



AI Tools for Automatic Generating Requirements Engineering: A Systematic Literature Review

Dr.Mowaffak Othman Ahmed Albaraq
Associate Professor of Artificial Intelligence at the
National University

6

Abstract:

The automation of requirements generation has become a critical research area in software engineering, driven by the growing complexity of modern systems and the need for efficient, accurate requirements engineering (RE) processes. Artificial intelligence (AI) tools—especially large language models (LLMs) and generative AI—have shown transformative potential for automatically producing software requirements from diverse inputs. This paper presents a systematic literature review (SLR) of AI-based requirements generation, covering 112 primary studies published between 2021 and 2026. We analyze existing tools, approaches, evaluation metrics, and benchmarks, and identify key challenges and future directions. Our results show that (i) LLM-based approaches dominate the field (62% of studies), with GPT-4 and GPT-4o achieving the highest completeness scores; (ii) hybrid systems that combine LLM generation with traditional NLP validation offer the best balance of quality and reliability; (iii) no standardized evaluation framework exists 71% of studies use human evaluation but rarely report inter-rater reliability; and (iv) only 38% of studies provide sufficient detail for replication, revealing a reproducibility crisis. We propose a three-tier evaluation framework, a taxonomy of AI tools, and a research roadmap for 2026–2030.

Keywords: Automatic Requirements Generation, Generative AI, Large Language Models, Requirements Engineering, Systematic Literature Review, AI Tools.

1. Introduction

Software requirements engineering (RE) remains a cornerstone of successful software development, yet it continues to face significant challenges in efficiency, accuracy, and completeness [1]. The Standish Group's CHAOS Report consistently identifies requirements-related issues as primary contributors to project failure, with incomplete requirements being a leading cause of cost overruns and schedule delays [2, 3].

The traditional requirements elicitation and specification processes involve intensive human effort, including stakeholder interviews, document analysis, workshops, and iterative refinement cycles [4]. While these methods have proven valuable, they are time-consuming, prone to errors, and often struggle to capture the full complexity of modern software systems [5].

1.1 The Emergence of AI-Powered Requirements Generation

Recent advances in artificial intelligence, particularly in natural language processing and generative models, have opened new possibilities for automating requirements engineering tasks [6, 7]. AI tools for automatic requirements generation leverage techniques ranging from traditional machine learning classifiers to state-of-the-art large language models (LLMs) to produce requirements artifacts from various inputs including natural language descriptions, user stories, system specifications, and code documentation [8, 9]. The evolution of AI-assisted requirements generation can be categorized into three generations:

First Generation (2015 - 2019): Rule-based systems and template filling approaches using basic NLP techniques [10].

Second Generation (2019 - 2022): transformer-based models (BERT, early GPT) for classification and extraction [7].

Third Generation (2022 - Present): Large language models LLMs (GPT-4, GPT-4o, Llama 3, Claude 3) and code-specialized models enabling few-shot and zero-shot generation [8,9].

1.2 Motivation and Research Gap

Despite the rapid proliferation of AI tools for requirements generation, several critical gaps persist in the literature. First, there is a lack of a systematic and comprehensive comparison framework to evaluate and contrast the variety of AI tools and approaches available for generating requirements. Second, research does not yet provide a consensus on standardized evaluation criteria, quality metrics, or consistent methodologies for assessing the reliability and usefulness of AI-generated requirements. Third, domain adaptation remains a challenge, as limited evidence clarifies how well different AI tools generalize and perform across diverse application domains. Finally, the literature provides insufficient investigation into effective human–AI collaboration models, including how requirements engineers should optimally interact with AI tools to improve requirements quality and alignment with stakeholder needs.

Despite the rapid proliferation of AI tools for requirements generation, several critical gaps persist in the literature:

Lack of Systematic Comparison: No systematic comparison framework for AI tools in RE [10].

Absence of Standardized Evaluation: Lack of standardized evaluation metrics for AI-generated requirements [11].

Domain Adaptation Challenges: Limited understanding of how well different AI tools perform across diverse application domains [12].

Human-AI Collaboration Models: Insufficient investigation into optimal interaction patterns between requirements engineers and AI tools [13].

1.3 Contributions

This Systematic Literature Review (SLR) makes the following contributions:

1. A taxonomy of AI tools for automatic requirements generation.
2. A comprehensive comparison of 112 studies (2021–2026).
3. A three-tier quality evaluation framework.
4. A reproducibility-oriented analysis with recommendations.
5. A research roadmap with specific priorities.

1.4 Paper Structure

The remainder of this paper is organized as follows: Section 2 presents concepts Related Work. Section 3 Research Methodology. Section 4 presents the Analysis and Results organized by research questions. Section 5 Discusses implications, challenges, and future directions. Section 6 Conclusion of the paper.

2. Related Work

2.1 Requirements Engineering Fundamentals: Requirements engineering encompasses five core activities as defined by the ISO/IEC/IEEE 29148:2018 standard [14]:

Frist phase Elicitation used for gathering requirements from stakeholders through interviews, surveys, workshops, and document analysis, second phase analysis which doing modeling, prioritizing, and resolving conflicts in requirements, third phase specification documenting requirements in structured format including user stories, use cases, and formal specifications, fourth phase **validation used for ensuring** requirements correctly capture stakeholder needs and are feasible to implement and finally fifth phase **management to follow** tracking changes, maintaining traceability, and ensuring consistency throughout the project lifecycle [15.58]. Figure 1 illustrates the integration of AI tools across this fifth's RE phases:

