Qualification Cases for AI-Supported Workflows

Simon Diemert
Department of Computer Science
University of Victoria
Victoria, Canada
sdiemert@uvic.ca

Baichuan Zhao
Department of Computer Science
University of Victoria
Victoria, Canada
baichuanzhao@uvic.ca

Jens H. Weber
Department of Computer Science
University of Victoria
Victoria, Canada
jens@uvic.ca

***Abstract*—Existing approaches for tool evaluation and qualification are based on assumptions that do not hold for AI-enabled tools. Building on the idea of preparing an assurance case for a safety-critical system, this paper introduces the notion of a "qualification case" that marshals arguments and evidence about the adequacy of AI-supported workflows used in the development of critical systems. An AI-supported workflow for preparing Failure Modes, Effects, and Diagnostic Analysis (FMEDA) is used to illustrate the notion of a qualification case.**

***Index Terms*—AI Workflow, Qualification, Assurance Case**

I. Introduction

The complexity of software-intensive systems continues to grow as organizations introduce automated, autonomous, and adaptive capabilities to address challenges in high-risk application areas in transportation, healthcare, energy, manufacturing, and defence. Many of these systems are *safety critical*: malfunction or insufficient behaviour could lead to severe injury or loss of life. To manage this risk, systems and software engineers follow rigorous development standards that prescribe specific activities across the product development lifecycle, like those described in ISO 26262.

When tools are used to support the development of a critical system, their impact must be *evaluated* and, if the tool could compromise the safe operation of the system, it must be *qualified*. For instance, static source code analysis tools are usually evaluated as "moderate risk" because they do not directly influence the deployed code, but they might fail to detect a software defect. By comparison, a compiler is usually classified as "high risk" because it directly influences the deployed executables. Industry-specific standards prescribe tool qualification activities with the level of rigour increasing with risk. Qualification activities range from black-box testing on representative inputs to compliant development lifecycles.

The remarkable capabilities of modern artificial intelligence (AI) technologies, driven by foundation and large language models (LLMs), are rapidly re-shaping systems and software engineering. A growing body of work shows that AI-enabled tools can support critical systems development, including for hazard analysis and assurance case preparation [1]–[3]. However, given the substantial risk posed by critical systems, methods for establishing trust in these tools that balance their benefits with potential risks are urgently needed [4]. Moreover, understanding the impact of AI-enabled tools on human performance is also essential, due to the importance of experience and judgement in engineering [5].

Our view is that, despite their impressive capabilities, modern AI-enabled and agentic tools ***are still tools***. They must be applied by experts who bring knowledge, experience, and judgment, and who remain accountable for decisions.

A. What's Different About AI-Enabled Tools?

Existing tool evaluation and qualification methods rest on three assumptions that do not hold for AI-enabled tools.

1) Humans Think, Tools Automate: Existing methods assume there are **distinct roles** for humans and tools in systems development: tools automate repetitive tasks while humans perform creative and intellectual tasks requiring judgement, including interpreting the output from tools. AI-enabled tools blur this boundary, often "collaborating" with humans to perform tasks requiring interpretation and synthesis. For example, hazard analysis methods used for critical systems involve structured brainstorming of "what if" scenarios, an activity that AI can support [3], [6].

2) Determinism: Existing methods assume that tools are **functionally deterministic**, producing the same output for identical inputs[1]. As a result, qualification methods are focused on discovering and mitigating systematic defects in tools that could impact system development. Modern AI-enabled tools, especially those built with LLMs, produce variable outputs that are sensitive to subtle changes in the context or prompt, and so do not offer the same degree of determinism. This undermines defect-oriented qualification approaches based on functional or black-box testing, which in our experience is heavily relied on to qualify non-AI tools.

3) Tools Must Be Reliable: Following the two assumptions above we see that existing tool evaluation and qualification methods implicitly focus on tool reliability; e.g., a static analysis tool must detect all syntax rule violations. However, the broader applicability of AI-enabled tools means that there might be other properties of interest. For instance, for brainstorming "what if" hazard scenarios, there is value in a tool producing variable results that stimulate thinking, even if some of the results produced are incorrect (i.e., unreliable).

