MWSUG 2025 - Paper AE-82

Prompt, Program, Submit: Generative AI for Faster SDTM, ADaM, and TLFs

Matt Becker, Pankaj Attri; SAS Institute Inc.

ABSTRACT

The life sciences industry is seeing more requests for quick, compliant clinical trial submissions, which makes it even more important to improve programming operations. Generative artificial intelligence (GenAI), especially large language models (LLMs), could change the way SDTM, ADaM, and TLFs are created in a big way.

This session will investigate the practical applications of GenAI to automate and enhance critical clinical programming duties. From the mapping of raw data to SDTM domains, to the crafting of ADaM specifications, and the generation of common code or statistical summaries, we will analyze real-world use cases. These examples will demonstrate how to reduce manual effort while ensuring traceability and compliance. Additionally, we will illustrate the integration of these AI-driven procedures into environments to improve productivity without compromising regulatory compliance.

Whether you are a statistical programmer, data manager, or biostatistician, this session will help you reimagine what is possible when human expertise meets machine intelligence.

INTRODUCTION

Clinical programming is really at a turning point. Clinical trials are getting more complicated, and regulations are becoming stricter, making the old ways of handling and analyzing clinical data just not cutting it anymore. Even though we've widely adopted CDISC standards and have some advanced statistical platforms, a lot of clinical programming is still done manually, which means it's repetitive and sucks up a ton of resources.

But here's the exciting part: we're in the middle of a tech revolution driven by GenAI and large language models (LLMs). These aren't just cool research projects anymore; they're powerful tools that can automate tasks and are ready to be integrated into our clinical development processes. In this paper, we're going to dive into how LLMs can speed up and improve the development of SDTM, ADaM, and TLFs. The goal isn't to replace programmers like us, but to make our expertise even more effective.

WHY AUTOMATION IS A GROWING NEED IN CLINICAL PROGRAMMING

Clinical programming is currently experiencing a significant transition. Clinical trials are getting way more complicated, and the rules we must follow are getting stricter. The old ways of handling and studying clinical data just aren't cutting it anymore. Even though most of us are using CDISC standards and the latest stats software, a lot of clinical programming is still done by hand, repeatedly, and it eats up a ton of resources.

Think about it: trials are getting bigger, they're happening all over the world, and we're pulling in data from everywhere – real-world studies, wearable devices, even medical images. All this means clinical programmers are getting buried under a mountain of work.

Let's take a typical Phase 3 study as an example:

- We're still manually mapping raw data to SDTM domains.
- We're building ADaM datasets using derivation rules that we often must write from scratch.
- We're repeatedly coding TLFs (tables, listings, figures) from standard templates.
- And documentation, like define.xml files and reviewer guides, are put together separately, often by different teams.

This whole process is just begging for things to go wrong. It takes too long, costs too much, and is full of inconsistencies that can delay submissions or raise red flags with regulators. Plus, with deadlines getting tighter and global trials becoming more common, the lack of experienced programmers only makes the problem worse.

Automation is no longer a futuristic concept – it is a necessity.

WHAT ARE GENAI AND LARGE LANGUAGE MODELS (LLMS), AND WHY DO THEY MATTER NOW?

GenAI is all about algorithms that can whip up new stuff – think text, code, summaries, and even data. They learn how to do this by studying huge amounts of training data. Large Language Models (LLMs), like GPT-4, are a type of GenAI. These models are trained on billions of words, which helps them understand and create language and logic that sounds like it's coming from a human.

LLMs are capable of:

- Interpreting natural language specifications
- Writing syntactically correct code (e.g., SAS, R, Python)
- Summarizing and translating documentation
- Identifying patterns across structured and unstructured data

Why do they matter now?

- Accessibility: LLMs are available through APIs and cloud platforms, or can be fine-tuned and deployed in secure, on-prem environments.
- Performance: These models can now write production-quality code with appropriate prompting and validation.
- Domain Adaptation: With proper fine-tuning or prompting strategies, LLMs can be tailored to life science use cases.

We're reaching a point where Large Language Models are finally ready to make a real difference in how we build clinical software. The technology has matured, and frankly, the need for better solutions is pressing.

GOALS OF THIS SESSION

The purpose of this session is not to theorize about AI in life sciences, but to demonstrate real-world use cases where LLMs are already helping automate core clinical programming tasks:

- SDTM Mapping: Converting raw data into standard domains using natural language specs.
- ADaM Generation: Automating derivation logic based on statistical analysis plans.
- TLF Programming: Writing boilerplate tables, listings, and figures based on mock shells.

These use cases show that automation is not only possible—it’s practical, measurable, and safe when implemented with the right oversight.