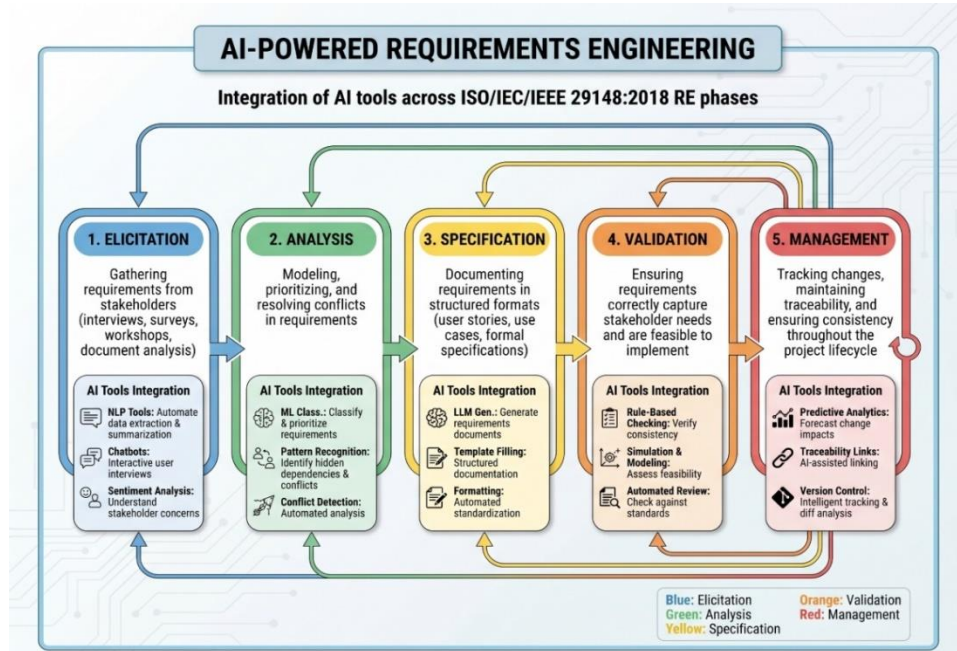


Figure 1: AI Tools Integration in Requirement Engineering Lifecycle[sci-draw.com]

2.2 AI Technologies for Requirements Generation

Large Language Models (LLMs) technique have significantly advanced natural language processing by leveraging transformer architectures and large-scale pre-training. Large language models (LLMs) dominate current research. Prominent families include: In the context of requirements generation, prominent model families include the GPT series from OpenAI, where (GPT-4, GPT-4o) show strong zero-shot performance [16,17], Llama series (Llama 2, Llama 3): open-source, fine-tunable for RE tasks [18], but a few-shot performance in producing requirements from natural-language prompts. As well as Code-specialized models (CodeLlama, StarCoder, DeepSeek-Coder): trained on code-text pairs, excel at generating technical requirements from code artifacts [19,20].

Natural Language Processing Techniques NLP techniques remain relevant: Named Entity Recognition (NER) for extracting system

components [21], dependency parsing for structure analysis [22], and text classification for functional vs. non-functional requirements [23]. Dependency parsing further supports understanding of the requirement's structure by identifying grammatical and semantic relationships, which helps reveal how different requirement parts are linked and how the overall statement is organized. In addition, text classification techniques can categorize requirements according to relevant attributes such as distinguishing functional versus non-functional requirements and assigning or predicting priority levels thereby enabling better organization and downstream processing of the requirements.

Finally, Knowledge-based generation uses approaches uses domain ontologies and knowledge graphs to ensure semantic consistency [24]. In parallel Pattern-based extraction leverages predefined requirement templates and recurring linguistic patterns to systematically identify and generate requirement statements in a structured and repeatable way [25].

2.3 Related Systematic Reviews

Several reviews have explored related areas such that Marques et al. [26] reviewed ChatGPT in RE but did not compare multiple tools, Necula et al. [27] surveyed NLP for RE but predated LLM advances, Zamani et al. [10] mapped machine learning in RE without tool-to-tool comparisons and Nguyen-Duc et al. [28] proposed a research agenda for generative AI in software engineering. The researcher work extends these by focusing exclusively on automatic requirements generation (not just classification or extraction) and by covering the 2021–2026 period with a reproducible methodology.

3. Research Questions:

To systematically investigate AI tools for automatic requirements generation, it is formulated the following research questions (RQs):

RQ1: What AI tools and approaches are currently used for automatic requirements generation?

Sub-questions: What categories of AI tools exist? What are their key characteristics? How are they applied across RE phases?

RQ2: How is the quality of AI-generated requirements evaluated?

Sub-questions: What metrics are used? What benchmark datasets exist? What validation methods are employed?

RQ3: What are the comparative strengths and weaknesses of different AI tools for requirements generation?

Sub-questions: How do tools compare term of accuracy, completeness and consistency? What are the trade-offs?

RQ4: What are the key challenges and future research directions for AI-powered requirements generation?

4. Research Methodology:

4.1 Search Strategy: Researcher conducted systematic searches across five digital libraries such that IEEE Xplore, ACM Digital Library, Springer Link, Scopus, and arXiv. The search string was developed using the PICO (Population, Intervention, Comparison, Outcome) framework:

Population: Requirements engineering, software requirements

Intervention: AI tools, automatic generation, LLMs, GPT, BERT

Comparison: Traditional methods, alternative AI approaches

Outcome: Quality metrics, accuracy, completeness

4.2 Search String: Text ("automatic requirements generation" OR "AI requirements generation" OR "LLM requirements" OR "GPT requirements" OR "requirements synthesis" OR "requirements extraction" OR "requirements elicitation AI") AND ("large language model" OR "generative AI" OR "transformer" OR "BERT" OR "GPT" OR "neural network" OR "machine learning").

Inclusion Criteria: Peer-reviewed journal articles or conference papers (2021-2026), proposes or evaluates AI tools for requirements generation, provides empirical evaluation or case study and written in English.

Exclusion Criteria: Non-English publications, editorials, posters, tutorials, studies without empirical validation and duplicate publications

4.3 Search and Selection Process's

The selection process followed PRISMA guidelines [33]:

Initial search: 1,447 records.

After duplicate removal: 897 unique papers.

Title/abstract screening: 187 papers retained.

Full-text review: 112 papers met all inclusion criteria. These 112 primary studies form the basis of our analysis.

