Knowledge Requirements, Agentic Architectures, and Prompt Evolution for LLM-Supported Human Reliability Analysis

Michael Hildebrandt
Institute for Energy Technology, Norway, michael.hildebrandt@ife.no

Human Reliability Analysis (HRA) is a method-dependent and context-dependent activity. Large language models (LLMs) may reduce some clerical burden in HRA, but unconstrained use also introduces predictable failure modes: unsupported plant assumptions, generic human-factors prose, incomplete traceability, and plausible but unreviewed quantification. This paper argues that a useful safety function of LLMs in HRA is disciplined support within a structured analysis process. The paper proposes that method guidance, reference data, plant-specific information, prompt instructions, and agent skills should be represented in an LLM-facing repository (e.g. wiki) whose contents are reviewed, versioned, access-controlled, and retrievable as ground truth. A worked example shows how a safety analysis method can be captured as a formal specification, using method-step contracts, explicit uncertainty states, and invariants that prevent unsupported claims. The specification does not replace expert HRA judgement, but instead makes method steps explicit, constrains the agent to valid intermediate products, requires evidence for each claim, and exposes missing information before the analysis is accepted. The paper also discusses prompt quality assurance, the distinction between prompts and agent skills, adversarial review, and the role of multi-agent review in HRA workflows.

1. INTRODUCTION

Human Reliability Analysis is an interpretive engineering activity in which scenario definition, task decomposition, performance conditions, operating practice, and dependence between actions must be made sufficiently explicit to support probabilistic risk assessment. Many HRA decisions are made at the boundary between method guidance, plant-specific facts, and analyst judgement. This boundary is also where inconsistency and untraceable assumptions tend to enter the analysis.

LLMs are therefore attractive and hazardous for the same reason. They can summarise documents, propose decompositions, compare candidate interpretations, and draft structured justifications quickly. They can also produce fluent but unsupported analysis when they are asked to reason from incomplete context. In safety analysis this is a poor failure mode because a polished paragraph can hide that a PIF rating, a recovery assumption, or a timing judgement has no documentary basis. In this paper, the term slop is used to refer to low-discipline output that is plausible, generic, weakly evidenced, or padded beyond the available facts.

The proposed remedy is to make the work less free-form. The agent should be held to a scaffold consisting of a method specification, approved sources, schema-constrained intermediate products, explicit uncertainty states, and review gates. For HRA, this scaffold can be built around existing method structure.

The paper makes three contributions. First, it describes how structured specifications can reduce hallucination and slop in LLM-supported HRA by making the method executable as a sequence of constrained analytical steps. Second, it proposes an LLM-wiki as the controlled knowledge base for method guidance, HRA data, operational reference data, and plant-specific information. Third, it describes how prompts, agent skills, adversarial review, and multi-agent review can be used as quality-assurance mechanisms.

2. WHY FREE-TEXT LLM SUPPORT IS INSUFFICIENT

A general LLM model has no stable obligation to distinguish method guidance from plant procedure, generic nuclear knowledge from local conduct of operations, or accepted HRA data from a convenient

estimate. Retrieval-augmented generation can place relevant documents in the model context, and the original RAG work showed the value of combining parametric models with external retrieval for knowledge-intensive tasks [5]. HRA-specific work has also begun to combine IDHEAS-DATA, knowledge graphs, and LLMs for base HEP estimation, with expert validation used to reduce data sparsity and hallucination risk [6]. The limitation is that retrieval supplies material, but it does not control the use of that material. A paragraph retrieved from a method report can still be applied at the wrong stage, blended with plant assumptions, or used to justify a rating that the method would treat as unresolved.

For HRA, the main error modes include that the model may invent plant details, assume that a procedure contains a step it has not seen, over-generalise from one reactor type to another, assign a PIF based on ordinary language rather than method criteria, or produce a HEP discussion without preserving the qualitative basis. Such errors may be reduced when the agent must produce small, typed outputs that a validator, a reviewer, or a second agent can inspect.

