PharmaSUG China 2026 - Paper AA227

AI-Native Real-Time SDTM Mapping and Exploration Agent

Hao Chen, Xuejie Zhou, Haomin Yang, and Qianwang Wang

ABSTRACT

This paper introduces SATA, the SDTM Automated Transformation Agent, an AI-native, real-time solution for SDTM mapping, transformation, and exploration. SATA is designed as an integrated all-in-one web application that enables data ingestion, transformation, validation, and exploration across diverse inputs, including raw clinical datasets, annotated CRFs, and metadata repositories. The platform provides real-time analysis, traceability, and on-demand partial reruns, allowing users to dynamically refine mappings and immediately assess results. This reduces feedback cycles and improves development efficiency.

The system combines intelligent SDTM specification prediction, automated SAS/R code generation, sandbox-based execution, iterative self-testing, and human-in-the-loop validation. Designed as a “digital employee,” SATA integrates domain knowledge, clinical metadata, and generative AI to enable end-to-end automation from EDC data to SDTM. Beyond transformation, SATA also supports SDTM-based dashboards, TFL generation, and ad hoc question answering, accelerating data-driven decision-making processes.

Initial implementation has demonstrated substantial efficiency gains, reducing SDTM mapping timelines from days to hours or even minutes while improving consistency, scalability, and transparency.

This paper presents SATA’s system design, architecture, development progress, use cases, and future directions, including AI-native mapping and continuous model optimization. SATA establishes a scalable foundation for next-generation clinical data analysis automation.

INTRODUCTION

The landscape of clinical development is currently experiencing a profound data complexity explosion. Modern clinical trials are characterized by a rapid growth in data volume, velocity, and diversity, driven by the integration of novel biomarkers, wearable devices, decentralized trial designs, and real-world data sources. While this wealth of information holds the key to accelerating precision medicine, it simultaneously creates a severe operational bottleneck in data management and statistical programming pipelines.

At the core of this bottleneck is the mandatory requirement to standardize clinical trial data for global regulatory submissions, specifically adhering to the Clinical Data Interchange Standards Consortium (CDISC) standard. Bridging the gap between dynamic, real-world clinical data—which is filled with human factors, exceptions, and variability—and the strictly controlled, standardized formats required by regulatory agencies is exceptionally challenging.

Historically, traditional SDTM transformation has been a profoundly labor-intensive, slow, and error-prone process. Conventional business intelligence (BI) and rule-based automation systems have proven insufficient for modern needs, often resulting in an inherent "decision lag" that prevents real-time, adaptive decision-making. As data scales exponentially, relying purely on human programmers to manually interpret metadata, draft mapping specifications, and write transformation code is no longer sustainable.

We have reached a strategic inflection point where Artificial Intelligence (AI) is the only viable path to simultaneously improve efficiency, quality, and speed at scale. This paper presents an innovative framework that harnesses Generative AI (GenAI) and open-source technologies to fully automate the SDTM conversion process. By shifting from manual, fragmented programming to an AI-native, end-to-end orchestrated workflow, this approach not only drastically reduces turnaround times but also enhances data quality, establishing a single source of truth for downstream clinical insights.

THE CDISC MAPPING CHALLENGE

Before examining how automation has evolved, it is worth being precise about what makes SDTM mapping hard. The difficulty is not a single problem but a set of compounding ones, each of which defeats naive automation in a different way.

- **Structural reshaping —** many source datasets must be deconstructed and reassembled into CDISC domains plus SUPPQUAL and CO.
- **Cardinality complexity —** 1:1, 1:n, n:1, transpose, and value-change transformations all occur between EDC and SDTM representations.
- **Annotation fragility —** CRF page-position shifts cause inaccurate mapping when annotations are treated as positional rather than semantic.
- **Loss of linkage on transpose —** complex transposes break data relationships between fields and records unless linkage is explicitly preserved.
- **Constant standard drift —** the CRF library grows with every version, so any rule set encoded once begins decaying immediately.
- **Broad domain coverage —** CDISC domains span Trial Design, Special Purpose, Interventions, Events, Findings, and Relationships, each with its own conventions.
- **Heavy derivation load —** CRF copy, assigned, derived, and protocol all contribute materially to the total mapping effort.
- **Study-specific derivations —** derived variables change from study to study, requiring flexible and reusable logic rather than fixed rules.
- **Weak variable-role context —** ID, Topic, Qualifier, and Timing roles must be preserved for the mapping to be semantically correct.
- **Process pain points —** accuracy, protocol reading, data quality, and long-term maintenance apply across the full pipeline rather than at any single step.

Taken together, these dimensions explain why rule-based automation plateaus: rules can encode the stable parts of the problem, but the study-specific, contextual, and drifting parts are precisely where the effort concentrates.

THE EVOLUTION OF SDTM MAPPING

To fully appreciate the impact of a Generative AI-driven architecture, it is essential to understand the historical progression of clinical data transformation. The evolution of SDTM mapping can be categorized into five distinct stages, each representing a leap in automation.

STAGE 1: MANUAL PROGRAMMER-CENTRIC

In this foundational stage, workflows consisted of study-specific programs with highly limited reusability, leading to exceptionally long cycle times. Each clinical study demanded its own bespoke set of transformation programs, with little opportunity for code sharing or standardization across projects.

STAGE 2: METADATA-DRIVEN

Organizations moved toward central repositories to store mapping rules, allowing specifications to be auto-generated or copied. This represented a significant improvement in reusability, though the underlying logic still required substantial human oversight and manual refinement.