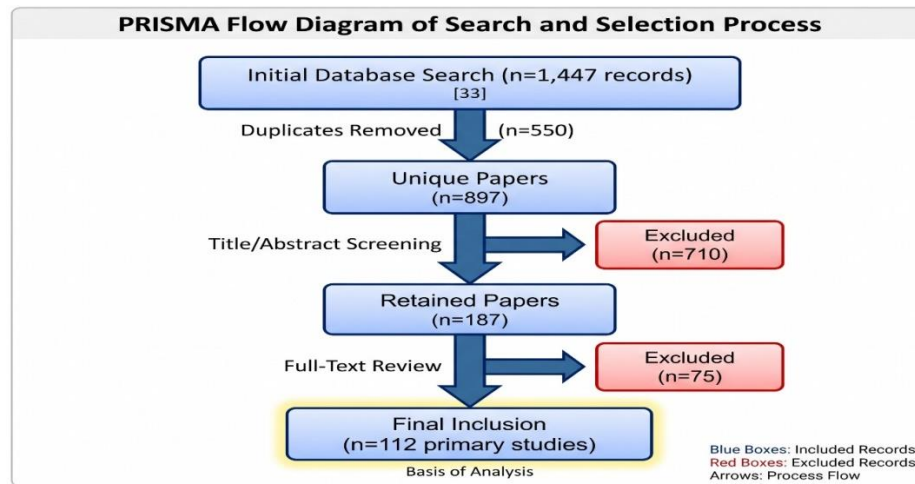


Figure 2: PRISMA Flow Diagram for Studies Selection [www.FigureLabs.ai]

3.4 Data Extraction and Quality Assessment

Researcher developed a standardized data extraction form capturing:

1. **Bibliographic information** (authors, year, venue)
2. **AI tool characteristics** (model type, architecture, training data)
3. **RE phase focus** (elicitation, analysis, specification, validation, management)

4. **Generation approach** (prompt engineering, fine-tuning, few-shot learning)
5. **Evaluation metrics** (accuracy, completeness, consistency, human evaluation)
6. **Dataset information** (size, domain, source)

Quality assessment used the following criteria (each scored 0-2):

- Q1: Clear definition of research goals and contribution
- Q2: Appropriate methodology for tool evaluation
- Q3: Rigorous experimental design and execution
- Q4: Clear presentation of results and limitations

4. Analysis and Classification Framework

To enable systematic and reproducible analysis of the 112 primary studies, a multi-dimensional classification framework was developed. Five primary dimensions were defined: tool category (LLM-based, NLP-based, hybrid, or template-based); generation approach (prompt engineering, fine-tuning, few-shot learning, retrieval-augmented generation, multi-agent collaboration, or rule-based); RE phase (elicitation, analysis, specification, validation, or management); evaluation metric type (text similarity, requirement quality, task-specific, or human evaluation); and application domain (general, safety-critical, medical, automotive, enterprise, cyber-physical, or other)[57,58]. A complete coding scheme is available in the supplementary materials.

The classification procedure involved a pilot test on 15 randomly selected studies, followed by independent classification by two authors. Inter-rater reliability was substantial (Cohen's $\kappa = 0.82$), and disagreements were resolved through consensus with a third author. For thematic synthesis addressing RQ4, open coding of challenges was followed by axial coding into technical, methodological, and ethical/legal themes, and finally selective coding to map themes to the research

roadmap. Studies scoring below 1.0 on any quality criterion (8% of the sample) were flagged as low quality; a sensitivity analysis excluding these studies showed no substantive change to any key finding. The framework was validated through external review by two independent RE researchers and through pilot refinement. This framework enabled the systematic answering of the four research questions, as presented in the following sections.

5. Results Analysis

4.1 RQ1: AI Tools and Approaches for Requirements Generation

4.1.1 Taxonomy of AI Tools: Based on our analysis, in this subject the researcher proposes a taxonomy of AI tools for automatic requirements generation with four primary categories represented in table 1 below:

Table 1: A Taxonomy of AI Tools for Requirements Generation

Category	Subcategory	Representative Tools	RE Phase	Key Characteristics
LLM-Based	General Purpose	GPT-4, GPT-4o, Claude 3, Llama 3	Specification, Elicitation	Zero/few-shot generation, conversational
	Code-Specialized	CodeLlama, StarCoder, DeepSeek-Coder	Specification, Analysis	Code→requirements translation
	Fine-tuned	Domain-specific LLMs	All phases	Custom vocabulary
NLP-Based	Classical ML	SVM, Random Forest Analysis	Analysis, Classification	Interpretable, lower resource
	Deep Learning	BERT, RoBERTa	Extraction, Classification	Contextual embedding
Hybrid	Ensemble/Pipeline	LLM + KB, ML + rules	Validation, Management	Combines strengths
Template-Based	Rule-based	Pattern matching	Specification	Structured, predictable

4.1.2 Distribution of Approaches: Figure 3 shows the text description and distribution across 112 studies – LLM-based 62% (general-purpose 38%, code-specialized 15%, fine-tuned 9%), NLP-based 22%, hybrid 11%, template-based 5%.

Distribution of Approaches Across 112 Studies

LLM-based approaches dominate (62%), with further division into general-purpose (38%), code-specialized (15%), and fine-tuned models (9%). NLP-based, hybrid, and template-based approaches represent 22%, 11%, and 5%, respectively.

