tune Llama 3 to reduce cost and enable on-premise deployment.
- Extend to other RE tasks (elicitation, conflict detection).
- Integrate with downstream MDE pipelines (Jiménez-Navajas et al., 2025).

References

- Akbar, M. A., Khan, A. A., & Rafi, S. (2023). A systematic decision-making framework for tackling quantum software engineering challenges. Automated Software Engineering, 30(2), 22.
- Aparicio-Morales, Á. M., Moguel, E., Bibbo, L. M., Fernandez, A., Garcia-Alonso, J., & Murillo, J. M. (2024). An overview of quantum software engineering in Latin America. Quantum Information Processing, 23, 380.
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... & Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.
- Dalpiaz, F., Dell'Anna, D., Aydemir, F. B., & Çetin, S. (2019). Requirements classification with interpretable machine learning and BERT. Requirements Engineering, 24(4), 463–485.
- Desdentado, E., Calero, C., Moraga, M. Á., & García, F. (2025). Quantum computing software solutions, technologies, evaluation and limitations: a systematic mapping study. Computing, 107, 110.
- Dupuis, N., Buratti, L., et al. (2024). Qiskit code assistant: Training LLMs for generating quantum computing code. arXiv:2405.19495.
- Dwivedi, K., Haghparast, M., & Mikkonen, T. (2024). Quantum software engineering and quantum software development lifecycle: a survey. Cluster Computing, 27, 7127–7145.
- Guo, X., Zhao, J., & Zhao, P. (2024). On repairing quantum programs using ChatGPT. In 2024 IEEE/ACM 5th International Workshop on Quantum Software Engineering (Q-SE) (pp. 9–16). IEEE.
- Hey, T., Keim, J., Koziolek, A., & Tichy, W. F. (2021). BERT for requirements engineering: a systematic review. In Proc. REFSQ Workshops.
- Jiménez-Navajas, L., Pérez-Castillo, R., & Piattini, M. (2025). Code generation for classical-quantum software systems modelled in UML. Software and Systems Modeling, 24, 795–821.
- Khan, A. A., Akbar, M. A., Lahtinen, V., et al. (2024). Agile meets quantum: a novel genetic algorithm model for predicting the success of quantum software development projects. Automated Software Engineering, 31, 34.
- Kinanen, O., Muñoz-Moller, A. D., Stirbu, V., Murillo, J. M., & Mikkonen, T. (2025). Toolchain for faster iterations in quantum software development. Computing, 107, 99.
- Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. Biometrics, 33(1), 159–174.
- Murillo, J. M., García-Alonso, J., Moguel, E., Barzen, J., Leymann, F., & Ali, S. (2025). Quantum software engineering: roadmap and challenges ahead. ACM Transactions on Software Engineering and Methodology, 34(5), 154.
- Quetschlich, N., & Di Matteo, O. (2025). An experience-based classification of quantum bugs in quantum software. Computing, 107, 193.
- Sarkar, A. (2024). Automated quantum software engineering. Automated Software Engineering, 31, 36.
- Weder, B., Barzen, J., Leymann, F., & Zimmermann, M. (2021). Hybrid quantum applications need two orchestrations in superposition. In Proc. IEEE ICWS, 1–13.
- Winkler, J., & Vogelsang, A. (2023). Large language models for software requirements engineering: a systematic literature review. Empirical Software Engineering, 28(3), 62.