Table 1: Free-text failure modes and scaffold controls

Failure mode in LLM-assisted HRA	Scaffold control	Result expected from the agent
Invented or assumed plant facts	Plant-specific facts must come from approved wiki pages or uploaded source packets.	Report the claim as unsupported and ask for the missing source.
Generic PIF discussion	Each PIF is linked to a method definition, evidence item, and analyst judgement state.	Produce a PIF table with evidence and uncertainty, not prose alone.
Method drift	Workflow is represented as typed steps with allowed outputs.	Complete the current method step before moving to the next.
Hidden quantification assumptions	HEP inputs are separated from rationale and reviewer decisions.	Expose assumptions before calculation and mark unresolved values.
Overconfident report language	Draft text must preserve evidence status and open issues.	Use qualified statements where evidence is incomplete.

3. IDHEAS-ECA AS A METHOD SCAFFOLD

IDHEAS-G was developed by the U.S. Nuclear Regulatory Commission as a general methodology for HRA method development, with emphasis on application scope, scientific basis, HRA variability, and data for HRA [1]. IDHEAS-ECA is an application-specific method derived from that basis and intended for event and condition assessment in probabilistic risk assessment applications [2]. Its structure is directly relevant to LLM-supported work because it already divides the analysis into named objects and intermediate decisions.

In IDHEAS-ECA, the analyst defines the human action and its context, identifies critical tasks, considers whether timing alone can lead to failure, and analyses macrocognitive functions such as detection, understanding, decision-making, action execution, and interteam coordination [2]. The failure of the human action is modelled through cognitive failure modes and PIFs that support HEP calculation. The report also includes a worksheet, integrated human error data, and examples, which means the method contains both process guidance and data-oriented elements [2].

This is a strong starting point for agent support. The agent does not need a broad instruction to perform HRA. It needs a sequence of permitted operations: define the scenario scope; extract candidate human actions; formulate the HFE; identify critical tasks; map each task to macrocognitive functions; identify applicable failure modes and PIFs; separate evidence from interpretation; prepare HEP inputs; and mark open review questions. Each operation can have an input schema, output schema, source policy, and review criterion.

Table 2: IDHEAS-ECA elements represented as typed analytical objects

Method element	Structured object	Why structure matters

Scenario and context	Scenario packet with initiator, plant state, systems affected, available cues, time window, and source list.	Prevents the model from importing facts from a different operating state or plant design.
Human failure event	HFE object with action boundary, success criterion, failure definition, PRA linkage, and exclusions.	Forces agreement on what failure means before PIF assessment begins.
Critical task	Task object with actor, procedure reference, cue, action, required coordination, and timing relation.	Keeps task analysis separate from quantification.
Macrocognitive function	Function assessment for detection, understanding, decision-making, action execution, and interteam coordination.	Makes cognitive interpretation explicit and reviewable.
PIF	PIF assessment with method definition, evidence, rating, rationale, uncertainty, and reviewer status.	Reduces vague human-factors language and supports independent review.
HEP calculation input	Quantification packet with accepted PIF states, timing assessment, dependency notes, and unresolved assumptions.	Separates calculation from unsupported narrative judgement.

4. WORKED EXAMPLE: FROM METHOD TEXT TO STRUCTURED SPECIFICATION

The example below is not a completed HRA and it does not claim plant-specific validity. The case is an illustrative post-initiator action in which operators must diagnose indications consistent with a steam generator tube rupture and transition to an appropriate response path. The purpose is to show how a method such as IDHEAS-ECA can be captured in a formal specification that an LLM agent can follow one step at a time.