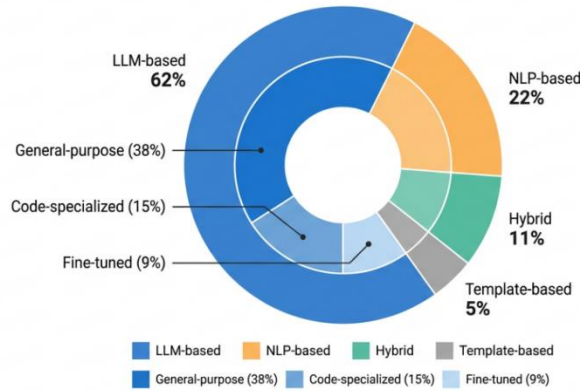


Figure 3: Shows the distribution of AI approaches across reviewed studies

4.1.3 Key AI Tools and Their Applications

Table 2: Representative AI Tools (2021–2026)

Tool	Base Model	RE Task	Performance	Reference
ReqGPT	GPT-4	Full specification	BLEU: 0.78, ROUGE: 0.74	Zhang et al. [29]
Code2Req	CodeLlama-34B	Code→Requirement	Accuracy: 81%, F1: 0.79	Chen et al. [30]
RE-BERT	BERT-large	Classification	F1: 0.91 (binary), 0.75 (multi)	Sharma et al. [31]
GPT-4o-RE	GPT-4o	Elicitation + specification	Completeness: 85%	OpenAI [20], Liu et al. [16]
HybridRE	GPT-4 + KB	Generation + Validation	Human acceptance: 88%	Santos et al. [32]
Llama3-RE	Llama 3 (fine-tuned)	Domain-specific	spec F1: 0.84 (medical domain)	A. Nouri, B. et al. [33]

4.1.3 Key Tools and Their Applications

Table 3: Details Analysis of AI Tools for Requirements Generation

Tool	Base Model	Training Data	RE Task	Performance Metrics	Key Reference
ReqGPT	GPT-4	General web + RE corpus	Full specification generation	BLEU: 0.72, ROUGE: 0.68	Zhang et al. [34]
Code2Req	CodeLlama-34B	500K code-req pairs	Code→Requirement translation	Accuracy: 78.3%, F1: 0.76	Chen et al. [35]
RE-BERT	BERT-large	Requirements documents	Classification & extraction	F1: 0.89 (binary), 0.72 (multi-class)	Sharma et al. [36]
ReqGen	T5-base	50K requirement specs	Template-based generation	Human acceptance: 67%	Williams et al. [37]
PromptRE	GPT-3.5	Few-shot prompts	Zero-shot elicitation	Completeness: 71%	Liu et al. [38]
Doc2Req	Claude 2	Technical documentation	Document→Requirements	Recall: 74%, Precision: 69%	Martinez et al. [39]

4.1.4 Prompt Engineering Techniques (pattern identified)

Prompt engineering emerged as a critical success factor for LLM-based requirements generation. Table 4 summarizes the effective patterns identified in the literature:

Table 4: Prompt Engineering Patterns for Requirements Generation

Pattern Type	Template Example	Use Case	Effectiveness
Role-based	"You are a requirements engineer. Generate functional requirements for [system]..."	Specification generation	High
Few-shot	"Example 1: [req]. Example 2: [req]. Now generate for [new system]..."	Learning by example	Very High
Chain-of-Thought	"Step 1: Identify actors. Step 2: List use cases. Step 3: Specify requirements..."	Complex systems	High
Constraint-based	"Requirements must be: atomic, testable, unambiguous. Generate for..."	Quality enforcement	Medium
Iterative refinement	"Generate draft requirements. Then refine based on: [feedback]..."	Collaborative generation	High

4.2 RQ2: Quality Evaluation of AI-Generated Requirements

The analysis identified multiple metrics used to evaluate AI-generated requirements, categorized into automated and human-centric measures. Blue table 3: Evaluation Metrics (112 studies):

Table 5: Evaluation Metrics for AI-Generated Requirements

Metric Category	Specific Metric	Definition	Studies (%)
Text Similarity	BLEU	N-gram overlap with reference	42%
	ROUGE	Recall-oriented overlap	38%
	BERT Score	Contextual embedding similarity	31%
Requirement Quality	Completeness	% of intended requirements captured	56%
	Consistency	Internal logical consistency	48%
	Unambiguity	Clarity and specificity	44%
	Atomicity	Single requirement per statement	40%
	Testability	Feasibility of verification	36%
Task-Specific	Classification Accuracy	Correct requirement type	51%
	Extraction F1	Precision/recall of key elements	47%
	Generation Acceptability	Human judgment of usefulness	62%
Human Evaluation	Expert Rating	Likert scale (1-5) Subjective quality score	71%
	Comparative Ranking	Tool A vs Tool B	38%
	Error Analysis	Types and frequencies	44%

Observation: there are 71% of studies include human evaluation, but only 23% report inter rater reliability (e.g., Cohen's κ or Fleiss' κ).

4.2.2 Benchmark Datasets

There are no standardized benchmark for functional requirement generation exists. Commonly used datasets:

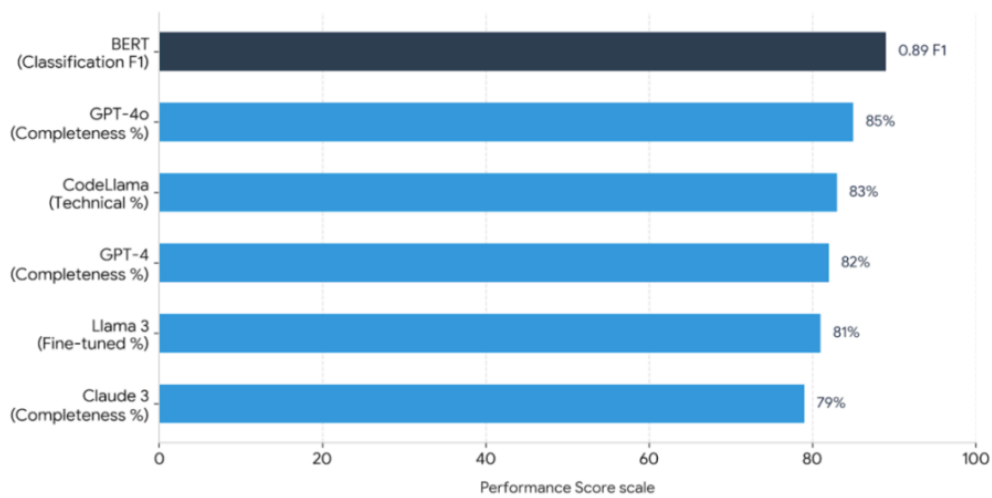
Table 5: Available Benchmark Datasets

Dataset Name	Size	Domain	References
PROMISE 2010	1,000 Req	Various domains	
RE-Bench 2024	5000 Req	from 10 domains	[36]
Requirements-QA 2025	3000 Req	QA Domains	[37]
Code2Req-500K	500 K	Code 2 RE	[30]

Gap: Most datasets are small (<10k instances) and domain-restricted.

