PharmaSUG China 2026 - Paper AA-202

AI-Powered Specification Verification for Data Review Outputs Using ALS Cross-Checks

Keting Lv, Daiichi Sankyo (China) Holdings Co., Ltd.

ABSTRACT

We developed an AI-driven tool to streamline and standardize specification verification for clinical data review deliverables. Manual creation and review of output specifications (e.g., Exception Listings, Reconciliation Listings, and Metric Reports) are prone to typographical errors, inconsistent wording, and cross-document mismatches; they also require substantial effort to check and maintain. Our workflow automatically ingests key inputs such as the ALS and output specification documents, performs structured comparisons across fields and rules, and flags discrepancies including redundancies, typos, missing/extra requirements, inconsistent terminology, and logic mismatches. The tool generates issues in a batch and traceable manner, producing review-ready findings that can be used for peer review and vendor oversight. By shifting verification from ad hoc manual checks to systematic, repeatable AI-assisted controls, this approach improves consistency, reduces cycle time, and helps minimize specification-to-output defects before downstream programming and QC.

INTRODUCTION

Clinical trial programming teams produce a broad set of data review deliverables—exception listings, reconciliation listings, review datasets, and metric reports—that enable medical monitors, data managers, and statisticians to evaluate data quality and trial conduct. These deliverables are typically driven by two foundational artifacts:

1. **ALS (Analysis/Annotated Listing Specification or analogous ALS workbook)**, which frequently captures variable definitions, controlled terminology expectations, derivation rules, and dataset/structure conventions; and
2. **Output specification documents**, which define report-level requirements such as selection criteria, flags, grouping logic, column order, labels, titles/footnotes, and calculation rules.

In many organizations, the output specification is authored manually (often as Excel) and then reviewed through peer review or QC checklists. This manual process is vulnerable to several persistent issues:

- Typographical errors in variable names, formats, and labels
- Inconsistent terminology (e.g., “Screen Failure” vs “Screening Failure”) across outputs
- Missing or extra requirements that drift away from ALS definitions
- Logic mismatches (e.g., a flag definition that contradicts the ALS rule)
- Cross-tab inconsistencies (e.g., column order differs across similar listings)

These defects rarely remain “documentation-only.” They propagate into downstream programming, validation, and vendor deliverables, increasing cycle time and rework. In regulated environments, specification defects can also erode traceability and raise audit questions: “Where is the documented justification for this logic?” or “Why do two documents describe different requirements for the same variable?”

This paper describes an AI-powered workflow for **specification verification** that cross-checks ALS against draft output specifications and automatically produces a standardized discrepancy log. You will learn:

- How to structure inputs and normalize Excel-based specs for machine comparison
- How to design prompt and output schemas for consistent, review-ready issue logs
- How to implement a repeatable workflow that supports peer review and vendor oversight

- Practical controls and limitations when applying AI in clinical programming documentation QC

PLATFORM: USING DIFY TO OPERATIONALIZE THE WORKFLOW

To implement the workflow in a controllable, enterprise-friendly way, we leveraged **Dify**, a platform for building, deploying, and managing AI assistants and AI-enabled applications. Dify supports:

- **Visual workflow orchestration** (end-to-end pipelines, branching, tool steps)
- **Prompt management** and version control for iterative tuning
- **Dataset and knowledge-base management**, including RAG (retrieval-augmented generation) patterns
- **Tool/function calling**, enabling integration with deterministic utilities (e.g., Excel-to-JSON conversion, schema validation, export generation)
- **APIs** to integrate the AI workflow into internal systems and products

In this use case, Dify acts as the orchestration layer: it manages inputs (ALS/spec Excel files), triggers normalization, calls an LLM comparison step with strict schema constraints, and exports a reviewer-friendly issue log artifact. This design enables repeatable execution, easier governance (prompt/model/run metadata), and controlled scaling for multiple studies or vendors.

A SYSTEMATIC VIEW OF SPECIFICATION VERIFICATION

Specification verification is not merely “spellcheck.” In clinical deliverables, a high-impact discrepancy is often semantic or logical: an output may use a variable in a way that conflicts with ALS expectations, or it may claim to display a derived field that does not exist or is defined differently. Our approach frames verification as a structured comparison problem with three layers:

- **Syntactic consistency**: variable names, dataset names, formats, controlled terms, casing
- **Semantic consistency**: meaning of flags, definitions of populations, label wording consistency
- **Logic consistency**: selection rules, derived computations, grouping logic, thresholds

To operationalize these layers, we implemented an AI workflow that ingests two Excel workbooks (ALS and Draft Spec), converts them into structured JSON, and uses an LLM-based comparison agent to generate discrepancy findings in a fixed schema. **Figure 1** illustrates the end-to-end AI workflow architecture and control points.

