PharmaSUG China 2026 - Paper AI-186

Automating SDTM Programming with a Human-in-the-Loop Multi-Agent System

Wenqiang Chu

ABSTRACT

This paper presents a multi-agent system for automating the SDTM programming lifecycle with an explicit human-in-the-loop control model. The system comprises three core agents with distinct but complementary responsibilities. The Coder Agent performs completeness checking, generates structured implementation plans, and produces executable SAS programs from SDTM mapping specifications. The Reviewer Agent evaluates the generated outputs, produces a structured review report, and returns findings for downstream routing rather than modifying code invisibly. The Docs-Researcher Agent handles standards-related knowledge queries from other agents by searching SDTMIG documents and the CDISC Library, returning grounded evidence snippets. These agents are coordinated by a deterministic orchestrator that advances execution and pauses only at explicit human decision gates. The current system supports question-answer clarification, plan review, validation handoff, review decision, audit manifests, and a lightweight web UI for monitoring and intervention. This architecture reduces manual effort, improves consistency, strengthens traceability, and turns SDTM standardization into a scalable AI-assisted workflow rather than a purely manual bottleneck.

INTRODUCTION

Clinical trial data standardization is a critical component of regulatory submission. As a core part of CDISC standards, SDTM requires transforming raw clinical data into standardized domain datasets. Traditional SDTM programming is highly labor-intensive because programmers must simultaneously interpret SDTMIG rules, study-specific mapping specifications, source data structure, macro usage conventions, and derivation logic. This work is time-consuming, difficult to review consistently, and vulnerable to omission and interpretation drift.

Recent advances in large language models (LLMs) make it possible to move from isolated code-generation prompts toward workflow-aware agent systems. In such a system, the LLM provides reasoning and language generation ability, while an orchestrator manages state, tool usage, inputs, outputs, and decision boundaries. This distinction is important for regulated programming work: autonomy is useful, but uncontrolled autonomy is risky. As a result, our system does not rely on a free-form agent swarm. Instead, it uses a fixed orchestration model with explicit states, structured outputs, auditable artifacts, and required human intervention at key checkpoints.

This project aims to develop a multi-agent SDTM automation system that transforms study mapping specifications into implementation plans, SAS programs, validation artifacts, review reports, and replayable audit records. By encapsulating SDTM programming knowledge into reusable Skills and combining the collaborative behavior of the Coder Agent, Reviewer Agent, and Docs-Researcher Agent, the system improves both the efficiency and the governance of clinical data standardization.

SYSTEM ARCHITECTURE

Figure 1 provides a detailed description of the system architecture, including the agent collaboration mechanism as well as the system's input, output, and other related content. Figure 2 provides the overall orchestrator workflow of the system and the human intervention points. the Further details will be introduced in the next section.

CORE COMPONENTS

MULTI-AGENT COLLABORATION MECHANISM

The current system contains three core agents with isolated roles and context boundaries.

The Coder Agent is responsible for:

- checking whether the current inputs are sufficient
- requesting clarification when critical information is missing
- generating a structured implementation plan
- generating SAS code and related delivery artifacts such as Todo items