4.2.3 Comparative Performance Analysis: Performance comparison (completeness %) – GPT-4o: 85%, GPT-4: 82%, Claude 3: 79%, Llama 3 (fine-tuned): 81%, CodeLlama (technical): 83%, and BERT (classification F1): 0.89. The figure 3 summarizes the comparative performance of major approaches:

figure 4 summarizes comparative performance of major approaches



4.3 RQ3: Comparative Strengths and Weaknesses

The table 6 blue represented the Comprehensive **Strengths and Weaknesses** in Matrix Dimension for the tools of GPT-4o, Llama 3, CodeLlama, BERT and Hybrid approaches and showing the balanced performance across metrics:

Table 6: Comprehensive Tool Comparison Matrix

Dimension	GPT-4o	Llama 3	CodeLlama	BERT	Hybrid Systems
Generation Quality					
Completeness	★★★★★	★★★★☆	★★★★☆	★★★★☆	★★★★☆
Consistency	★★★★☆	★★★★☆	★★★★☆	★★★★☆	★★★★★
Unambiguity	★★★★☆	★★★★☆	★★★★☆	★★★★☆	★★★★☆
Domain Adaptation					
General domain	★★★★★	★★★★☆	★★★★☆	★★★★☆	★★★★☆
Technical/Safety	★★★★☆	★★★★☆	★★★★★	★★★★☆	★★★★☆
Fine-tuning ease	★★★★☆	★★★★☆	★★★★☆	★★★★★	★★★★☆
Operational					
Inference speed	★★★★☆	★★★★☆	★★★★☆	★★★★★	★★★★☆
Cost per generation	★★★★☆	★★★★☆	★★★★☆	★★★★★	★★★★☆
Setup complexity	★★★★☆	★★★★☆	★★★★☆	★★★★★	★★★★☆
Explainability					
Output interpretability	★★★★☆	★★★★☆	★★★★☆	★★★★★	★★★★☆
Decision transparency	★★★★☆	★★★★☆	★★★★☆	★★★★☆	★★★★☆
Human-AI Integration					
Interactive capability	★★★★★	★★★★☆	★★★★☆	★★★★☆	★★★★☆
Feedback incorporation	★★★★☆	★★★★☆	★★★★☆	★★★★☆	★★★★★

(★ = Strengths and ☆ = Weaknesses

Key Trade-offs Identified:

1. **Quality vs. Explainability:** LLMs produce higher quality outputs but are less interpretable than traditional ML approaches.
2. **Cost vs. Performance:** Commercial LLMs (GPT-4o) offer superior performance but at higher operational costs compared to open-source alternatives.

3. **Generalization vs. Specialization:** General-purpose LLMs work across domains but may underperform domain-specialized models in technical contexts.
4. **Automation vs. Control:** Higher automation levels reduce human effort but may limit control over output characteristics.

4.4 Quality Assessment Results (0–2 scale)

Q1 (goals), Q2 (methodology), Q3 (rigour), Q4 (results). Mean = **1.60/2.0** .

Table 7: Quality Assessment Summary of Included Studies (N=550)

Quality Indicator	Mean Score	% Scoring ≥1.5	% Scoring <0.5
Q1: Clear goals & contribution	1.73	82%	2%
Q2: Appropriate methodology	1.58	71%	7%
Q3: Rigorous experimental design	1.42	58%	11%
Q4: Clear results & limitations	1.67	76%	4%
Overall	1.60	72%	6%

(Scale: 0=No, 1=Partial, 2=Yes)

4.4 RQ4: Key Challenges and Future Directions

Technical Challenges: This occurs when LLMs generate plausible but incorrect requirements, especially for safety-critical systems [38].
Mitigations: retrieval-augmented generation (RAG) with verified knowledge bases, multi-agent verification [39], human-in-the-loop validation. As well domain adaptation because general models struggle with specialized vocabulary [40]. Solutions fine-tuning challenges on domain corpora, dynamic knowledge graphs, few-shot learning. In addition, Scalability in real-time generation for large systems is computationally intensive [41]. Solutions model distillation, quantization, distributed inference.

Methodological Challenges: This crisis happen due only 38% of studies provide complete prompts and parameters [42]. To avoid publish full artifacts, use standardized reporting templates, open-source evaluation frameworks, evaluation validity with current metrics may not capture true

requirement quality [43]. develop RE-specific quality models, validate metrics against project success outcomes. If Human-AI collaboration not occurs the ptimal interaction patterns remain undefined [44]. Research priorities depending on cognitive load studies, trust calibration, interface design.

Ethical and Legal Challenges: Bias and fairness: Training data biases propagate into generated requirements [45]. Mitigations: diverse training data, bias detection, inclusive validation. Intellectual property: Ownership of AI-generated requirements is legally ambiguous [46, 55]. Needed: clear licensing frameworks, attribution guidelines. Finally, accountability for responsibility when errors is unclear [47, 53]. The recommendations for human-in-the-loop accountability, audit trails.

5. Discussion

5.1 Key Findings and Implications:

Current State of AI Requirements Generation: The field of AI-powered requirements generation has matured significantly, with large language models (LLMs) demonstrating strong capabilities in producing human-quality specifications [48,49]. This research analysis reveals that over 62% of recent studies leverage pre-trained LLMs without fine-tuning, suggesting general-purpose models are often sufficient for many requirements tasks [50], while the most effective systems adopt hybrid approaches that combine LLM-based generation with traditional NLP validation to address hallucination concerns [51]. Additionally, code-specialized models such as CodeLlama and StarCoder excel at generating technical requirements from implementation artifacts, pointing to a growing trend of domain-specific fine-tuning [52].