The specification style is markdown-like because the same text should remain readable to analysts, editable in a wiki, and parseable by tools. The design principle is similar to computer-interpretable guideline languages such as PROforma, where procedural knowledge is represented as plans, decisions, actions, and enquiries rather than as prose alone [3, 4]. This does not require that HRA become software code. The point is to represent the analysis workflow in a structured text form with stable identifiers, typed steps, explicit inputs, and allowed transitions.

```
---
type: hra-method-spec
method-spec-version: 0.1
method-id: idheas-eca-assisted-analysis
title: IDHEAS-ECA Assisted HRA Workflow
source-method: NUREG-2256
requires-sources: [method-guidance, scenario-packet, procedures, plant-context]
---

# IDHEAS-ECA Assisted HRA Workflow

## Step 1 [id: define-hfe, enquiry]
Check: scenario packet contains initiator, plant state, action boundary, success
criterion, and PRA linkage.
- Complete -> #identify-critical-tasks
- Missing required field -> #request-source-gap-review

## Step 2 [id: identify-critical-tasks, decision]
Decision: decompose the HFE into tasks whose failure can produce the HFE.
- Task is necessary for HFE success -> #map-macrocognitive-functions
- Task is contextual but not necessary -> #record-exclusion

## Step 3 [id: map-macrocognitive-functions, action]
```

```
Action: map each critical task to detection, understanding, decision-making,
action execution, and interteam coordination.
Output: one function-assessment object per applicable function.
- All applicable functions mapped -> #assess-pifs
- Function applicability uncertain -> #request-review-question

## Step 4 [id: assess-pifs, action]
Action: assess only PIFs with direct evidence or explicit analyst judgement.
Rule: do not infer plant-specific practice from generic method text.
Output: pif-assessment table with evidence, rating, rationale, uncertainty, and
reviewer status.
- PIF table complete -> #prepare-quantification-packet
- Unsupported rating found -> #request-source-gap-review

## Step 5 [id: prepare-quantification-packet, action]
Action: prepare HEP input packet using accepted PIF states and timing assessment.
Output: quantification-packet marked draft until human HRA review is complete.
- Reviewer accepts packet -> END
- Reviewer rejects packet -> #assess-pifs
```

A free-text instruction such as "analyse this HFE using IDHEAS-ECA" leaves too much hidden. It does not say what a valid HFE object contains, when a missing procedure excerpt should stop the analysis, or whether the agent is allowed to infer a PIF from general operational knowledge. The specification above narrows the task. It gives the agent a current step, a small set of allowed transitions, and an explicit source policy. It also provides a basis for validation, as required fields can be checked, step identifiers can be resolved, output tables can be inspected, and unsupported values can be rejected before they enter the report.

4.1. Method step contracts and analysis state

The example can be made stronger by treating each method step as a contract. A contract is a disciplined description of what must be true before the agent may act, what the agent is allowed to produce, and what must remain visible for review. This is the level at which proceduralization becomes useful, as it prevents the model from turning a method name into a broad request for plausible HRA prose.

A formalized HRA step should therefore name its purpose, required inputs, permitted sources, output object, uncertainty states, invariants, stop conditions, and review gate. The purpose keeps the step tied to the method. The required inputs prevent premature reasoning. The source policy separates method text, plant evidence, HRA data, and analyst judgement. The output object gives the reviewer something smaller and more stable than a paragraph. The uncertainty states prevent missing evidence from being laundered into narrative language. The invariants state what the analysis is never allowed to do, even if the model can produce a fluent answer.

Table 3: Formalized HRA method-step contract

Contract field	Meaning in HRA work	Example control
Required inputs	The minimum scenario, source, and method information that must exist before the step can proceed.	A PIF assessment cannot start until the HFE boundary and critical task are accepted or marked for review.
Permitted sources	The source classes the agent may use at this step, with method guidance separated from plant-specific evidence.	Generic IDHEAS-ECA text can define a PIF, but cannot prove local staffing or procedure practice.
Output object	The structured product of the step, small enough to inspect and stable enough to carry forward.	A critical-task object contains actor, cue, action, coordination need, timing relation, and source links.
Uncertainty state	The explicit status of each claim or value when evidence is incomplete or judgement is pending.	unsupported, source-present, analyst-judgement-required, reviewed-accepted, reviewed-rejected, not-applicable.