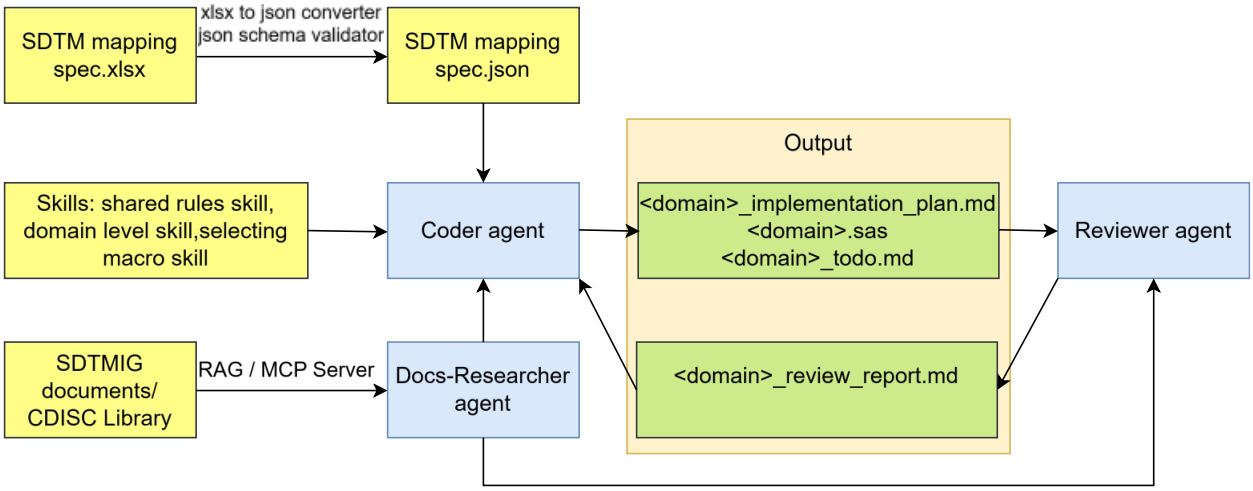

Figure 1. Overall Architecture Design of The Multi-Agent System

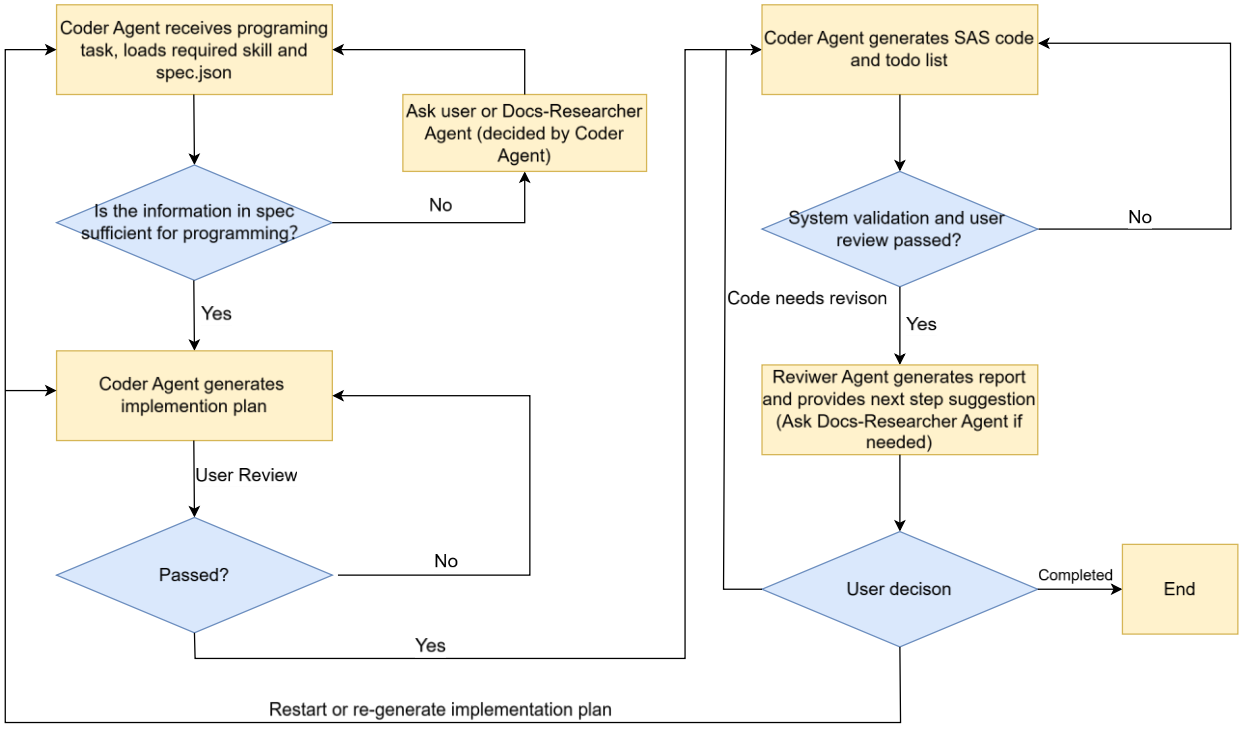

Figure 2. The overall orchestrator workflow of the system

The Reviewer Agent is responsible for:

- reading the factual outputs of the current iteration
- evaluating implementation quality and standards alignment
- producing a structured review report with stable finding identifiers
- routing findings back into the orchestrated workflow through a human decision gate

In the current design, the Reviewer Agent does not silently overwrite the Coder output. Instead, it produces a review report, after which a human decides whether to approve the run, ignore selected findings, return the work to the Coder, or fail the run. This preserves traceability and makes the correction path explicit.