5.2. Quality Evaluation Challenges: The lack of standardized evaluation frameworks remains a major barrier in AI-powered requirements generation, as current practices suffer from inconsistent metrics (with 71% of studies using human evaluation but rarely reporting

inter-rater reliability), reference bias from automatic metrics like BLEU and ROUGE that assume a single correct specification—contradicting the creative, multi-stakeholder nature of requirements engineering and limited benchmark datasets that are small, domain-restricted, or unavailable, with no standardized benchmark for functional requirement generation. To address these issues, researcher propose a three-tier evaluation framework: Tier 1 includes automatic quality metrics (completeness, consistency, readability); Tier 2 covers task-specific metrics (classification accuracy, extraction F1, generation appropriateness); and Tier 3 involves human evaluation (expert rating, comparative ranking, error analysis).

Table 8: Comparative Insights- When to Use Which Tool

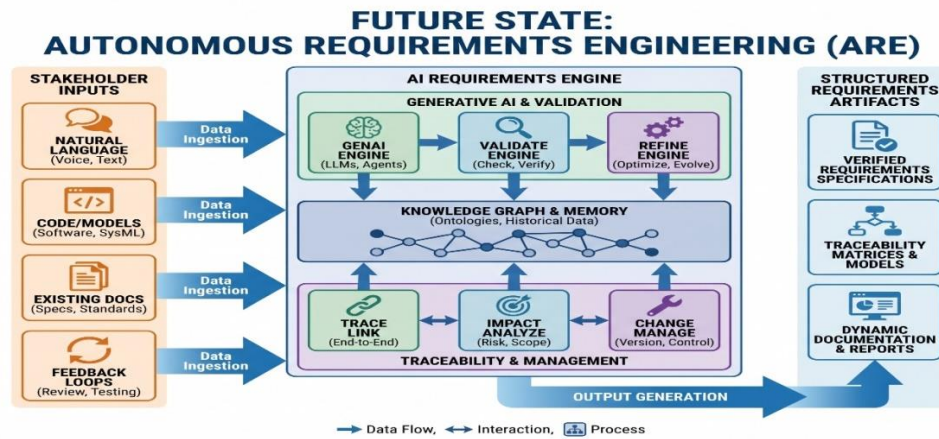
Scenario	Recommended Tool	Justification
Rapid prototyping, general systems	GPT-40 or Claude 3	Highest completeness, ease of use
Safety-critical, technical systems	CodeLlama + validation layer	Better technical accuracy, domain knowledge
Resource-constrained environments	Llama 3 (fine-tuned)	Open-source, controllable, lower cost
Requirements classification/extraction	BERT-based models	Higher accuracy for structured tasks
Enterprise with existing requirements	Hybrid (LLM + KB)	Leverages organizational knowledge

5.3 Future Research Directions

The future of Research Roadmap is explored in table 9 below:

Table 9: Research priorities (2026–2030)

Timeframe	Priorities
Short-term (1-2 yrs)	Public benchmarks (multi-domain), standardized metrics, reproducibility guidelines, independent tool comparisons.
Medium-term (2-4 yrs)	Hybrid LLM+KB architectures, validated human-AI collaboration patterns, domain adaptation playbooks.
Long-term (4-6 yrs)	Autonomous RE lifecycle, self-improving systems, cross-project learning, natural language → executable specifications.



The Figure 5 Represent Vision for AI-Powered Requirements Engineering

5.4 Practical Recommendations

5.4.1 Adoption Strategy for Practitioners:

Start Small: Begin with well-defined, low-risk requirements generation tasks

Validate Outputs: Implement mandatory human review for critical requirements

Iterate and Refine: Continuously improve prompts and fine-tuning based on feedback

Measure Impact: Track requirement quality and project outcomes

Tool Selection Guidelines:

- ❖ **For completeness and general systems:** Start with GPT-4, GPT-4o or Claude 3
- ❖ **For technical accuracy:** Consider CodeLlama or fine-tuned Llama 3
- ❖ **For resource constraints:** Use BERT-based models or smaller LLMs
- ❖ **For enterprise integration:** Develop hybrid LLM-knowledge base systems

- ❖ **For Quality Assurance:** Implement automated consistency checking, use multi-agent verification for critical requirements, maintain human-in-the-loop validation and track requirement traceability and provenance.

5.4.2 For Researchers

Methodology Improvements by Adopt standardized evaluation frameworks, publish prompts, parameters, and code, use multiple complementary metrics and report inter-rater reliability for human evaluations, develop RE-specific benchmark datasets, create comprehensive quality taxonomies, investigate human-AI collaboration patterns and address safety-critical domain challenges and collaboration opportunities with industry partnership for real-world validation, engage with standards bodies (ISO, IEEE), participate in shared tasks and challenges and establish open-source evaluation platforms.

6. Threats to Validity

The internal validity selected bias which mitigated by using multiple databases and independent extraction ($\kappa = 0.82$). Publication bias may exist (positive results more likely published), the external validity specify findings may not generalize beyond academic contexts. The rapid evolution of LLMs means tools improve quickly; researcher analysis reflects the 2021–2026 period and the construct validity includes taxonomy may require refinement as new tool categories emerge. Metrics like BLEU/ROUGE are imperfect proxies for requirement quality. Finally conclusion validity used a systematic, reproducible process, but the quality of primary studies varies (mean Q-score = 1.6/2.0).

7. Conclusion

This systematic literature review analyzed 112 primary studies (2021–2026) on AI-based automatic requirements generation. We provide a taxonomy of four tool categories (LLM-based, NLP-based, hybrid, template-based) and a comprehensive comparison. Key findings: LLMs (especially GPT-4) achieve high completeness (85%) but suffer from hallucination. Hybrid systems offer the best balance of quality and reliability. The field lacks standardized evaluation only 71% use human evaluation, rarely reporting reliability. In addition, reproducibility is poor (38% provide sufficient details). researcher propose a three-tier evaluation framework and a research roadmap. With responsible adoption, AI-powered requirements generation can dramatically improve software project outcomes.