STAGE 3: AI/ML-ASSISTED MAPPING (NLP, CLASSIFICATION)

Traditional Machine Learning models were deployed to automatically suggest potential mappings, though extensive manual effort remained. Natural Language Processing (NLP) and classification algorithms helped surface candidate mappings, but the final decision-making rested firmly with human programmers.

STAGE 4: GENERATIVE-AI COPILOTS

The advent of Large Language Models (LLMs) introduced the "Copilot Mode." Here, AI acts as an assistant to draft code and explain standards. The operational model is akin to "Mentoring an Intern" (Augmentation): the human programmer provides hand-in-hand guidance to save individual effort. While a meaningful step forward, this mode still fundamentally depends on the programmer's active participation throughout the process.

STAGE 5: END-TO-END (E2E) GENERATIVE AI ORCHESTRATION

The current frontier—and the focal point of this paper—is the deployment of fully orchestrated GenAI workflows. This paradigm transcends task-level assistance by autonomously managing the entire data lifecycle, from raw data ingestion to the generation of final SDTM datasets. Operating under a framework of strict human governance and "human-in-the-loop" validation, Stage 5 represents a definitive shift from mere human augmentation to true, scalable, and intelligent automation.

THE POWER OF OPEN-SOURCE: R AND THE PHARMAVERSE ECOSYSTEM

The foundation of our automated framework rests heavily on the power, flexibility, and transparency of open-source software, specifically leveraging the R programming language. R has evolved into a mature, production-grade platform for statistical computing, data analysis, and regulatory reporting, supported by an expansive ecosystem (such as tidyverse and ggplot2) that sees broad adoption in the pharma industry.

This large, open-source package ecosystem enables rapid innovation, extensive reuse, and seamless integration with emerging AI technologies. A prime example is the Pharmaverse, a curated network of R packages for clinical reporting workflows. Within our architecture, we heavily utilize sdtm.oak, an EDC and Data Standard agnostic transformation engine that automates the conversion of raw clinical data into structured SDTM datasets using standardized mapping algorithms.

To solve the regulatory requirement for human oversight, our framework employs R Shiny to build the user interface. Shiny provides a robust framework for developing interactive analytics applications and dashboards, supporting critical Human-in-the-Loop (AI + HI) workflows. This allows subject matter experts to interact with the data in near real-time via web-deployed R logic.

In addition to R, the framework leverages Python for orchestration, AI model integration, and infrastructure-level tasks, along with Databricks for cloud-scale compute, advanced data analysis, and the real-time data lakehouse architecture that underpins the system's storage and retrieval layers.

STANDARDS STILL MATTER IN THE AI ERA

A common assumption is that sufficiently capable models make standards and curated packages redundant. Evidence from the pharmaverse community points the other way. In a published benchmark of ADaM generation across ADSL, ADAE, ADVS, and ADLB, an agent equipped with a curated, domain-aware skill achieved 88–100% pass rates across domains, while the same agent without that skill achieved only 17–59% pass rates, with high variance.

The variance matters as much as the mean. Inconsistent output is not a defensible process in a GxP context: a skill-guided agent produces consistent, traceable, standards-anchored code, whereas an unguided agent produces something different every time. The skill does not replace the validated package; it connects the model to it, supplying which functions to use for which derivations, how to structure a program for QC readability, which variables require special handling, and what assertions to include.

The same principle governs our SDTM work. CDISC standards, controlled terminology, and the sdtm.oak transformation engine are what convert a general-purpose model into a reliable clinical data component. Standards are not an obstacle that AI routes around — they are the substrate that makes AI output auditable.

SATA: THE SDTM AUTOMATED TRANSFORMATION AGENT

To operationalize the convergence of GenAI and open-source computing, We have developed Sata (SDTM Automated Transformation Agent). Sata is an intelligent virtual agent and all-in-one integrated web application designed to autonomously transform raw clinical data into fully compliant SDTM datasets.

Sata eliminates the need for manual mapping specifications; instead, it can ingest metadata directly from raw data sources, Architect Loader Spreadsheet (ALS) files, BlankCRFs, or data descriptions. Operating through three core engines—AI Prediction, a Coding Agent, and Online Review & Update—Sata utilizes standard CDISC rules to generate structured mappings.

The Sata framework is composed of five interconnected modules:

MODULE 1: DATA INGESTION AND PROFILING ENGINE

This module securely connects to EDC systems or raw data repositories. It ingests raw datasets and automatically generates metadata profiles (e.g., data types, missingness, value distributions). This profiling provides the necessary context for the AI to understand the structural realities of the incoming data.

MODULE 2: AI METADATA INTERPRETER & MAPPER

Acting as the "brain" of the system, this module feeds the raw metadata and study protocols into the LLM. The AI evaluates the source data against CDISC SDTM standards and generates a comprehensive mapping specification. It identifies the target domain (e.g., AE, DM, VS) and maps source variables to target variables, applying necessary derivations or terminology conversions.

MODULE 3: DYNAMIC CODE GENERATOR

Once the mapping specification is approved (or auto-validated based on confidence scores), Sata's code generator translates the logic into executable Python or R scripts. This module uses prompt engineering techniques to ensure the generated code adheres to industry best practices and internal coding standards.

MODULE 4: EXECUTION AND TRANSFORMATION PIPELINE

Hosted on Databricks, this pipeline executes the AI-generated code against the raw data. It performs the physical data transformation, handling complex joins, transpositions, and formatting to produce the final SDTM datasets.