COMMON CHALLENGES IN CLINICAL PROGRAMMING WORKFLOWS

To appreciate where GenAI fits in, we must first examine where traditional workflows break down.

A. MANUAL SDTM AND ADAM MAPPING

Despite the use of standardized templates, creating SDTM and ADaM datasets is still largely a manual task. Programmers interpret Excel specifications, write code, and validate outputs with clinical teams. Reuse is limited,

and logic varies between programmers or studies. Errors in mapping often result in rework or inconsistencies that delay analysis.

B. REDUNDANT COMMON TLF CODE

TLFs, especially in large pivotal trials, follow familiar templates: demographic summaries, adverse event tables, efficacy plots. Yet, programmers often write similar code from scratch or copy/paste from prior studies, leading to duplication and variable coding styles. This reduces efficiency and increases QC overhead.

C. DOCUMENT-HEAVY OUTPUTS

Creating submission packages involves a lot of important documents, such as define.xml, reviewer guides, annotated CRFs, and derivation summaries. However, these documents are often managed separately from the programming pipeline. This means they need to be manually maintained, cross-referenced, and formatted. If there are any changes to the datasets, it can lead to revisions in these documents, which can be a significant waste of time.

BUSINESS IMPACT: TIME, COST, AND SUBMISSION READINESS

Automation using LLMs offers measurable improvements:

- Time Reduction:
    - SDTM and ADaM mapping: Faster with AI-assisted code generation.
    - TLF creation: More efficient development of standard tables.
    - Document updates: Expedited population of define.xml fields and reviewer guides.
- Cost Efficiency:
    - Reduces reliance on external FTEs or CRO overages.
    - Frees up senior programmers to focus on complex derivations, insights, innovation, or exploratory analysis.
- Submission Readiness:
    - More consistent outputs across studies.
    - Improved traceability through AI-generated annotations.
    - Fewer last-minute surprises due to standardized and quality-controlled automation.

These gains compound across studies and therapeutic areas, creating a compelling case for adoption.

CAPABILITIES OF LLMS RELEVANT TO LIFE SCIENCES

Let's explore specific ways LLMs can support clinical programming.

A. NATURAL LANGUAGE TO CODE TRANSLATION

LLMs can take input like:

"Create a variable for baseline weight using the first non-missing measurement prior to randomization"

… and return executable code in SAS, Python, or R. With prompting techniques, the model can even follow organizational coding standards.

B. SPEC-TO-CODE GENERATION

When provided with SDTM or ADaM metadata specifications in Excel or structured text, LLMs can generate starter code that reads source data, applies mappings, and performs necessary transformations. This is not about eliminating validation - but accelerating the first draft and reducing common errors.

C. METADATA INTERPRETATION AND MAPPING SUGGESTIONS

LLMs can read define.xml files, annotated CRFs, documents (e.g. Statistical Analysis Plans) or controlled terminology catalogs and suggest:

- Domain-variable mapping logic
- Missing documentation
- Inconsistencies between planned and programmed datasets

This allows human reviewers to focus on adjudication and decision-making, not rote documentation.

EXAMPLE FRAMEWORK OF A SOLUTION

The below Figures 1-4 provide an example framework of a solution. From our goals and common workflow, these highlight:

- Figure 1. Selecting file locations of data and agents (LLM(s)) to use.
- Figure 2. Selecting standard target metadata for data to be transformed.
- Figure 3. Extracting logic from documents (e.g. SAP, TLF shells, standard macros/programs).
- Figure 4. Generating the code.