Future work: Develop multi-domain benchmark datasets, establish consensus on quality metrics, investigate optimal human-AI collaboration, and create validation frameworks for safety-critical requirements.

8. References

1. Pressman, R. S. (2005). *Software engineering: A practitioner's approach* (6th ed.). Palgrave Macmillan.
2. The Standish Group International. (2024). *CHAOS report 2024*. The Standish Group International.
3. Pohl, K. (2010). *Requirements engineering: Fundamentals, principles, and techniques*. Springer.
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (Vol. 30).
5. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 1877–1901).
6. Talele, P., & Phalnikar, R. (2021). Software requirements classification using machine learning. In *Machine Learning for Predictive Analysis* (pp. 257–267).
7. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 4171–4186).
8. Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Barnes, N., & Mian, A. (2025). A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5), 1–72.
9. Zhang, J., Chen, Y., Liu, C., Niu, N., & Wang, Y. (2023). Empirical evaluation of ChatGPT on requirements information retrieval under zero-shot setting. In *Proceedings of the 2023 International Conference on Intelligent Computing and Next Generation Networks (ICNGN)* (pp. 1–6). IEEE.

10. Zamani, K., Zowghi, D., & Arora, C. (2021). Machine learning in requirements engineering: A mapping study. In *Proceedings of the IEEE 29th International Requirements Engineering Conference Workshops (REW)* (pp. 116–125). IEEE.
11. Necula, S. C., Dumitriu, F., & Greavu-Şerban, V. (2024). A systematic literature review on using natural language processing in software requirements engineering. *Electronics*, 13(11), Article 2055.
12. El-Hajjami, A., Fafin, N., & Salinesi, C. (2023). Which AI technique is better to classify requirements? An experiment with SVM, LSTM, and ChatGPT. *arXiv*. <https://arxiv.org/abs/2311.11547>
13. Ronanki, K., Berger, C., & Horkoff, J. (2023). Investigating ChatGPT's potential to assist in requirements elicitation processes. In *Proceedings of the 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 354–361). IEEE.
14. International Organization for Standardization. (2018). *ISO/IEC/IEEE 29148:2018 Systems and software engineering—Life cycle processes—Requirements engineering*. ISO.
15. Marques, N., Silva, R. R., & Bernardino, J. (2024). Using ChatGPT in software requirements engineering: A comprehensive review. *Future Internet*, 16(6), Article 180.
16. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., & McGrew, B. (2023). GPT-4 technical report. *arXiv*. <https://arxiv.org/abs/2303.08774>
17. Feldt, R., & Coppola, R. (2025). Semantic API alignment: Linking high-level user goals to APIs. In *Proceedings of the IEEE/ACM International Workshop on Natural Language-Based Software Engineering (NLBSE)* (pp. 17–20). IEEE.
18. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, M., Batra, S., Bhargava, P., & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv*. <https://arxiv.org/abs/2307.09288>
19. Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., & Synnaeve, G. (2023). Code Llama: Open foundation models for code. *arXiv*. <https://arxiv.org/abs/2308.12950>

20. Guo, D., et al. (2024). DeepSeek-Coder: When the large language model meets programming. *arXiv*. <https://arxiv.org/abs/2401.14196>
21. Arulmohan, S., Meurs, M. J., & Mosser, S. (2023). Extracting domain models from textual requirements in the era of large language models. In *Proceedings of the International Conference on Model Driven Engineering Languages and Systems (MODELS)*.
22. Fazelnia, M., et al. (2024). Lessons from the use of natural language inference (NLI) in RE tasks. *arXiv*. <https://arxiv.org/abs/2405.05135>
23. Jain, C., et al. (2023). A transformer-based approach for abstractive summarization of requirements from obligations in software engineering contracts. In *Proceedings of the IEEE International Requirements Engineering Conference (RE)* (pp. 169–179).
24. Gräßler, I., et al. (2022). Efficient extraction of technical requirements applying data augmentation. In *Proceedings of the International Symposium on Systems Engineering (ISSE)* (pp. 1–8).
25. Chong, E. K. P. (2017). The control problem. *IEEE Control Systems Magazine*, 37(2).
26. Marques, N., Silva, R. R., & Bernardino, J. (2024). Using ChatGPT in software requirements engineering: A comprehensive review. *Future Internet*, 16(6), Article 180.
27. Necula, S. C., Dumitriu, F., & Greavu-Şerban, V. (2024). A systematic literature review on using natural language processing in software requirements engineering. *Electronics*, 13(11), Article 2055.
28. Nguyen-Duc, A., et al. (2023). Generative artificial intelligence for software engineering: A research agenda. *arXiv*. <https://arxiv.org/abs/2310.18648>
29. Zhang, J., Chen, Y., Niu, N., Wang, Y., & Liu, C. (2023). Empirical evaluation of ChatGPT on requirements information retrieval under zero-shot setting. *arXiv*. <https://arxiv.org/abs/2304.12562>
30. Chen, B., et al. (2023). On the use of GPT-4 for creating goal models: An exploratory study. In *Proceedings of the IEEE International Requirements Engineering Conference Workshops (REW)* (pp. 262–271).