MODULE 5: VALIDATION AND CONTINUOUS LEARNING LOOP (HUMAN-IN-THE-LOOP)

Crucial for regulatory compliance, Sata includes an automated validation suite (similar to Pinnacle 21) that checks the output against CDISC rules. Any discrepancies or low-confidence AI mappings are flagged for human review. When a human programmer corrects a mapping or modifies the code, Sata captures this feedback, updating its vector databases and few-shot prompting examples to improve future accuracy.

SYSTEM ARCHITECTURE

The automated SDTM framework is built upon the Databricks Lakehouse Platform Foundation, utilizing an "AI Acceleration Turbine" to process data. This robust infrastructure is comprised of several interconnected layers:

- **Data & Governance Layer:** Utilizes Unity Catalog to manage data pipelines securely.
- **Compute & Scheduling Layer:** Provides serverless compute and job scheduling.
- **Model & Validator Layer:** Houses the core AI models and business rules.
- **AI Knowledge Base Layer:** Leverages Vector Search and a comprehensive CDISC Library.

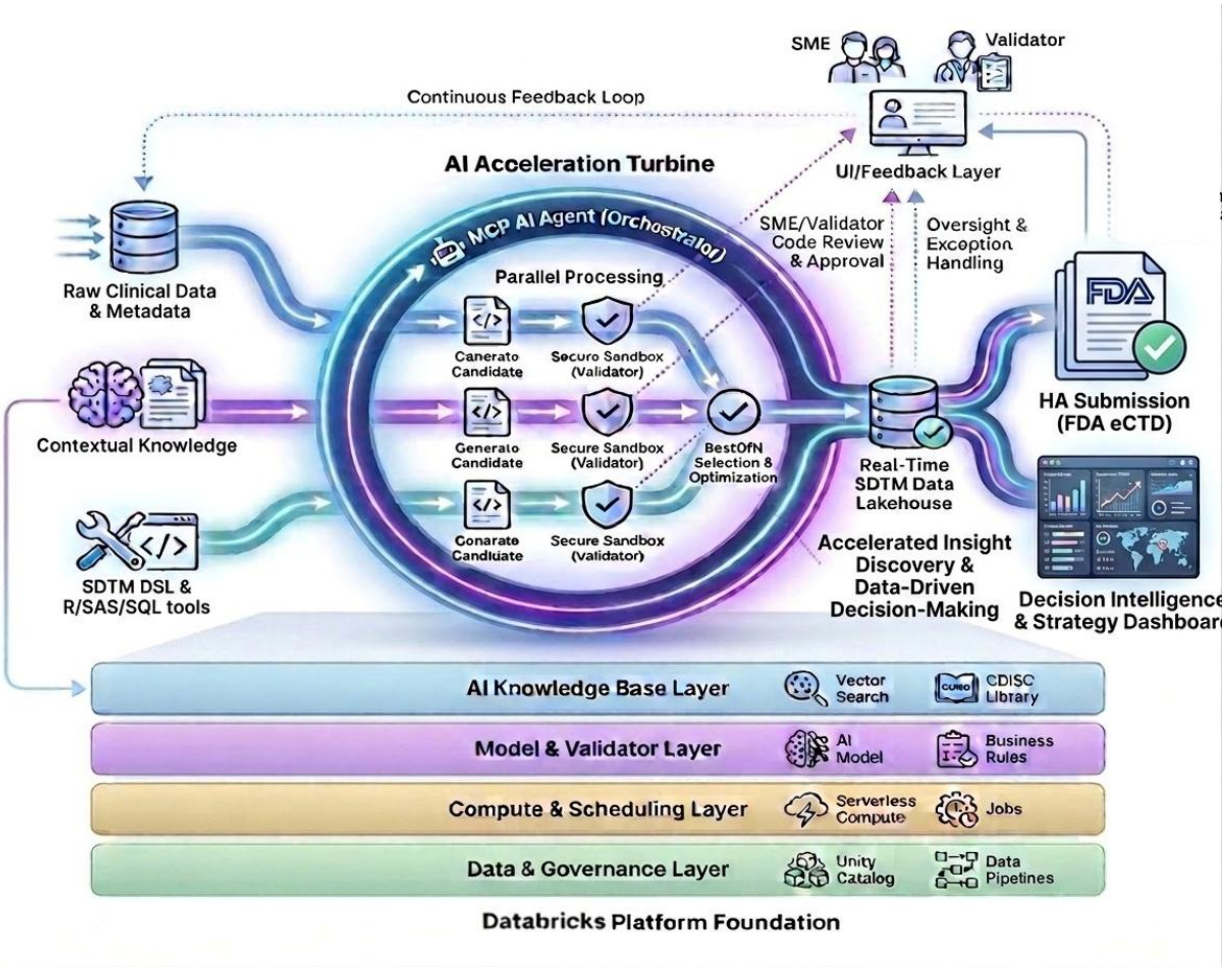

Figure 1. System Architecture

When raw clinical data streams are ingested in real-time, they are enriched with contextual knowledge (via RAG and Vector Search) and processed using the SDTM DSL together with R, SAS, and SQL tooling. An MCP AI Orchestrator Agent drives the data into a Parallel Processing loop where multiple candidate mappings are generated. Each candidate is rigorously tested within a Secure Sandbox (Validator), and a "BestOfN Selection & Optimization" algorithm chooses the superior logic. The output is then pushed to a Real-Time SDTM Data Lakehouse, which feeds three downstream paths: health authority submission via FDA eCTD, decision intelligence and strategy dashboards, and accelerated insight discovery supporting data-driven decision-making.

KEY FEATURES AND CORE BENEFITS

The Sata framework is delivered as an all-in-one integrated web application, providing a unified interface for data upload, mapping review, code inspection, execution monitoring, and output validation. This design eliminates the need for users to navigate between multiple tools and environments, reducing friction and cognitive overhead.