The Docs-Researcher Agent serves as the standards knowledge retrieval specialist and is described separately below.

HUMAN-IN-THE-LOOP DESIGN

Human participation is a first-class feature of the system rather than an exception path. The current workflow includes four explicit intervention gates:

1. Clarification gate
   Triggered when the Coder or Reviewer needs study-specific or non-authoritative information that should not be guessed.
2. Plan review gate
   Triggered after the Coder produces the implementation plan. A human may approve the plan, request revision, or fail the run.
3. Validation handoff gate
   Triggered after validation. A human may return the work to the Coder, route it to the Reviewer, or fail the run.
4. Review decision gate
   Triggered after the Reviewer produces a report. A human may approve the run, ignore selected findings and still approve, return the work to the Coder, or fail the run.

This design is important for SDTM programming because many gaps are not purely technical. Some missing information concerns sponsor conventions, source sheet interpretation, macro parameter choices, or contradictory mapping instructions. In those cases, the system should surface the uncertainty instead of fabricating a plausible but unverified answer.

SPECIFICATION CONVERTER AND VALIDATOR

The Specification Converter transforms SDTM mapping specifications from Excel (.xlsx) format into JSON and validates the output against a predefined schema. Compared with Excel, JSON offers better support for programmatic processing, nested structure, schema-driven validation, LLM consumption, and version-controlled change tracking. This hybrid design keeps Excel as the authoring format while providing a machine-friendly representation for downstream automation.

The converted JSON is expected to preserve common metadata such as:

- variable name
- label
- type and format
- codelist

- source and origin
- derivation rules
- implementation notes

In practice, implementation notes are especially valuable when they indicate that a variable can be built through a standard macro, because this improves macro-selection accuracy during automated planning and generation.

SKILLS SYSTEM

The system uses a structured skill layer to package reusable programming knowledge. Three core skill categories are used in the current design:

- selecting-macros: provides macro selection rules and usage guidance
- common-sdtm-programming-rules: defines general SDTM programming workflow and cross-domain best practices
- sdtm-<domain>-builder: provides domain-specific build rules and implementation expectations

These skills are loaded into context in a controlled manner so that the Coder and Reviewer receive the knowledge relevant to the current state and domain without unnecessary context pollution.

MACRO LIBRARY

The Macro Library serves as a centralized repository of reusable SAS macros for SDTM programming as described in Table 1, following a hierarchical JSON structure. It provides a structured, machine-readable catalog of macro definitions that enables the Coder Agent to select and apply appropriate macros during code generation. The macro library is referenced by the `selecting-macros` skill, which provides guidance on selecting appropriate macros for specific SDTM programming tasks. The Coder Agent consults this library when generating domain programs to ensure correct macro usage and parameter passing. Here is an example of the macro library in JSON format:

```json
{
  "metadata": {
    "source": "path/to/source",
    "total_macros": 10,
    "description": "SDTM Macro Library for SAS programming"
  },
  "macros": [
    {
      "name": "%macro_name",
      "description": "Macro purpose and functionality",
      "note": "Additional notes or prerequisites",
      "parameters": [
        {
          "name": "parameter_name",
          "description": "Parameter description"
        }
      ],
      "examples": [
        {
          "description": "Example description",
          "code": "%macro_name(param=value)"
        }
      ]
```

```
      }
    ]
}
```