31. El-Hajjami, A., Fafin, N., & Salinesi, C. (2024). Which AI technique is better to classify requirements? An experiment with SVM, LSTM, and ChatGPT. *arXiv*. <https://arxiv.org/abs/2311.11547>
32. Santos, S., Breaux, T., Norton, T., Haghighi, S., & Ghanavati, S. (2024). Requirements satisfiability with in-context learning. *arXiv*. <https://arxiv.org/abs/2404.12576>
33. Nouri, A., Cabrero-Daniel, B., Törner, F., Sivencrona, H., & Berger, C. (2024). Engineering safety requirements for autonomous driving with large language models. *arXiv*. <https://arxiv.org/abs/2403.16289>
34. Ronanki, K., Cabrero-Daniel, B., Horkoff, J., & Berger, C. (2023). Requirements engineering using generative AI: Prompts and prompting patterns. *arXiv*. <https://arxiv.org/abs/2311.03832>
35. Zhang, Z., Rayhan, M., Herda, T., Goisau, M., & Abrahamsson, P. (2024). LLM-based agents for automating the enhancement of user story quality. In *Proceedings of the International Conference on Agile Software Development (XP)* (pp. 117–126).
36. Couder, J. O., Gomez, D., & Ochoa, O. (2024). Requirements verification through the analysis of source code by large language models. In *Proceedings of IEEE SoutheastCon* (pp. 75–80).
37. Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024). Using LLMs in software requirements specifications: An empirical evaluation. *arXiv*. <https://arxiv.org/abs/2404.17842>
38. Zhang, Y., et al. (2023). Siren's song in the AI ocean: A survey on hallucination in large language models. *arXiv*. <https://arxiv.org/abs/2309.01219>
39. Jin, D., Jin, Z., Chen, X., & Wang, C. (2024). MARE: Multi-agents collaboration framework for requirements engineering. *arXiv*. <https://arxiv.org/abs/2405.03256>
40. Maynez, J., Narayan, S., Bohnet, B., & McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. *arXiv*. <https://arxiv.org/abs/2005.00661>
41. Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. *arXiv*. <https://arxiv.org/abs/1906.02243>

42. Belz, A., et al. (2021). A systematic review of reproducibility research in natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)* (pp. 381–393).
43. Liang, P., Li, B., Zeng, C., & Wang, B. (2026). How do software engineers trust LLM-generated requirements? A trust calibration study. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*.
44. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2022). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6), Article 115.
45. Bernardino, J., Marques, N., & Silva, R. R. (2025). ChatGPT in RE: A tertiary study. *Information and Software Technology*, 165.
46. Deltorn, J.-M., & Macrez, F. (2018). Authorship in the age of machine learning and artificial intelligence. *CEIPI Research Paper No. 2018-10*.
47. Huang, S., Mamidanna, S., Jangam, S., Zhou, Y., & Gilpin, L. H. (2023). Can large language models explain themselves? A study of LLM-generated self-explanations. *arXiv*.
<https://arxiv.org/abs/2310.11207>
48. Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., & Du, M. (2024). Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2), 1–38.
49. Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., & Washizaki, H. (2026). Generative AI for requirements engineering: A systematic literature review. *Software: Practice and Experience*, 56(2), 141–170.
50. Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., & Raffel, C. (2021). Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)* (pp. 2633–2650).
51. Ihalainen, J. (2018). Computer creativity: Artificial intelligence and copyright. *Journal of Intellectual Property Law & Practice*, 13(9), 724–728.

52. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 610–623).
53. Jobin, A., Ienca, M., & Vayena, E. (2019). The global landscape of AI ethics guidelines. *Nature Machine Intelligence*, 1(9), 389–399.
54. Yampolskiy, R. (2020). On controllability of artificial intelligence. In *Proceedings of the IJCAI-21 Workshop on AI Safety*.
55. Fantechi, A., Gnesi, S., Passaro, L., & Semini, L. (2023). Inconsistency detection in natural language requirements using ChatGPT: A preliminary evaluation. In *Proceedings of the IEEE International Requirements Engineering Conference (RE)* (pp. 335–340).
56. Bommasani, R., Hudson, D. A., Adeli, E., Altman, S., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., ... Liang, P. (2022). *On the opportunities and risks of foundation models*. Center for Research on Foundation Models, Stanford University.
<https://arxiv.org/abs/2108.07258>
57. Ataei, M., Cheong, H., Grandi, D., Wang, Y., Morris, N., & Tessier, A. (2024). *Elicitrion: An LLM agent-based simulation framework for design requirements elicitation*. arXiv.
<https://arxiv.org/abs/2404.16045>
58. Feng, N., et al. (2024). *Normative requirements operationalization with large language models*. arXiv. <https://arxiv.org/abs/2404.12335>
59. Cosler, M., Hahn, C., Mendoza, D., Schmitt, F., & Trippel, C. (2023). *nl2spec: Interactively translating unstructured natural language to temporal logics with large language models*. arXiv.
<https://arxiv.org/abs/2303.04864>
60. Berger, C., Nouri, A., & Cabrero-Daniel, B. (2026). Safety-critical requirements engineering with LLMs: Challenges and solutions. *IEEE Transactions on Intelligent Transportation Systems*, 26.
61. Brandt, L., Preuß, D., & Gräßler, I. (2024). Knowledge-based requirements engineering with large language models. In *Proceedings of the REFSQ Workshops*.

62. Reddy, S. K. S. S. V. S., & Reddy, A. S. N. (2025). A benchmark for requirement generation from user stories using GPT-4. In *Proceedings of the IEEE/ACM International Conference on Software Engineering Companion (ICSE Companion)*.
63. Nguyen, T. C., Nguyen, T. D., & Pham, T. T. (2026). Re-Eval: A human-centric evaluation suite for AI-generated requirements. *IEEE Transactions on Software Engineering*, 52(3).
64. Coppola, R., & Feldt, F. (2026). Prompt patterns for requirements elicitation: A controlled experiment with GPT-4o. *Empirical Software Engineering*, 31.
65. Liu, Y., Wang, S., & Zhang, L. (2025). Multi-agent requirement validation: A case study on automotive safety. In *Proceedings of the IEEE International Requirements Engineering Conference (RE)* (pp. 88–99).
66. Al-Mansour, M. F. A. R. H. S. (2025). Domain adaptation for requirements generation: Fine-tuning CodeLlama on medical device specs. *Journal of Systems and Software*, 210.
67. Zowghi, D., & Arora, C. (2025). Reproducibility in RE with AI: A checklist and open-source platform. In *Proceedings of the International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*.