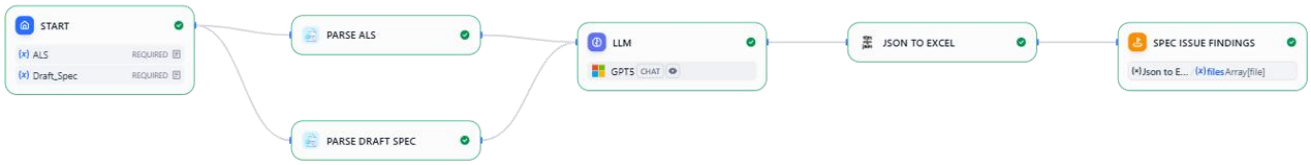

Figure 1. End-to-End Workflow for AI-Powered ALS vs. Spec Verification

The workflow’s key design principle is standardization: the tool does not “write” a spec. It verifies a spec by flagging differences, leaving final acceptance and updates to accountable reviewers. This keeps the system aligned with quality-by-design practices: AI is a structured assistant, not the ultimate decision maker.

INPUT NORMALIZATION: WHY EXCEL-TO-JSON MATTERS

Most ALS and specification documents are Excel files containing multiple tabs, merged cells, and human-oriented layout choices. These are readable for humans, but difficult for deterministic comparison. The

first workflow step converts Excel to JSON so that the comparison agent receives consistent, parseable records.

A minimal normalized record model often includes:

- Source (ALS vs Spec)
- Sheet name / output name
- Entity type (dataset, variable, report column, rule, label, footnote)
- Key fields (variable name, label, format, derivation rule, condition)
- Free text (notes)
- Row/column location (to support traceability back to Excel)

This design is critical because reviewers want to locate issues quickly. An "AI said there is a mismatch" statement is not helpful without actionable context such as which output, which column, and what the expected ALS definition is.

Table 1 summarizes typical discrepancy categories and examples that our workflow is designed to detect.

Discrepancy Category	Example Symptom in Draft Spec	Example ALS Reference	Risk if Missed
Typo / naming mismatch	AESEV typed as AESV	Variable list contains AESEV	Program fails or wrong variable used
Inconsistent label	"Serious AE Flag" vs "Serious Adverse Event Flag"	ALS label standard	Inconsistent output, reviewer confusion
Missing requirement	Spec column not in ALS	ALS includes required variable	Missing output content
Extra requirement	Spec requests a variable absent from ALS	ALS has no such definition	Unplanned derivation, scope creep
Controlled terminology drift	Values listed differ from ALS CT	ALS CT list	Inconsistent categorization

Table 1. Common Discrepancy Categories Detected by ALS Cross-Checks

WORKFLOW DESIGN AND IMPLEMENTATION DETAILS

The implemented workflow contains four functional stages:

1. **File intake**: user uploads ALS and Draft Spec (Excel workbooks)
2. **Normalization tools**: Excel-to-JSON conversion for both inputs
3. **LLM comparison**: AI agent compares the two JSON representations and outputs a structured discrepancy list
4. **Report generation**: discrepancy JSON is converted back into an Excel issue log for reviewer consumption

This pattern is intentionally "enterprise-friendly": it produces an output artifact that looks like a conventional QC log, which fits into existing SOPs and reviewer habits.

The workflow proceeds as follows::

1. Upload the ALS workbook as the first input.
2. Upload the Draft Output Specification workbook as the second input.
3. Run the workflow to normalize both workbooks into JSON.

4. Execute the LLM comparison step to identify discrepancies.
5. Export the discrepancy list into an Excel file to support review, triage, and sign-off.
6. Store the issue log with versioning metadata to support traceability across iterations.

To ensure traceability and compliance, we recommend the following operational controls:
• **Version locking**: record ALS/spec version, date, and file hash in the issue log.
• **Role separation**: developers tune prompts; reviewers approve resolutions.
• **Defect taxonomy**: use consistent categories (typo, missing requirement, logic mismatch).
• **Thresholding**: allow "severity" levels to prioritize review.
• **Audit trail**: store the model name, prompt version, and run ID.

The critical operational insight is that AI is most effective when it produces *structured, sortable, and reviewable* findings. Free-form narrative responses force humans to re-interpret output; structured results reduce cycle time.

Display 1 shows an example of the reviewer-facing discrepancy log layout

Date	Specification Name	Form	Compare QC Findings
2026/1/26	DS0000_000_Patient_Profile_Specification_V0.1_draft.xlsx	RE_CT	**Redundant** title 'Chest CT CHEST CT SCAN'; remove duplication.
2026/1/26	DS0000_000_Patient_Profile_Specification_V0.1_draft.xlsx	VS/VSM/VS_WT	**Typo** 'Date of Asssessment'; should be 'Assessment'.
2026/1/26	DS0000_000_Patient_Profile_Specification_V0.1_draft.xlsx	AESILIPR	**Typo** in dataset prefix: 'AESILIPRLIPR:LICOAMPCM' should be 'AESILIPR:LICOAMPCM'.

