AI-Enhanced Requirements Traceability Using MBSE and Large Language Models for Complex Systems

Henock Legesse (NASA/GSFC-592)
Systems Engineer & MBSE/AI Innovation Lead

Stanley 'Skip' Bicknell (ASRC Federal Analytical Service)
Requirement Manager

AI4SE & SE4AI Research Workshop — 09/18/2025

Outline

1. Introduction
2. Methodology
3. Experimental Setup
4. Results
5. Discussion
6. Conclusion

The Challenge of Requirements Traceability

Current Challenges

- Manual tracing is labor-intensive and error-prone
- Complex systems involve hundreds or thousands of requirements
- Inherited requirements often have incomplete traceability
- SE resource constraints lead to traceability gaps

Practical Impact

- Engineers spend 5-10 minutes per requirement for tracing [1]
- For our dataset: 53-106 hours of engineering time
- Inconsistent results between engineers
- Technical debt accumulates throughout lifecycle

Problem Statement

Requirements traceability is essential for complex systems engineering but often proves labor-intensive and error-prone when performed manually.

Research Objectives

Develop an AI-enhanced approach that:

- Automates the labor-intensive aspects of requirements traceability
- Preserves human oversight and judgment
- Integrates with existing systems engineering workflows
- Improves both efficiency and quality of traceability analysis

Key Research Questions

- How can LLMs be effectively applied to systems engineering problems?
- Can AI-augmented approaches achieve accuracy comparable to human experts?
- How to balance automation with necessary human oversight?

Methodology: Multi-Layered AI Approach

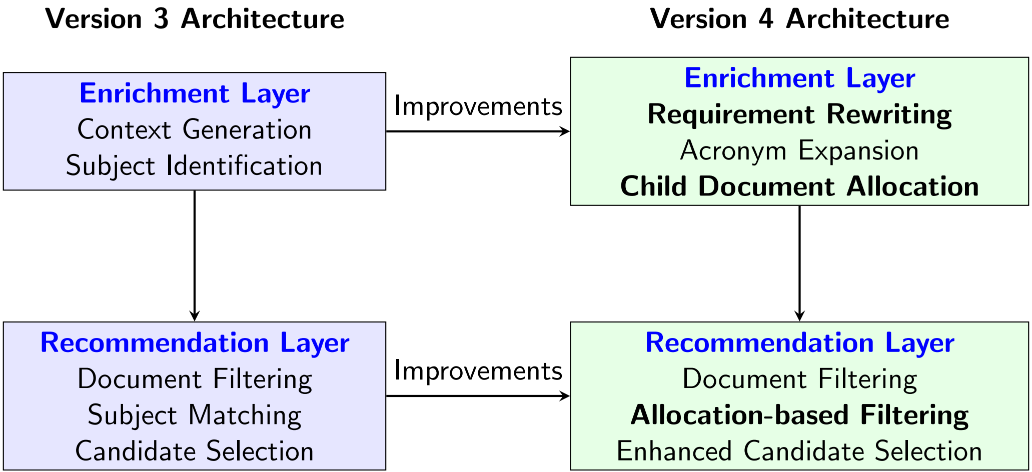
Five-Phase Methodology

1. Requirement Enrichment: Generate context, improve clarity
2. Document Identification: Filter potential parent documents
3. Detailed Requirement Analysis: Identify candidate parents
4. Confidence Assessment: Evaluate recommendation quality
5. Human Validation: Review and confirm final linkages

Key Principle

"Rather than attempting to fully automate trace link establishment, the approach focuses on providing systems engineers with high-confidence recommendations that can be efficiently reviewed and validated."

System Architecture Evolution



Key Architectural Improvements

1. Requirement Rewriting

- Rewrites requirements based on SE best practices
- Expands acronyms
- Clarifies ambiguous references
- Restructures for clarity

Example

Original: "SOCC shall provide command capability."

Rewritten: "The Satellite Operations Control Center shall provide command capability for spacecraft operations."