The efficiency gains are dramatic. The time required to produce initial SDTM datasets can be reduced from hundreds of programmer-hours to merely a few hours during the automated prediction and code generation phases. Furthermore, by orchestrating the entire end-to-end process—from raw data upload to final SDTM output, compliance checking, and iterative refinement, the system can deliver complete, finalized results in just a few days, compared to the weeks or months demanded by conventional approaches.

The framework supports broad data inputs, accommodating both corporate-standard and industry-standard raw data formats. This flexibility is essential in an industry where different EDC platforms, CROs, and partner organizations produce data in varying formats and conventions.

The core benefits extend across the entire clinical data value chain. Faster SDTM delivery and AI-augmented cloud-scale compute reduce the time from data collection to analysis-ready datasets. Less manual work and fewer errors translate to higher data quality and lower rework costs. Earlier availability of standardized data enables better and faster decisions about trial conduct, safety signals, and efficacy trends. Accelerated downstream workflows—including analysis dataset creation, statistical analysis, and regulatory submission preparation—contribute directly to faster time-to-market for new therapies. And by establishing a single source of truth for SDTM transformation logic, the system enhances consistency, reproducibility, and audit readiness across the organization.

THE END-TO-END MULTI-AGENT WORKFLOW

Sata avoids relying on a single monolithic model by instead orchestrating a highly specialized, multi-agent end-to-end workflow. This workflow autonomously processes data from raw upload through AI mapping prediction, code generation, execution, debugging, SDTM checking, and final output generation.

PHASE 1: INFORMATION EXTRACTION

The Parser Agent ingests ALS files to create JSON inventories, while the Librarian Agent retrieves CDISC standards and the Rules Curator formats company-specific mapping rules. Together, these agents establish the complete contextual foundation required for downstream mapping.

PHASE 2: PREDICTIVE MAPPING

The Domain Assignment Agent and Mapping Agent propose logical connections between raw data fields and SDTM target variables. These are iteratively verified by a Critic Agent, culminating in human validation to produce a final Mapping Result JSON. The iterative critic loop ensures that mapping proposals are internally consistent before they reach human reviewers.

PHASE 3: EXECUTION & TESTING

The Code Generator translates the mappings into executable R code, leveraging the sdtm.oak framework. A Code Parser analyzes the oak codebase to allow the Test Case Generator to build automated tests. The R code is run by an Executor in Sandbox and monitored by a Code Tester for auto-debugging.

PHASE 4: FINALIZATION

An SDTM Executor produces the final structured SDTM tables, subject to a final round of human validation. The output datasets are accompanied by full provenance metadata, including the mapping JSON, generated code, test results, and validation logs.

Phases 3 and 4 are where the framework stops behaving like a copilot. Extraction and predictive mapping are valuable, but they remain recommendation engines: the agent proposes, the programmer disposes, and the programmer still writes, runs, and debugs the code. The leap to end-to-end automation happens when generative AI takes on execution.

In Phase 3 the Code Generator writes the R code, the Test Case Generator builds its own tests against the sdtm.oak codebase, the Executor runs the result in a sandbox, and the Code Tester reads the failures and repairs them without a human intervening. In Phase 4 the SDTM Executor produces the final tables together with the provenance needed to defend them. That closed generate-execute-test-repair loop is the difference between an assistant that drafts and a system that delivers. It is also where the largest share of the cycle-time reduction reported in this paper originates, because writing and debugging transformation code — not proposing mappings — is what consumed most of the programmer-hours in the manual process.

The following illustrates the Mapping Prediction and Code Executor agents:

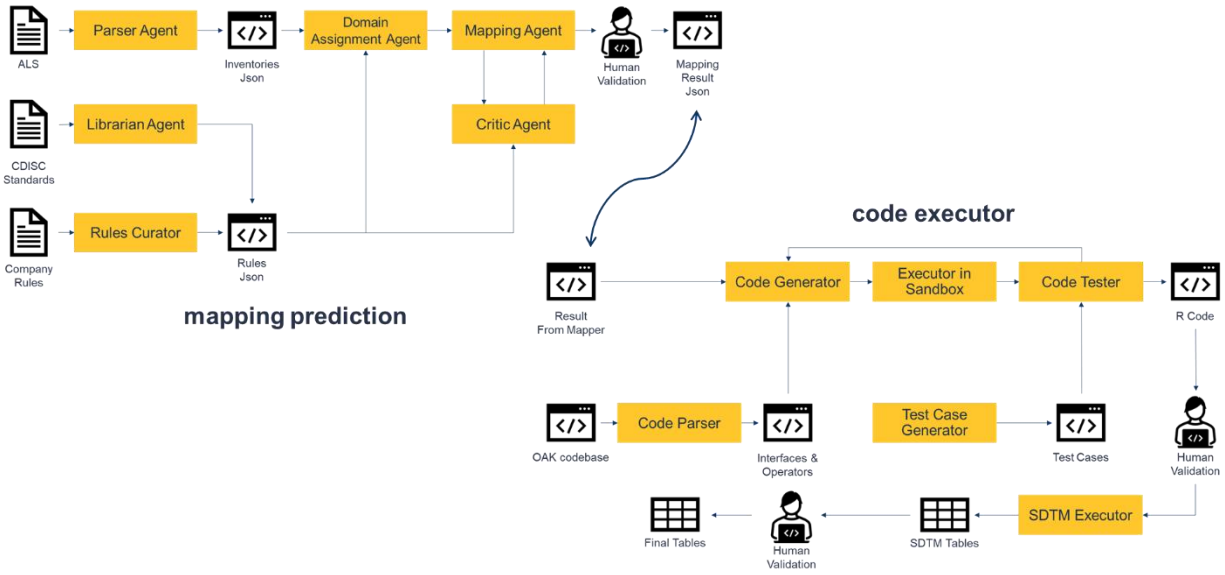

Figure 2. Mapping Prediction and Code Executor agents

PREDICTION METHODS AND PERFORMANCE METRICS

To balance computational efficiency with regulatory accuracy, the Sata predictive engine cascades through four distinct prediction methodologies.