Macro Name	Function	Use Case
%sdtm_date2iso8601	Date to ISO8601	--DTC variable conversion
%sdtm_studyday	Study day derivation	--DY variable calculation
%sdtm_visit	Visit mapping	VISIT/VISITNUM derivation
%sdtm_blfl	Baseline flag	ABLFL assignment
%sdtm_lobxfl	Extended baseline flag	Baseline based on RFXSTDTC
%sdtm_epoch	Epoch derivation	EPOCH assignment
%sdtm_seq	Sequence number	--SEQ derivation
%sdtm_spilt_supp	Create SUPP-- dataset	Main domain and SUPP domain separation
%sdtm_rfpendtc	Participation end date	RFPENDTC derivation
%sdtm_finalize	Dataset finalization	Attribute setting and output

Table 1. Macro library integrated in the system

AGENTIC RAG

The Docs-Researcher Agent is the knowledge retrieval specialist responsible for querying SDTM standards and metadata. It employs a hybrid retrieval approach combining CDISC Library API integration via Model Context Protocol (MCP), local vector database retrieval (RAG), and knowledge graph querying. The local SDTMIGs are converted into structured Markdown knowledge bases and a Neo4j knowledge graph, enabling semantic search and relationship-based queries across domains, variables, rules, examples, and controlled terminology. Through the CDISC Library MCP server, the agent can access structured tools for querying SDTM, ADaM, CDASH standards, and controlled terminology packages in real time. This hybrid, multi-source architecture ensures standards compliance by grounding AI responses in authoritative CDISC documentation while providing flexibility for both online and offline development environments. The agent serves both the Coder Agent during code generation and the Reviewer Agent during validation, reducing manual documentation lookup and minimizing hallucination errors.

VALIDATION LAYER

The system includes a validation stage between code generation and review.

The validation stage currently focuses on structured, deterministic checks such as:

- placeholder and unfinished marker detection
- simple syntax-like or formatting issues
- rule-based variable coverage checks
- machine-readable pass/fail reporting

The validation output is written as a structured validation report that can be consumed by both the human handoff gate and the Reviewer Agent.

WEB UI AND OPERATIONAL VISIBILITY

The current system also includes a lightweight web UI for operational use. The UI is intentionally simple, but it adds important governance and usability capabilities:

- starting a run for a specific study + domain + SDTMIG version
- observing a chronological run feed and run state
- responding to human clarification questions
- reviewing plans before code generation
- making post-validation and post-review routing decisions
- inspecting business outputs