[1]We distinguish "functional determinism" from true "determinism", since many AI tools have hidden configuration inputs (e.g., random seeds) that mean they are still deterministic. To a user, AI tools might appear non-deterministic.

B. Qualification Cases for AI-Supported Workflows

A new approach to qualification is required in cases where AI-enabled tools are used to support engineering for critical systems. More specifically, it is necessary to lift qualification from focusing on deterministic tools to ***AI-supported workflows***, where tasks are performed collaboratively by humans, AI models, and conventional (non-AI) tools.

Test-based methods of tool qualification are insufficient at the workflow level. A holistic approach for establishing trust in AI-supported workflows is required that accounts for conventional tools, AI desiderata, and the roles and capabilities of human experts in workflows. Establishing trust in a workflow entails describing the role of humans, AI models, and tools for each task and then presenting an evidence-based argument that they adequately fulfill these roles.

Leveraging the well-known assurance case method, this short paper introduces the concept of a ***qualification case*** (QC) for marshalling these arguments and evidence. To illustrate our proposal, the paper briefly describes an AI-supported workflow for performing Failure Modes, Effects, and Diagnostic Analysis (FMEDA) and sketches a qualification case.

II. Introducing Qualification Cases

Assurance cases (AC) are arguments, supported by evidence, that a system satisfies its intended objectives. They are often expressed using specialized notations, such as the Goal Structuring Notation (GSN), and are required for certification in several sectors [7]. A sample AC expressed using GSN appears in Fig. 1; claims ("goals") are recursively decomposed into sub-claims through argument strategies and are supported by evidence ("solutions"). Preparing an AC is especially important for systems using novel designs or depending on unconventional assurance rationale, like autonomous vehicles [8].

In the last 30 years, researchers have developed a suite of methods for developing and managing ACs. Analogously, to qualify AI-supported workflow an ***qualification case*** (QC) can prepared: *a reasoned and compelling argument, supported by a body of evidence, that a workflow will achieve its intended objectives when applied to the task(s) that it was designed to complete in a defined context* (adapted from [7]).

In the above, a ***workflow*** is a structured sequence of sub-tasks performed by some combination of conventional (deterministic) tools, humans, and AI models, a definition generalized from [9]. A workflow's tasks might be performed in sequence, conditionally, or iteratively and might be orchestrated by a higher-order entity, a human or an AI.

A workflow targets a specific context whose definition might include assumptions about human expertise, available data, or the type of system. For instance, the QC for a workflow designed for an automotive failure mode analysis might not apply to a nuclear power plant.

A workflow is an ***AI-supported workflow*** when one or more of its (sub-)tasks is performed by an AI, including generative models (LLMs, VLMs, etc.), models trained for classification or prediction, or other forms of AI such as expert systems.

A. Themes in Qualification Cases for AI-Supported Workflows

Since our notion of a QC is new, we are not ready to define a generic argument structure (i.e., a "pattern" [7]). However, we believe the following themes should be woven into QCs.

1) Task Decomposition and Desiderata: QCs should decompose a workflow's tasks into sub-tasks that are delegated to conventional tools, AIs, or humans. For each (sub-)task, the QC should argue that the entity performing the task satisfies desirable properties ("desiderata"). E.g., if an LLM is used to generate reports, then *correctness* of the statements in the report might be important.

2) Empirical Evidence: Where desiderata are defined, evidence should be available to support claims that the entity adequately satisfies the desirable properties. For an LLM generating a report, the QC should reference experimental results showing that the LLM generates reports containing incorrect statements at an acceptably low rate.

3) Verification of AI Output: AI-supported workflows should include verification tasks, performed by humans or conventional tools, that check an AI's output. The QC should contain an argument about the adequacy of the checking procedure(s). E.g., if the workflow includes a task for a human to spot-check an AI-generated report, then the adequacy of the sampling procedure should be argued in the QC.