PREDICTION METHODOLOGY CASCADE

- **Method 1 – Exact Match (100% Confidence):** Direct alignment of labels and definitions requiring no deep thinking, offering "Good" traceability. This method is applied first and resolves straightforward, unambiguous field mappings.
- **Method 2 – Fuzzy Match (>95% Confidence):** High-confidence similarity algorithms that resolve minor discrepancies without deep reasoning. Fuzzy matching handles variations in naming conventions, abbreviations, and minor label differences across studies.
- **Method 3 – Fuzzy + Reasoning (70–95% Confidence):** Generates batches of candidates with AI-generated rationale and deterministic workflow checks. This method is invoked when simple string matching is insufficient and contextual understanding of the variable's purpose is required.
- **Method 4 – Deep Reasoning (Search):** Concept-level inference that requires understanding the CDISC Implementation Guide and historical mappings. This method naturally requires post-processing and strict workflow controls, and is reserved for the most complex or novel mapping scenarios.

Methods 1 and 2 are not generative. Exact and fuzzy matching are deterministic string techniques that predate large language models, and they resolve the straightforward fields quickly and cheaply, which is exactly why they run first. But they resolve only the easy cases.

Methods 3 and 4 are where generative AI earns its place. Fuzzy matching with reasoning handles fields whose labels are ambiguous and whose purpose has to be inferred from surrounding context. Deep reasoning handles concept-level inference that requires reading the CDISC Implementation Guide and reconciling a field against historical mappings from comparable studies. These are precisely the cases a rule-based engine cannot reach at any useful confidence, and the cases where a programmer historically had to stop and think. Automating them is what converts the tool from a copilot that accelerates the easy majority into an end-to-end agent that can carry a domain through unattended. It is equally why the residual error concentrates in these two methods, and why they carry the strictest post-processing and workflow controls of the four.

Accuracy Progression Over Time

Because the system learns continuously from reviewer corrections, accuracy has been tracked longitudinally against a fixed evaluation set. Table 1 shows the progression from the pre-AI baseline method through three subsequent measurement cycles.

Metric	Baseline	March	May	July	Change
Field Accuracy (Exact)	86.89	90.35	92.18	92.65	+5.76 pp
Annotation Rule Accuracy	92.39	94.09	95.09	96.02	+3.63 pp
Variable (Name) Accuracy	—	96.62	97.64	98.43	+1.81 pp*

Table 1. Overall Accuracy Progression (%)

*Variable (Name) Accuracy is a newer metric with no baseline-method equivalent; its change is measured against the March cycle.

Two observations are worth drawing out. First, the gains are monotonic but decelerating — field accuracy improved by 3.46 points in the first cycle and 0.47 points in the most recent one, which is the expected shape as the remaining errors shift from systematic to genuinely ambiguous cases. Second, annotation rule accuracy consistently exceeds field accuracy, confirming that the harder problem is not identifying the applicable rule but resolving the field-level detail underneath it.

Output Data Comparison

Prediction accuracy measures the mapping specification. A separate and stricter question is whether the resulting datasets match the reference output. Across six studies, variable-level error rates ranged from 0.5% to 4.4%, with a weighted average of 2.9%, while record-grouping error rates ranged from 0.4% to 1.5%, with a weighted average of 0.8%.

Weighted averages are computed as the sum of numerators over the sum of denominators rather than as an unweighted mean of study-level rates. Record-grouping error is consistently the smaller of the two at 0.8% against 2.9% for variables, indicating that the system reliably determines which records belong together even where individual variable derivations still need review.

Benchmarking Against Rule-Based Automation

Compared against the rule-based automation, the AI-native approach halves the error rate — 6% against 12% — while producing the mapping specification substantially faster. Spec generation takes approximately 15 minutes against 55 minutes for the rule-based path, a reduction of 73%, or roughly 40 minutes saved per specification. The two effects compound: faster specification generation means more review cycles fit into the same calendar time, and each cycle starts from a more accurate draft.

TRANSLATING THE PRODUCTION SAS ENGINE INTO R

Most organizations adopting an R-based SDTM pipeline face a prerequisite problem: the authoritative production engine is written in SAS. In our case this was not a single macro but a top-level SDTM macro supported by an entire family of sub-macros — deeply nested, sharing state, and dependent on call order — driven by metadata through a specification, a generator, domain code, and finally data.

The objective was not a literal port. It was a system-level rebuild that produces an R code generator emitting the same SDTM data: modular, organized by domain, readable by both humans and AI agents, reproducible and cloud-native under Databricks and renv, and required to match the SAS output cell for cell at variable level. The SAS engine remained the gold standard against which the R engine was judged.

Validation compared R engine output against the production SAS engine across seven studies, scoring variable-level exact match. Three of the seven studies matched exactly at 100.00%, and the remaining four ranged from 99.52% to 99.76%, giving an overall perfect rate above 99%.

On inspection, the remaining differences are mostly reference-side UTF-8 artifacts rather than defects in the R engine, which means the effective agreement is higher than the raw figures suggest. This result is what makes the R track credible as a production replacement rather than a parallel experiment.