Invariant	A rule that cannot be violated by a fluent answer.	No plant-specific claim from generic method text; no quantification packet before unresolved assumptions are exposed.
Stop condition	A condition that prevents transition to the next step.	Missing procedure excerpt, unresolved task boundary, unsupported PIF rating, or conflicting source status.
Review gate	The human or independent review decision required before the object becomes accepted analysis.	Reviewer accepts, rejects, requests source, or marks the issue as analyst judgement with rationale.

The uncertainty state model is equally important. HRA contains judgements, but not all incomplete information is the same. A missing source, a source that has been retrieved but not yet interpreted, an issue requiring analyst judgement, a rejected inference, and a non-applicable factor should not all collapse into vague language. A small set of reviewable states gives the agent a disciplined alternative to filling gaps. It also helps the reviewer see whether the analysis is incomplete because evidence is absent, because interpretation is pending, or because a qualified analyst has deliberately accepted a judgement.

These contracts also make method invariants explicit. Invariants are rules that protect the meaning of the method. No PIF rating should be accepted without either evidence or a declared analyst judgement. No quantification packet should be prepared before the HFE boundary, critical tasks, timing basis, and unresolved assumptions are visible. No plant-specific conclusion should be drawn from generic method guidance. No final report sentence should upgrade draft evidence into accepted analysis. When such rules are written into the scaffold, a model failure becomes easier to detect because it appears as a contract violation rather than merely as a doubtful paragraph.

The next table shows a fragment of the same example after execution by a scaffolded agent. The values are illustrative. The important point is the shape of the output. Each claim has a method location, source status, and review status. The agent is required to expose missing information.

Table 4: Illustrative structured output for one critical task

Field	Illustrative value	Evidence or review status
HFE boundary	Failure to diagnose and enter the appropriate response path after indications consistent with SGTR.	Draft. Requires scenario packet and procedure excerpt.
Critical task	Recognise the diagnostic pattern and select the applicable response path.	Supported only if the procedure source lists the relevant cues.
Detection	Operators must detect abnormal radiation or level indications.	Evidence required from alarm list, procedure, simulator log, or event narrative.
Understanding	Operators must interpret the pattern as consistent with SGTR rather than a less severe transient.	Reviewer judgement required. Do not infer from generic PWR knowledge alone.
Decision-making	Operators must choose the correct transition path under time and information constraints.	Draft PIF assessment pending procedure timing and crew-role evidence.
Action execution	Execution depends on procedure step sequence and control availability.	Not rated until procedure and HSI references are present.
Interteam coordination	May apply if field teams, supervisor, or technical support centre are required before the credited action.	Applicability uncertain. Ask reviewer.

5. LLM-WIKI AS THE CONTROLLED KNOWLEDGE BASE

The scaffold requires a controlled place for source material. A document folder is not sufficient if the agent cannot distinguish method authority, plant-specific authority, examples, draft notes, and deprecated assumptions. The proposed LLM-wiki is a governed knowledge base designed for both

human reading and machine retrieval. It is a reviewed source layer with stable pages, metadata, ownership, review status, access control, and exportable structure.

The public or general section would contain method guidance, terminology, HRA data references, generic event-data summaries, worked examples, prompt and skill descriptions, and review checklists. A secured section would contain proprietary plant or installation information: procedures, conduct of operations, staffing assumptions, control-room and field interfaces, training material, simulator observations, local operating experience, and analysis assumptions approved for a specific project. An agent must know whether a statement is generic, project-specific, or not available to the current analysis.

The LLM-wiki should carry source status on each page: authoritative, reviewed, draft, historical, superseded, or restricted. It should also distinguish page types, because each page type has different authority. A method page can define a method concept. A procedure page can support a plant-specific task claim. An event record can support an operating-experience comparison. An assumption page can record an accepted project judgement. A prompt or skill page can define agent behaviour, but it cannot supply plant evidence. Retrieval should preserve these labels in the context sent to the agent. A PIF rating based on a reviewed procedure and a simulator observation should not have the same status as a rating based on a draft analyst note.