4) Human Cognition: QCs should argue that tasks allocated to humans are achievable for experts in the domain, and that the tasks adequately engage humans so they can maintain a sufficient cognitive context, perhaps through the use cognitive forcing functions [10]. The QC should contain evidence about the efficacy of interventions for supporting human cognition.

III. Illustrative Example: a QC for AI-FMEDA

To illustrate the idea of a QC, this section briefly describes an AI-supported workflow to support FMEDA and then sketches (part of) a QC for that workflow.

FMEDA is a hazard analysis methodology for evaluating hardware architectures in the development of critical systems. Because LLMs might produce inconsistent and unreliable results when a single prompt contains many instructions and requirements, we have decomposed an AI-supported FMEDA workflow with four sub-tasks which, in some cases, are further decomposed into individual queries issued to an LLM. The top-level tasks are: i) knowledge extraction; ii) failure mode identification; iii) diagnostic capability analysis; and iv) report generation. LLMs have different roles in each sub-task. Moreover, verification "gates" are used to check AI output using a combination of human review and automated checks written in conventional programming languages. Human tasks are carefully designed to encourage cognitive engagement while also not burdening experts with lengthy review tasks.

We evaluated the FMEDA workflow using ISO 26262's tool evaluation framework and ranked it at *Tool Confidence Level 2* (moderate criticality). The QC for the AI-FMEDA workflow is sketched in Fig. 1. The top-level of the argument is decomposed over the workflow's four tasks (`S0002`). Then it focuses on a sub-argument for merging similar failure modes,

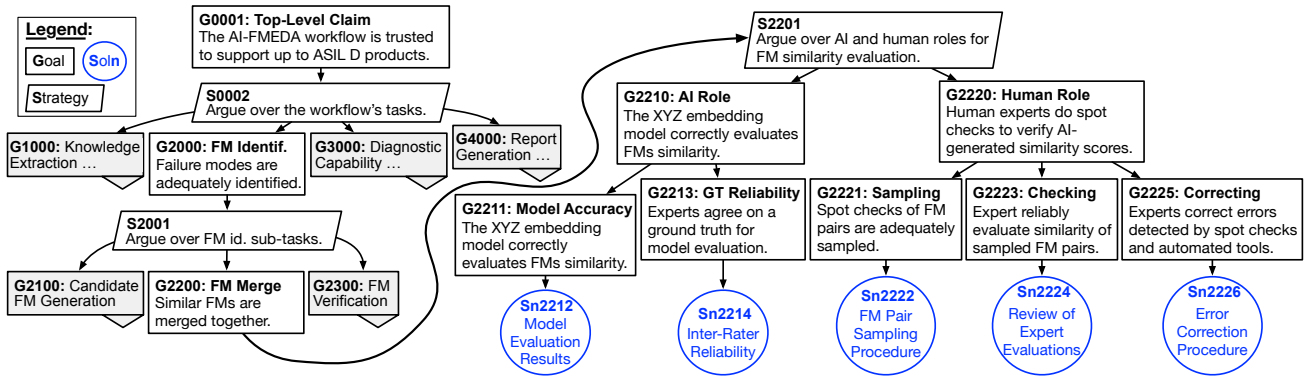


Fig. 1. GSN sketch of the QC for the AI-FMEDA workflow, focused the failure mode (FM) merge sub-task in the workflow.

as shown in the argument under G2200. The AI-FMEDA workflow uses cosine similarity between pairs of failure mode text embeddings to determine if two failure modes are similar enough to be merged. This sub-task involves both the text embedding model and a human (S2201).

1) AI Role: Different AI models, and specifically text embedding models, might have different performance profiles for technical topics. For the failure mode merging sub-task, we performed an experiment to select an embedding model and similarity metric that accurately ranks failure mode similarities as compared to an expert derived ground truth. We defined a six-level failure mode similarity rubric. Two experts ranked a sample of failure mode pairs using the rubric and then we compared the expert rankings with similarity metrics produced using different embedding models (G2211, Sn2212). Correlation between the expert-derived ground truth and the AI derived similarity measures was measured to determine which embedding model(s) had acceptable performance for this task. Inter-rater reliability measures, such as Cohens Kappa, confirmed that the expert-derived ground truth was reliable (G2213, Sn2114). All evaluations are performed under fixed software and hardware configurations. The evaluation reports (Sn2212) are the evidence to support G2211.