HYBRID INFERENCE STRATEGY: LOCAL 14B WITH PROMPT FALLBACK

Running every prediction against a frontier model is accurate but expensive, and in some deployment contexts external API calls are constrained on data-governance grounds. We therefore evaluated a hybrid strategy: a locally hosted 14B model handles predictions where its own confidence is high, and only the remainder falls back to the external model.

At the recommended confidence threshold of 0.999, 63.4% of samples — 5,746 of 9,070 — are resolved fully locally, requiring no external model call. External API token consumption falls by 63%, since only 36.6% of samples reach the frontier model. On that confident subset the 14B model reaches 89.58% field accuracy, well above its 76.96% accuracy across all samples, confirming that the confidence signal is genuinely selective.

The cost is modest and measurable. Field accuracy declines from 91.53% under the prompt-only configuration to 89.58%, a loss of 1.95 percentage points, which retains roughly 98% of the quality while eliminating nearly two-thirds of external inference. For workloads where inference latency, token cost, or data residency dominate, this is a favorable trade; for final submission-grade runs, the prompt-only configuration remains available.

HUMAN-IN-THE-LOOP (AI + HI) GOVERNANCE

The deployment of AI in the highly regulated biopharmaceutical industry demands absolute traceability. Therefore, Sata is wrapped in a strict Human-in-the-Loop (HITL) governance model.

The system is designed to purposefully surface uncertainty and isolate potential errors. Through the R Shiny UI, programming subject matter experts (SMEs) can conduct rapid expert reviews. The interface highlights low-confidence mappings, flags inconsistencies with historical patterns, and provides side-by-side comparisons of AI-proposed versus reference mappings.

Crucially, this interaction creates a continuous feedback loop. Every SME correction is fed back into the AI Knowledge Base, allowing the models to continuously improve and adapt to specific data nuances over time, while strictly maintaining regulatory compliance. This learning mechanism ensures that the system becomes increasingly accurate with each study processed, embodying a virtuous cycle of human expertise and machine intelligence.

DELIVERY TRACKS AND SERVICE MODEL

The capability is delivered through two tracks, reflecting the reality that organizations migrate at different speeds and that a SAS-based pipeline may remain authoritative for some time.

- **Track 1 — SAS Platform:** the solution built on the existing SAS foundation, for teams whose validated production pipeline is SAS-based.
- **Track 2 — End-to-End SDTM R Solution:** the AI-native R pipeline, available in two access patterns. The first pairs a user interface with the AI agent, offering both an automatic mode and a step-by-step mode in which the reviewer advances the pipeline stage by stage. The second is agent-only access through chat, a callable skill, or a programmatic API.

The service itself spans both data deliverables and platform access. On the deliverables side, the outputs are the SDTM data, an SDTM data analysis report, the SDTM mapping specification, and the SDTM mapping code — the artifacts a downstream team would otherwise produce by hand. On the platform side, access is offered through the SDTM mapping platform, a mapping API, and a mapping chatbot. A fourth mode is worth noting separately: the mapping itself can serve as context for other AI solutions, since a system that has resolved raw data into SDTM has necessarily built a rich semantic model of that raw data which downstream agents can reuse.

DEMO

Both delivery tracks were demonstrated live. This section captures the same walkthrough in static form. Every screenshot below uses dummy data; no study data appears in any figure.

TRACK 1: THE SAS PLATFORM

Track 1 runs on the existing SAS foundation. The demonstration opens in the SDTM Mapper, where an annotated CRF page — here the Substance Use form — is shown alongside the connections drawn from each collected field to its SDTM target, with the form-level panel resolving the SU domain and its qualifiers. An AI Copilot is available from the same toolbar. The mapping specification view presents the same logic in spreadsheet form, one row per source field with SDTMIG version, CRF identifier, source dataset, form, and target variable. The SDTM Execution console then runs the generated code domain by domain, reporting status, executor, and last run time, with per-domain links to the code, the execution log, and the resulting dataset.