Display 1. Example Reviewer-Facing Issue Log Exported to Excel

PROMPT AND OUTPUT SCHEMA: MAKING FINDINGS REVIEW-READY

A common reason AI QC tools fail in production is inconsistent output. If the AI sometimes writes paragraphs and sometimes writes lists, it becomes hard to integrate into Excel-based review flows. We therefore enforce a strict output schema.

In our workflow, the LLM is instructed to output JSON only, as an array of objects, each with exactly these keys:

- **Date**: the execution date is recorded automatically
- **Form**: a category/area (for example, "Typo," "Missing Requirement," "Logic Mismatch," "Terminology")
- **Issue**: a reviewer-readable description of the discrepancy

To improve usability, you can extend the schema (if your tool allows) with optional but highly valuable fields such as: OutputName, Sheet, Cell, ALS_Expected, Spec_Observed, Severity, Recommendation. Even if you cannot add fields immediately, you can embed them in the Issue text with a consistent mini-template.

QUALITY, RISK, AND COMPLIANCE CONSIDERATIONS

An AI verification workflow must be designed with clinical programming realities in mind:

- **False positives**: the AI may flag legitimate differences (e.g., output-specific label variations). Mitigation: allow reviewers to mark "Not an issue" and capture rationale.
- **False negatives**: the AI may miss subtle logic conflicts. Mitigation: combine AI checks with targeted human QC for high-risk outputs.
- **Confidentiality**: ALS/spec content may include sensitive protocol logic. Mitigation: use enterprise-managed endpoints, data residency controls, access management, and retention policies.
- **Traceability**: auditors may ask how discrepancies were generated. Mitigation: store prompt templates, model versions, run timestamps, and immutable logs.

The goal is not to “replace QC,” but to transform QC into a more scalable and standardized activity where human expertise is focused on adjudication rather than hunting for mismatches.

EVALUATION STRATEGY: HOW TO MEASURE IMPACT

In our experience, the most persuasive metrics for stakeholders include:

- **Cycle time reduction**: time spent per output spec review iteration before vs after AI adoption
- **Defect capture rate**: number of discrepancies found pre-programming (and whether they would have become downstream defects)
- **Standardization**: consistency of issue taxonomy and completeness of logs across studies
- **Rework reduction**: number of programming change requests linked to spec defects

A practical evaluation design is a before/after study:

- Baseline: 3–5 historical studies with manual-only spec QC
- Intervention: same team, similar complexity outputs, AI-assisted verification
- Compare: median review hours, number of issues found per 100 spec rows, and post-programming defect counts

PROMPT ENGINEERING AS “QC RULE AUTHORING”

One creative but practical way to frame the method is to treat prompt templates as **QC rule authoring**. The LLM is not only interpreting language; it is executing a structured comparison policy encoded in the prompt:

- What counts as a discrepancy?
- What level of evidence is required to flag an issue?
- How should the issue be phrased so a reviewer can act immediately?

This perspective helps teams govern the tool: prompt changes should be reviewed like any other QC rule update.

HUMAN-IN-THE-LOOP: THE REVIEWER REMAINS THE DECISION MAKER

Even with strict schemas, reviewers must remain accountable for final decisions. The tool should support common adjudication outcomes:

- Accept issue and update spec
- Accept issue and update ALS
- Accept issue and document deviation (justified difference)
- Reject issue as false positive (with rationale)

When you implement this in Excel, add fields such as Status, Disposition, Rationale, Owner, and Due Date. This turns AI output into a controlled, SOP-compatible review package.

CONCLUSION

We presented an AI-powered specification verification workflow that cross-checks ALS and draft output specifications to identify discrepancies in a standardized, repeatable, and traceable manner. By converting Excel-based documents into structured JSON, applying an LLM comparison agent with a strict output schema, and exporting findings into an Excel issue log, the workflow shifts verification from ad hoc manual checking to systematic AI-assisted control.

This approach is especially valuable for clinical data review deliverables where specification errors propagate into programming rework and QC findings. The key to success is not simply model selection,

but workflow design: input normalization, deterministic schemas, reviewer-oriented output, and governance controls that preserve traceability and accountability.

As organizations prepare for higher volume, tighter timelines, and increased vendor collaboration, AI-assisted specification verification can become a practical quality enabler—improving consistency, reducing cycle time, and minimizing defects before downstream programming begins.

ACKNOWLEDGMENTS

The author thanks colleagues and stakeholders who supported the proof-of-concept evaluation, provided feedback on discrepancy taxonomy, and helped align the workflow with documentation QC practices and vendor oversight needs.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Keting Lv
Daiichi Sankyo (China) Holdings Co., Ltd.
86-18106850068
lvket3g3@daiichisankyo.com.cn