2) Human Role: Because LLM outputs are nondeterministic and exhaustive testing is infeasible, human oversight increases confidence that the sub-task goals are satisfied. For AI-FMEDA, two types of errors might arise. The first are structural errors, which are checked by automated tools (e.g., scripts). For example, data schema violations can be checked by a script. The second are semantic errors ("hallucinations"), where an LLM generates plausible results that are incorrect. The human expert is required to detect these errors. We embed "spot checks" into the AI-FMEDA workflow to detect semantic errors arising during failure mode merging (G2220). Failure mode pairs are scored for similarity by the AI are randomly sampled and are independently spot checked by a human expert, who also corrects errors. If too many errors are detected, then more checking might be required. The spot check mechanisms creates "cognitive friction" for humans in the workflow, helping to maintain awareness of the FMEDA's contents and managing "cognitive debt" [10], [11].

To detect errors, we define error correction procedure that accounts for both structural and semantic errors. For both failed structural checks and semantic spot checks, the errors are directly corrected by a human(G2225). Additional checking is warranted if enough spot checks fail (not shown in Fig. 1 for brevity).

IV. Closing Remarks and Next Steps

Developing QCs for AI-supported workflows is a new research direction with many open challenges, including: 1) deciding how to allocate (sub-)tasks to AI models, conventional automation, and humans; 2) defining standardized "evaluative protocols" that produce qualification evidence; and 3) establishing acceptable qualification arguments and common pitfalls. Moving forward, we will continue developing the AI-FMEDA workflow, and similar workflows, along with the corresponding QCs, and will distill reusable QC patterns based on our experience.

References

[1] S. Diemert and J. H. Weber, "Can Large Language Models Assist in Hazard Analysis?" in *SAFECOMP Workshops*, 2023, pp. 410–422.

[2] T. Viger *et al.*, "Evaluating AI-supported eliminative argumentation for developing reliable assurance cases," *Em. Sof. Eng.*, vol. 31, no. 4, 2026.

[3] J. Lee, S. Park, S. Oh, and B. Ma, "Can large language models automate the HAZOP process without human intervention?" *Saf. Sci.*, vol. 194, 2026.

[4] S. Diemert *et al.*, "Balancing the Risks and Benefits of Using Large Language Models to Support Assurance Case Development," in *Computer Safety, Reliability, and Security*. Springer, 2025, pp. 209–225.

[5] M. S. Graydon and S. M. Lehman, "Examining Proposed Uses of LLMs to Produce or Assess Assurance Arguments," Tech. Rep., Mar. 2025.

[6] S. Diemert, J. Goodenough, J. Joyce, and C. Weinstock, "Incremental assurance through eliminative argumentation," *Journal of System Safety*, vol. 58, no. 1, pp. 7–15, 2023.

[7] Assurance Case Working Group, "Goal Structuring Notation Community Standard (Version 3)," Safety Critical Systems Club, Standard, 2021.

[8] C. Paterson, R. Hawkins, C. Picardi, Y. Jia, R. Calinescu, and I. Habli, "Safety assurance of Machine Learning for autonomous systems," *Reliability Engineering & System Safety*, vol. 264, p. 111311, Dec. 2025.

[9] J. Zhang *et al.*, "AFlow: Automating Agentic Workflow Generation," in *The 13th International Conference on Learning Representations*, 2024.

[10] S. Passi, S. Dhanorkar, and M. Vorvoreanu, "Addressing Overreliance on AI," in *Handbook of Human-Centered AI*. Springer, 2025.

[11] M. A. Storey, "From Technical Debt to Cognitive and Intent Debt: Rethinking Software Health in the Age of AI," Apr. 2026. [Online]. Available: http://arxiv.org/abs/2603.22106