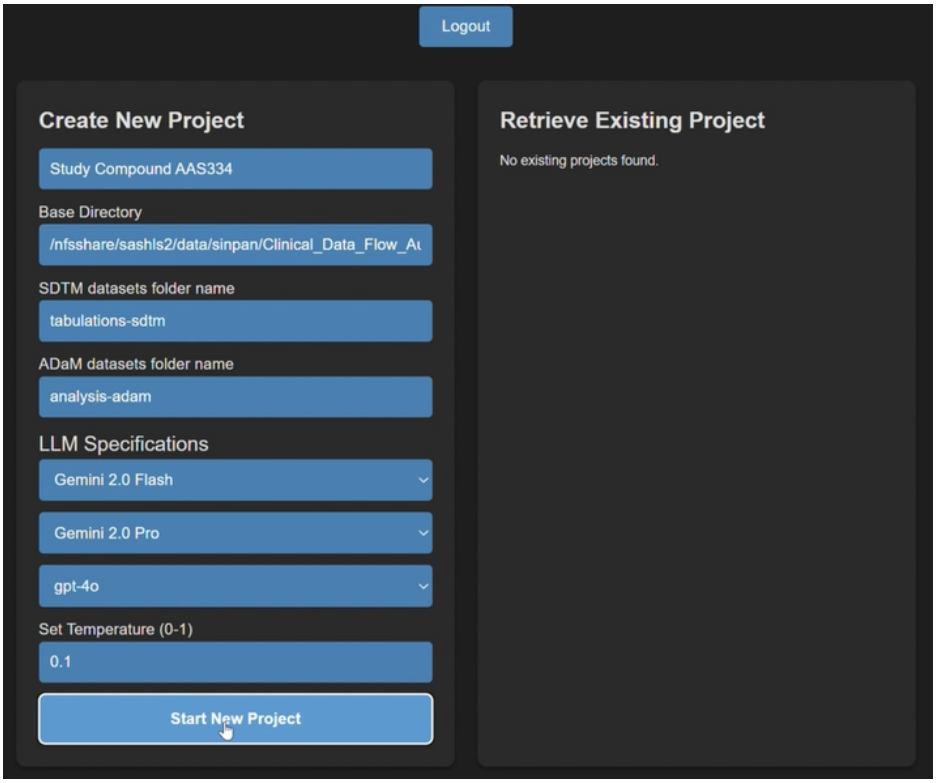

Figure 1. Specify File Locations and Choose Model(s)