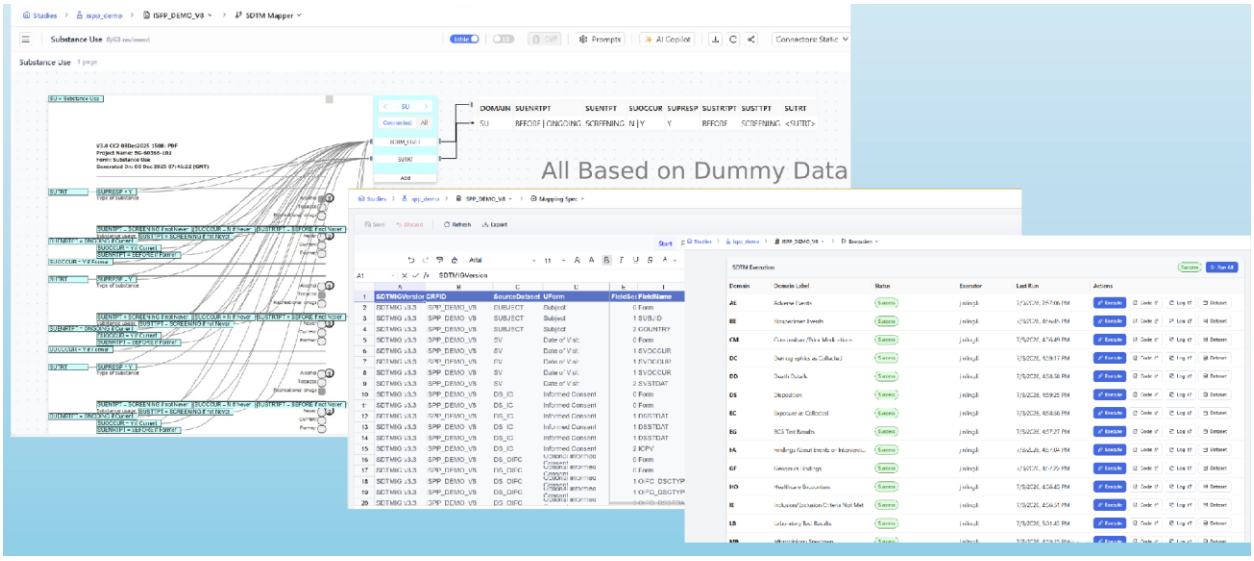

Figure 3. Track 1 — SAS Platform Demonstration

TRACK 2A: UI WITH AI AGENT

Track 2a is the AI-native R pipeline driven through the user interface with the agent in support. Specification generation runs as a visible sequence of steps — loading the ALS, building the library, multi-level LLM matching across forms and fields, parsing, post-processing, and saving — so the reviewer can see where the prediction is and what it has consumed. The generated specification is reviewed in the SDTM shell view, which shows the CRF-to-domain linkage for each form alongside the proposed variables. Execution produces a summary panel reporting how many domains completed, the record counts behind them, and per-domain status with direct links to code and logs.

Downstream, the same application presents study-level dashboards and a TFL panel that switches between safety and efficacy outputs, including swimmer, waterfall, and Kaplan-Meier displays. A chat pane sits alongside the outputs. When the reviewer asks for a change, the agent proposes it as a code diff against the underlying R script rather than as an opaque edit, which keeps the human review step meaningful even when the change originates with the model.

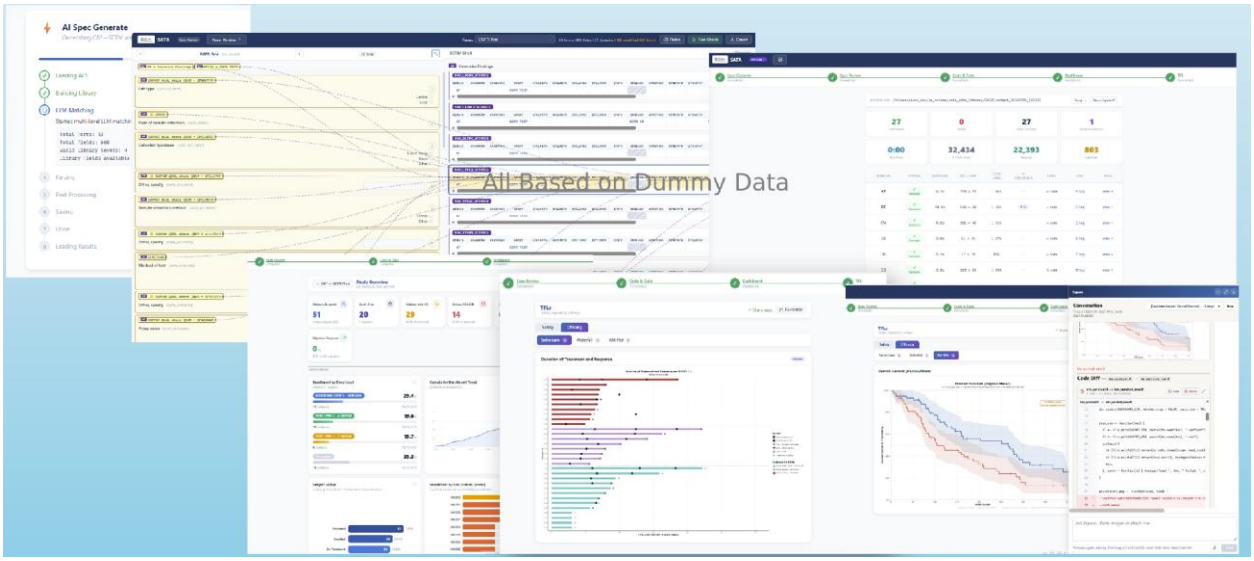

Figure 4. Track 2a — End-to-End SDTM Mapping and R Analysis with UI and AI Agent

TRACK 2B: AI AGENT ONLY

Track 2b removes the interface entirely. The same work is requested conversationally: the user supplies the study paths — ALS, annotated CRF, and raw data — and asks the agent to produce SDTM. The agent reports which domains it has produced, then proposes a catalog of tables, figures, and listings organized by domain, covering demography, disposition, adverse events, laboratory results, vital signs, ECG, and concomitant medications, for the user to select from.

Subsequent turns generate the requested outputs directly — a lattice plot of laboratory tests over time, an HTML dashboard, and a TEAE overview table — with the agent stating what it read, what it derived, and which files it wrote. This mode is what makes the chat, skill, and API access patterns described earlier possible: the same pipeline, addressed programmatically rather than through a screen.

Figure 5. Track 2b — End-to-End SDTM Mapping and Analysis, AI Agent Only

AI STRATEGY AND PROGRAMMING EXPERTISE

The technical results above rest on a small number of strategic commitments that we believe generalize beyond SDTM conversion.