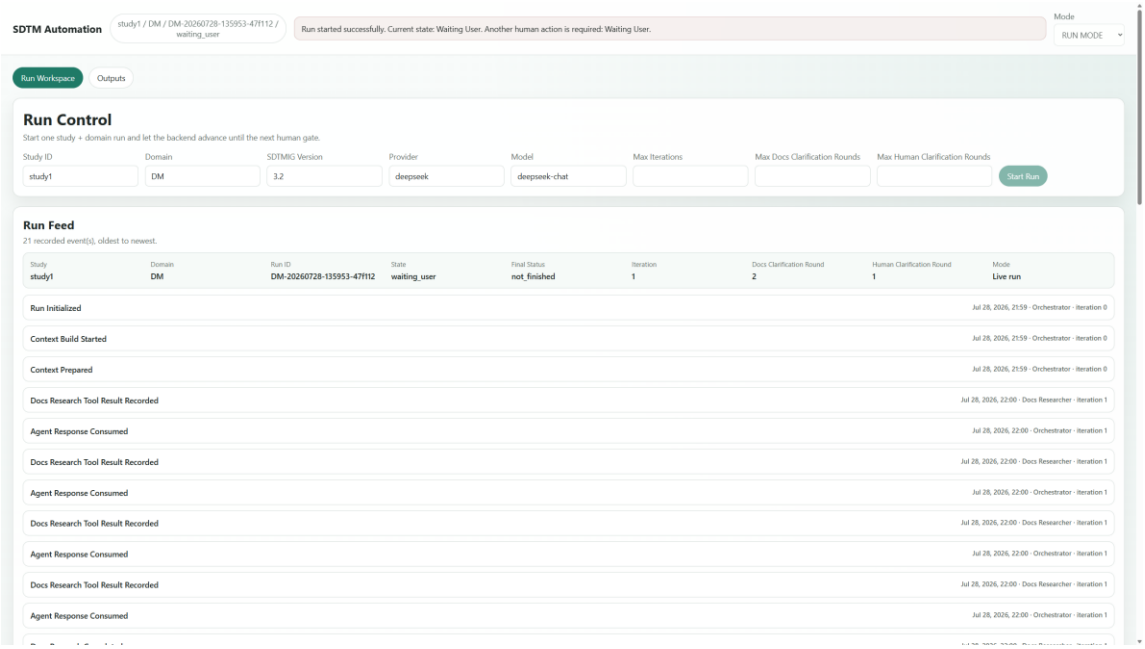

Figure 3. Screenshot of the web UI

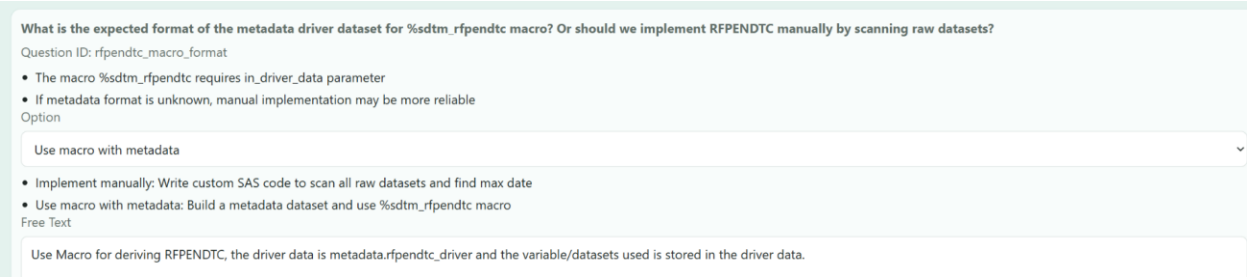

Figure 3. An example of request for user clarification

FUTURE OUTLOOK

While the current system already supports an end-to-end orchestrated workflow, several extensions would further strengthen its production readiness.

First, building a template code library would significantly accelerate domain implementation by providing reusable code blocks, improve consistency and reduce repeated prompting. Second, direct integration

with Pinnacle 21 or comparable compliance tools would extend the system from code generation governance into formal standards validation reporting. Third, the current UI could be expanded with run history, cross-run comparison, and richer artifact diffing. Fourth, given the sensitive nature of clinical data, generating synthetic data for validation purposes can help mitigate regulatory and compliance risks.

Together, these enhancements would move the system from a strong prototype with operational controls toward a production-grade SDTM automation platform.

CONCLUSION

This project has established an AI-assisted SDTM automation system that combines multi-agent collaboration with deterministic orchestration, structured artifacts, and explicit human governance. Several features are particularly important.

From a technical perspective, the system now supports:

- Excel-to-JSON specification conversion
- structured skill and reference loading
- state-aware context construction
- completeness checking and clarification routing
- structured implementation plan generation
- SAS code generation with Todo outputs
- deterministic validation reporting
- reviewer report generation
- replayable manifests and audit logs
- a web UI for run initiation, monitoring, intervention, and artifact inspection

From a governance perspective, the system introduces a practical human-in-the-loop model for SDTM programming. Rather than hiding uncertainty inside model output, it surfaces missing information and key routing decisions at explicit checkpoints. This makes the workflow more reviewable, more explainable, and better suited to regulated clinical programming work.

Overall, this project demonstrates that SDTM automation benefits not only from better code generation, but also from better orchestration, evidence management, and human decision design. As the runtime, validation, and retrieval capabilities continue to mature, this architecture provides a strong foundation for scalable, traceable, and standards-aware clinical data programming.

REFERENCES

1. Jinting Luo. 2025. “AI Application Attempt - Recognition and Conversion of Spec Files to SAS.” Proceedings of the PharmaSUG China 2025 Conference, Shanghai, CN: PharmaSUG China.
2. Teninq. 2025. cdisc-mcp. GitHub. https://github.com/Teninq/cdisc-mcp
3. Yanfen Zhu. 2025. “Automating SDTM Programming with Generative AI: A GPT-Based Framework for Clinical Data Transformation.” Proceedings of the PharmaSUG China 2025 Conference, Shanghai, CN: PharmaSUG China.

CONTACT INFORMATION

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

Wenqiang Chu

chuwq0810@163.com