6. PROMPTS, AGENT SKILLS, AND QUALITY ASSURANCE

A prompt is an instruction issued for a specific interaction. It may define the task, scope, sources, output format, and constraints. A prompt is useful for one-off work, but it is not a stable method asset by itself. An agent skill is a reusable capability package: instructions, source hierarchy, examples, schemas, tool-use rules, validation checks, and escalation behaviour.

Prompt quality assurance should therefore be treated like method support quality assurance. A prompt or skill version should be tested against benchmark HRA cases, known failure cases, and reviewer comments. Changes should be logged with the reason for change and the expected improvement. Automated prompt optimisation methods such as OPRO and DSPy are relevant as research examples because they treat prompts and pipelines as objects that can be improved against task metrics [8, 9]. They should not be read as validation evidence for safety analysis use. In safety analysis, optimisation cannot be allowed to optimise only for surface agreement or completed output. The metric must include evidence faithfulness, correct use of method steps, appropriate refusal to infer missing plant facts, and preservation of uncertainty.

A practical prompt QA loop would keep a set of reference scenarios, expected intermediate products, and known traps. For example, one case may omit the procedure excerpt needed to rate action execution. A good prompt should cause the agent to report an evidence gap. A poor prompt may cause it to fill the gap with generic operator-action language. Reviewer feedback should be classified by failure type: source misuse, method misuse, unjustified inference, missing uncertainty, poor traceability, or report language problem. Candidate prompt changes can then be tested before being accepted into the skill library.

Table 5: Prompt and skill governance examples

Artefact	Quality question	Example control
Task prompt	Does it constrain the current step and output format?	Require a typed table and prohibit unsupported plant-specific inference.
Agent skill	Does it encode method sequence and source hierarchy?	Load method guidance first, then scenario packet, then plant sources, then examples.
Benchmark case	Does it reveal known failure modes?	Include incomplete source packets where the correct output is a gap.
Reviewer feedback	Can it improve future work?	Classify each correction by failure type and affected prompt or skill section.

Version record	Can a regulator or peer reviewer reconstruct what was used?	Record skill version, model version, source packet, and accepted reviewer changes.

7. ADVERSARIAL REVIEW THROUGH THE ANALYSIS LIFE CYCLE

Adversarial review is a structured challenge to the analysis. In HRA it means asking whether the draft analysis has overstepped the evidence, missed an alternative interpretation, applied a method element incorrectly, or concealed uncertainty. This practice is already familiar in engineering review. The LLM contribution is that a reviewer agent can apply the same challenge pattern repeatedly and cheaply, provided it is itself constrained by the same source rules.

Useful review prompts may include: "List every plant-specific claim that is not supported by the provided sources"; "Challenge each PIF rating using the IDHEAS-ECA definitions and the evidence table"; "Identify where the analysis confuses task difficulty with information availability"; "List alternative interpretations of the same cues and the evidence needed to distinguish them"; and "Find any wording that makes a draft judgement sound accepted." These prompts work best when they are attached to a specific method stage rather than applied once at the end.

At scoping, adversarial review should challenge the HFE boundary and success criterion. During task analysis, it should look for missing critical tasks and accidental inclusion of contextual tasks. During PIF assessment, it should challenge evidence, double counting, and broad human-factors labels. Before quantification, it should check that unresolved assumptions have not silently become inputs. During report drafting, it should check that accepted analysis, draft analysis, and open questions remain visibly different.

8. MULTI-AGENT REVIEW: USEFUL, BUT NOT MAGIC

Multi-agent debate has been studied as a way to improve factuality and reasoning by having several model instances propose and critique answers over multiple rounds [7]. These results do not validate multi-agent HRA, but they do justify considering role-separated review as a controlled draft-stage technique. In HRA, the most useful form is a role-separated review. One agent may act as method analyst, another as evidence reviewer, and a third as quantification checker. The reviewer agent should not merely disagree. It should cite the exact object it challenges and state whether the problem is missing evidence, method misuse, unsupported inference, or unresolved judgement.