- **Accuracy matters; metrics are the goal.** Ignore precision and everything looks achievable with AI. Without production-grade accuracy, however, proofs of concept do not become production systems and chatbots do not become agents. Measurement is what separates the two.
- **Technique hierarchy.** In-context learning, then ontology, then vector search. Each step down that ladder can cost one to several orders of magnitude of accuracy, so the ordering should be deliberate rather than incidental.
- **Flywheel.** Use AI to standardize the data and the process; then use those standards to make AI more accurate, stable, and reliable. The loop is self-reinforcing, which is why standards investment compounds.
- **Engineering discipline.** Avoid over-reliance on complex LLM frameworks. Writing custom code for fine-grained control of the model has proven more maintainable than accepting framework abstractions that obscure what the model actually sees.

These commitments map onto four layers of engineering practice, summarized in Table 2.

Concept	Definition
Prompt Engineering	Telling the model what to do
Context Engineering	Determining what the model can see
Loop Engineering	Designing how the agent iterates toward goals
Harness Engineering	Ensuring agents operate reliably, safely, and at scale

Table 2. Layers of AI Engineering Practice

Finally, the results depend on statistical programming expertise rather than substituting for it. On the business side, statistical programmers understand the documents, data, and code spanning protocol through submission; they are detail-oriented across data, code, and specification; and they know the industry standards the output is judged against. On the technical side, the work required writing custom code for fine-grained LLM control, SAS-to-R automated translation, model fine-tuning, web application development, and platform engineering on Databricks. Neither skill set alone would have produced the system described here.

CONCLUSION

The SDTM Automated Transformation Agent represents a meaningful advance in the application of Generative AI to clinical data management. By combining LLM-powered mapping prediction, automated code generation in R, multi-agent orchestration, and human-in-the-loop governance, the framework addresses the fundamental limitations of manual and rule-based approaches to SDTM conversion.

The results reported here strengthen that case in three specific ways. Accuracy has improved monotonically across measurement cycles, reaching 92.65% field accuracy and 96.02% annotation rule accuracy, while output-level comparison shows weighted error rates of 2.9% for variables and 0.8% for record grouping. The production SAS engine has been rebuilt as an R system that matches it at variable level with an overall perfect rate above 99% across seven studies, removing the principal obstacle to an open-source pipeline. And a hybrid inference configuration retains roughly 98% of prediction quality while eliminating nearly two-thirds of external model calls, which makes the economics and the data-governance posture of large-scale deployment considerably more favorable.

Two design choices carry most of that result. The first is the upper half of the prediction cascade: exact and fuzzy matching dispose of the straightforward fields cheaply, but it is fuzzy-plus-reasoning and deep reasoning — the generative methods — that resolve the ambiguous and concept-level mappings a rule-based engine cannot reach. The second is the execution half of the agent workflow: generating the code, building tests for it, running it in a sandbox, and repairing it automatically. Together these shift the system's center of gravity from suggestion to delivery, which is the substantive difference between a copilot and an end-to-end agent, and they are where continued engineering effort is best concentrated.

Generative AI is reshaping the biopharma industry, and clinical trials represent a prime transformation area where AI can deliver faster cycle times and smarter execution. The Sata framework demonstrates

that this promise can be realized practically, with measurable accuracy across diverse studies and a workflow design that respects the compliance and quality requirements of regulated environments.

The combination of a curated knowledge library with LLM reasoning capabilities enables an end-to-end AI system in which AI handles mapping suggestions, code drafting, reasoning, testing, and iteration, while standard and deterministic methods ensure quality control and traceability. The HITL workflow ensures that this automation augments rather than replaces human expertise, surfacing uncertainty, isolating errors, and enabling rapid expert review. As reviewer corrections flow back into the system, the learning loop closes—models improve continuously while compliance is maintained.

Looking ahead, the next focus is derivation. Core mapping has now passed the threshold we set for it — roughly 15 minutes to produce a mapping specification, generate the R code, and create the SDTM-minus data, at approximately 95% accuracy — which leaves derived variables as the remaining constraint on unattended end-to-end SDTM production, and they are where the next phase of development will concentrate. Beyond that, we anticipate that this architectural pattern — AI agents governed by human oversight, built on open-source foundations, and continuously refined through operational feedback — will become the standard approach for clinical data transformation across the industry. The open-source nature of the underlying technologies, including R, the pharmaverse ecosystem, and sdtm.oak, ensures that the innovations demonstrated here remain accessible to the broader clinical data science community and can be adapted, extended, and improved collaboratively.

REFERENCES

CDISC. SDTM Implementation Guide https://www.cdisc.org

POSIT. Easy web apps for data science without the compromises. https://shiny.posit.co

Databricks. Bring AI to your data to help you bring AI to the world. https://www.databricks.com

Pharmaverse. Why we still need {admiral} in an age of AI — pharmaverse blog. https://pharmaverse.github.io/blog/posts/2026-06-14_admiral_in_age_of_ai/admiral_in_age_of_ai.html

Chunpeng Zhao, Mijun Hu, Jie Liu, Jiapeng He, Liming Xie, Aide Zhou, and Bingting Tao. Real-time SDTM. PharmaSUG China 2022, Paper AT-105. https://www.lexjansen.com/pharmasug-cn/2022/AT/Pharmasug-China-2022-AT105.pdf

Hao Chen and Daniel Huang. Revolutionizing Clinical Data Transformation: Harnessing GenAI and Open Source for Automated SDTM Conversion. PHUSE US Connect 2026, Austin, TX, Paper ML29. https://phuse.s3.eu-central-1.amazonaws.com/Archive/2026/Connect/US/Austin/PAP_ML29.pdf

ACKNOWLEDGMENTS

The authors would like to thank the leadership team for their support of this work, and the SATA team for their contributions to the development of the framework.

CONTACT INFORMATION

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

Author Name: Hao Chen
Email: mathhao@gmail.com
Website: https://www.linkedin.com/in/hao-harry-chen-98709128