2. Allocation-Based Filtering

- Analyzes which child documents should receive allocated requirements
- Creates allocation-to-identification mappings
- Ensures hierarchical appropriateness
- Filters out semantically similar but hierarchically inappropriate matches

Implementation and Integration

Technical Implementation

- State-of-the-art Large Language Models with validation layers [3]
- Hallucination detection and error handling [11]
- Confidence-based recommendation filtering

Integration with SE Practices

- MagicDraw MBSE Plugin for seamless workflow integration
- Requirements traceability established within existing SE environment
- Standard configuration management for ongoing maintenance
- Human-in-the-loop design preserves SE oversight [2]

Experimental Dataset

Active Space Mission Development Project

- 636 Level 3 spacecraft requirements
- 670 Level 2 parent requirements
- 5 different parent documents

Document	Focus Area	Requirements
L2RD	Project-level requirements	177
CRD	Communications architecture	74
ERD	Environmental verification	266
OTRAD	Technical resource allocation	146
RRD	Radiation environment	7

Table: Document Distribution of the 670 Parent Requirements

Evaluation Framework

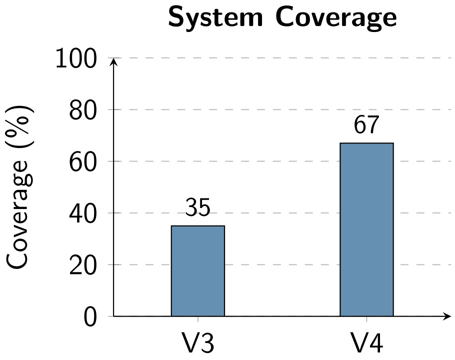
Evaluation Criteria

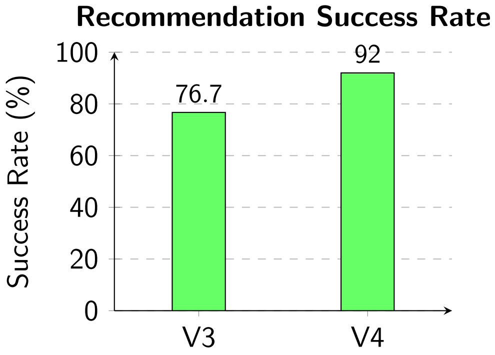
1. System Coverage: Percentage of requirements with recommendations
2. Recommendation Accuracy: Percentage of correct recommendations
3. Quality of Incorrect Recommendations: Weak/(Weak + Bad)

Two-Stage Assessment

- Automated assessment based on three criteria:
  - Correct document identification
  - Correct subject/allocation alignment
  - Correct document section placement
- Manual expert review with classification:
  - Strong, Moderate, Weak, Bad

Results: Performance Comparison





Key Results

Version 4 achieved significant improvements in both coverage and success rate.

Detailed Performance Metrics

Metric	Version 3	Version 4
Coverage	35% (220/636)	67% (425/636)
Total Links	275	797
Success Rate	76.7%	92%
Strong Links	155 (56.4%)	555 (70%)
Moderate Links	56 (20.3%)	176 (22%)
Weak Links	38 (13.8%)	57 (5.4%)
Bad Links	26 (9.5%)	7 (2.6%)
Moderate→Strong Conversion	67%	74.9%
Moderate→Weak/Bad Downgrade	8.7%	1.6%

Analysis of Key Improvements

What Drove the Performance Gain?

- "Bad" classifications (incorrect traces) were reduced by over 70%.
- The rate of downgrades from expert review (automated grade was too high) dropped from 8.7% to only 1.6%.
- "Strong" links (high-confidence, correct traces) increased from 56.4% to 70% of all recommendations.

Primary Drivers

The architectural enhancements of **Requirement Rewriting** and **Allocation-Based Filtering** were directly responsible for these gains.