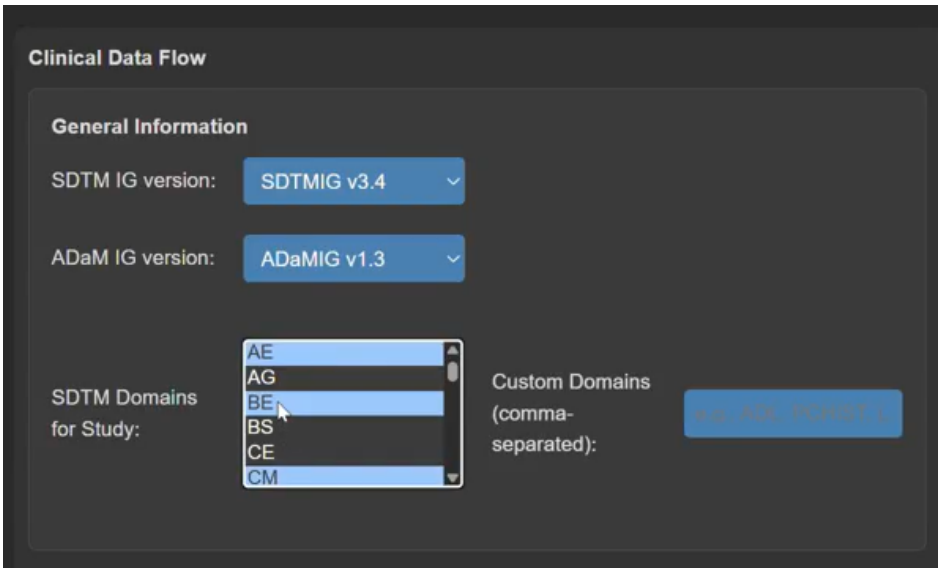

Figure 2. Select Implementation Guide(s) and Datasets to Transform

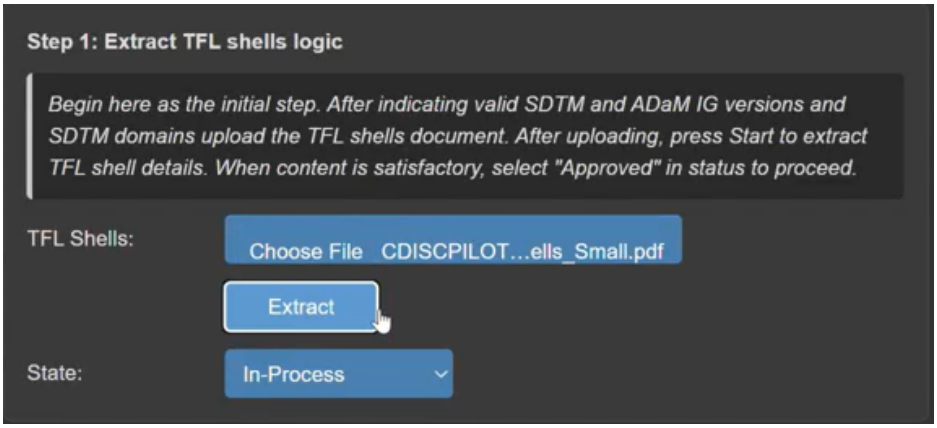

Figure 3. Extract TLF Shell Logic from SAP / TLF Shells

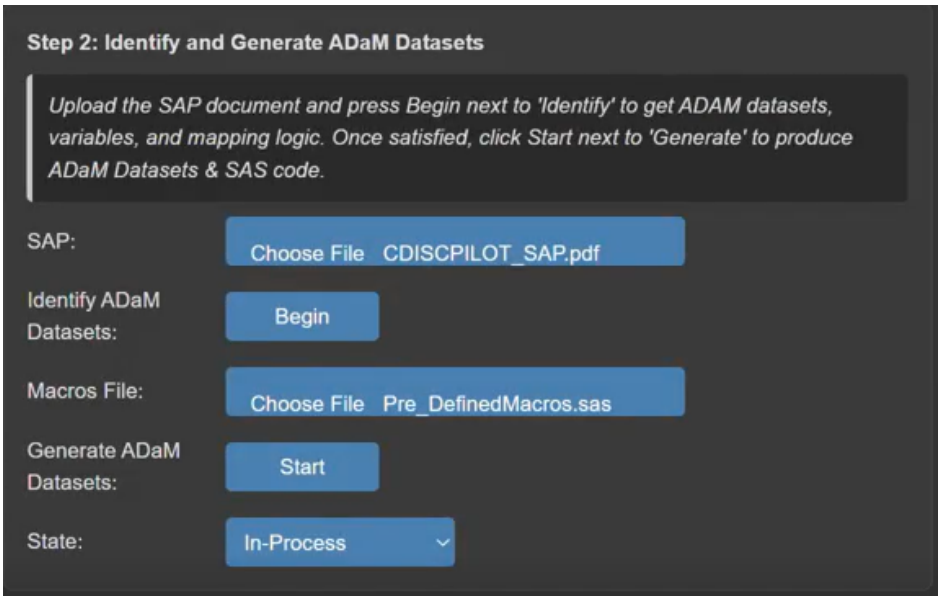

Figure 4. Generate ADaM Code

AI AS A COPILOT, NOT A REPLACEMENT

Let's be clear: LLMs will not replace clinical programmers. But they will fundamentally change how we work.

Here's the paradigm shift:

- Old Model: Programmer writes code → another reviews → documents built separately.
- New Model: AI generates first draft → human reviews, edits, and validates → AI assists in documenting.

Think of augmented intelligence as a team effort. AI can take care of the routine, repetitive tasks, freeing us up to focus on what we do best: applying our expertise, making insightful decisions, and coming up with innovative solutions. It's about combining the strengths of both humans and machines.

Framing AI as a copilot:

- Preserves human accountability
- Encourages trust and transparency
- Makes AI adoption easier to scale within regulated teams.

RESPONSIBLE ADOPTION: RISKS AND MITIGATIONS

No technology is without risk. With LLMs, concerns include:

- Inaccurate or hallucinated code
- Data privacy when using public models
- Regulatory readiness of AI-generated outputs.

Mitigation strategies:

- Use private or fine-tuned models within secure environments.
- Establish human-in-the-loop workflows with clear documentation.

- Validate AI outputs as you would with any junior programmer's work.
- Document AI usage and QC steps for traceability.

CONCLUSION

Clinical programming is evolving. The burden of manual coding, documentation, and rework is no longer sustainable - and we now have tools to break that cycle.

GenAI is not science fiction. It's already being used to:

- Map datasets
- Generate analysis-ready code
- Write submission documentation.

But it must be adopted deliberately, responsibly, and collaboratively - with humans firmly in control.

The organizations that embrace this shift will enjoy faster timelines, greater consistency, and empowered teams. Those that don't may find themselves struggling to keep up.

FINAL THOUGHTS

- Start small: Pilot LLMs on common code or document generation
- Measure impact: Time saved, errors reduced, consistency gained
- Upskill teams: Teach prompt engineering and review best practices
- Collaborate across functions: Involve QA, regulatory, and IT early

Together, we can reshape clinical programming - not by replacing the people, but by giving them the tools they need to thrive in a faster, more complex world.

ACKNOWLEDGEMENTS

I would like to acknowledge the SAS Life Science R&D team, specifically Kim Peplinski, Robertson Williams, and Neeraj Shrivastava for their continual support and innovation around GenAI. Moreover, our Head of Product Strategy, Next-Generation AI Technologies at SAS, Brett Wujek, has been instrumental in the innovation in this space.

I would like to acknowledge Pankaj Attri at SAS for his leadership and development of MVPs (minimally viable prototype) that were showcased in this paper.

Finally, there are so many individuals in the industry and at SAS who have helped guide our continual evolution in GenAI for our space: Paul Slagle (IQVIA), Mike McDevitt (Syneos Health), Jim Box (SAS), Pritesh Desai (SAS), Samiul Haque (SAS), Brittany Shriver (SAS) and so many more I have inadvertently left off this mention.