The benefit of a multi-agent arrangement is diversity of attention. A single agent often continues along the frame established in its first answer. A second agent with a different role can interrupt that frame and ask whether the current HFE boundary is wrong, whether a PIF rating rests on a plant assumption, or whether a timing argument has bypassed the IDHEAS-ECA structure. This is close to the practice of independent review in safety analysis, but should be treated as draft-stage challenge rather than accepted peer review.

Multi-agent workflows are more expensive, produce longer audit trails, and can create false confidence when agents share the same model and the same blind spots. They can also generate persuasive objections that are not method-valid. For this reason, multi-agent review should be used at selected gates rather than for every sentence. The recommended gates are HFE definition, critical task list, PIF table, quantification packet, and final report language. The human analyst remains responsible for accepting or rejecting the findings.

9. DISCUSSION: BENEFITS AND LIMITS OF FORMAL SPECIFICATION

The central benefit of structured specification is that it changes the unit of review. Free text invites the reviewer to read paragraphs and infer what the analyst meant. A method specification produces objects: HFE boundary, task, cue, function assessment, PIF rating, source, uncertainty state, review decision.

These objects can be checked for completeness and consistency. They can also be reused. A later analyst can see not only the final HEP discussion, but the sequence of accepted and rejected intermediate judgements.

There are also limits, and they should be stated plainly. A parser can check that a PIF assessment has an evidence field. It cannot decide that the evidence is operationally sufficient. A knowledge graph can show that a step references a procedure and a control-room indication. It cannot know whether the analyst has misunderstood crew practice. A complete schema can also create false confidence if every field is filled with weak evidence. Formalization therefore reduces uncontrolled reasoning, but it does not remove the need for qualified HRA review.

10. CONCLUSION

LLM-supported HRA should be designed around constraint. The model is useful when it extracts, structures, compares, challenges, and drafts within a visible method scaffold. It is unsafe when it is allowed to convert incomplete context into confident prose.

The paper has proposed an architecture in which a governed LLM-wiki holds method guidance, HRA data, operational reference data, and secured plant-specific information; agent skills encode reusable method workflows; prompts are versioned and tested; adversarial review is applied at defined gates; and multi-agent review is used selectively for role-separated challenge. More research is needed to evaluate if such systems can reduce hallucination and slop by making unsupported assumptions visible before they enter the accepted safety analysis.

References

[1] J. Xing, Y. J. Chang, and J. DeJesus Segarra, "The General Methodology of An Integrated Human Event Analysis System (IDHEAS-G)," NUREG-2198, U.S. Nuclear Regulatory Commission, Washington, DC, May 2021.
[2] J. Xing, Y. J. Chang, and J. DeJesus Segarra, "Integrated Human Event Analysis System for Event and Condition Assessment (IDHEAS-ECA)," NUREG-2256, U.S. Nuclear Regulatory Commission, Washington, DC, October 2022.
[3] D. R. Sutton and J. Fox, "The syntax and semantics of the PROforma guideline modeling language," Journal of the American Medical Informatics Association, vol. 10, no. 5, pp. 433-443, 2003. doi: 10.1197/jamia.M1264.
[4] J. Fox, N. Johns, C. Lyons, A. Rahmanzadeh, R. Thomson, and P. Wilson, "PROforma: a general technology for clinical decision support systems," Computer Methods and Programs in Biomedicine, vol. 54, no. 1-2, pp. 59-67, 1997. doi: 10.1016/S0169-2607(97)00034-5.
[5] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems 33, pp. 9459-9474, 2020.
[6] X. Xiao, P. Chen, B. Qi, H. Zhao, J. Liang, J. Tong, and H. Wang, "KRAIL: A knowledge-driven framework for human reliability analysis integrating IDHEAS-DATA and large language models," Reliability Engineering & System Safety, vol. 265, article 111585, 2026. doi: 10.1016/j.ress.2025.111585.
[7] Y. Du et al., "Improving Factuality and Reasoning in Language Models through Multiagent Debate," arXiv:2305.14325, 2023.
[8] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large Language Models as Optimizers," arXiv:2309.03409, 2023.
[9] O. Khattab et al., "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines," arXiv:2310.03714, 2023.