Technical Insights

1. Requirement Quality is Paramount

- The clarity and structure of requirements significantly impact analysis quality [6]
- AI systems both benefit from and can help improve requirement writing practices

2. Systems Context Awareness Is Critical

- Document hierarchies and allocation relationships provide essential context [7]
- Effective AI tools must incorporate both semantic understanding and architectural awareness

3. AI Augmentation Outperforms AI Replacement

- The human-in-the-loop design provides optimal results [2]
- AI handles labor-intensive analysis; humans provide final validation

Practical Benefits for SE Teams

1. Significant Time Savings

- 53-106 hours saved on the test dataset alone [1]
- Analysis completed in hours vs. weeks
- Engineers focus on validation rather than initial discovery

2. Improved Analysis Consistency

- Uniform evaluation criteria applied across the entire requirement set
- Reduces variability between different engineers' approaches

3. Seamless Workflow Integration

- MagicDraw plugin works within the existing SE environment
- No disruption to established processes or tools

Limitations and Future Work

Current Limitations

- Dependency on requirement quality and document structure [6]
- LLM context length restrictions [5]
- Focus limited to vertical (parent-child) traceability
- 33% of requirements still require manual analysis

Future Directions

- Extending to horizontal traceability
- Tracing to architecture and verification artifacts [9]
- Using AI for requirement quality improvement [6]
- Broader SE applications beyond traceability [11]

Conclusion

- Developed a system that integrates LLMs with MBSE principles to transform requirements traceability [4, 7]
- Achieved dramatic performance improvements:
  - Increased coverage from 35% to 67%
  - Improved accuracy from 76.7% to 92%
  - Reduced analysis time by over 80% on the test dataset
- Successfully implemented as a MagicDraw plugin for seamless integration into existing SE workflows [2]
- Established a practical framework for AI augmentation that amplifies human capability rather than replacing expertise

References I

[1] Accuris, "Manual Requirements Management: The Hidden Drain on Your Engineering Budget," Feb. 4, 2025. [Online]. Available: https://accuristech.com/manual-requirements-management-the-hidden-drain-on-your-engineering-budget/

[2] National Institute of Standards and Technology (NIST), "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf

[3] X. Hou *et al.*, "Large Language Models for Software Engineering: A Systematic Literature Review," *arXiv preprint arXiv:2308.10620*, 2023. [Online]. Available: https://arxiv.org/abs/2308.10620

[4] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large Language Models for Software Engineering: Survey and Open Problems," in *Proc. 2023 IEEE/ACM Int. Conf. on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, 2023. [Online]. Available: https://arxiv.org/abs/2310.03533

[5] W. X. Zhao *et al.*, "A Survey of Large Language Models," *arXiv preprint arXiv:2303.18223*, 2023. [Online]. Available: https://arxiv.org/abs/2303.18223

References II

[6] A. Vogelsang and J. Fischbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline," *arXiv preprint arXiv:2402.13823*, 2024. [Online]. Available: `https://arxiv.org/abs/2402.13823`

[7] T. W. W. Aung, H. Huo, and Y. Sui, "A Literature Review of Automatic Traceability Links Recovery for Software Change Impact Analysis," in *Proc. 28th Int. Conf. on Program Comprehension (ICPC)*, 2020, pp. 14–24. [Online]. Available: `https://yuleisui.github.io/publications/icpc20.pdf`

[8] J. Cleland-Huang, O. Gotel, and A. Zisman, Eds., *Software and Systems Traceability*. Springer, 2012. [Online]. Available: `https://link.springer.com/book/10.1007/978-1-4471-2239-5`

[9] A. Ferrari, S. Abualhaija, and C. Arora, "Model Generation with LLMs: From Requirements to UML Sequence Diagrams," *arXiv preprint arXiv:2404.06371*, 2024. [Online]. Available: `https://arxiv.org/abs/2404.06371`

[10] B. Wang, C. Wang, P. Liang, B. Li, and C. Zeng, "How LLMs Aid in UML Modeling: An Exploratory Study with Novice Analysts," *arXiv preprint arXiv:2404.17739*, 2024. [Online]. Available: `https://arxiv.org/abs/2404.17739`

References III

[11] Y. Zhang *et al.*, "ChatCoder: Chat-based Refine Requirement Improves LLMs' Code Generation," *arXiv preprint arXiv:2311.00272*, 2023. [Online]. Available: https://arxiv.org/abs/2311.00272

Thank you!

Questions?

Henock Legesse (henock.a.legesse@nasa.gov)
Stanley 'Skip' Bicknell (stanley.l.bicknell@nasa.gov)