



From Mock to Output: Using Generative AI to Write SAS Code for Standards-Based TFLs

Kala Shivali
Bhuvana Venkatappa





Kala Shivali

Executive Director,
Services



kala.shivali@
sycamoreinformatics.com



Cary , NC , USA



[https://www.linkedin.com/in/
chandrakala-shivalingaiah-
443b333/](https://www.linkedin.com/in/chandrakala-shivalingaiah-443b333/)



Professional with experience across BioPharma, Biotech, CRO, CRM, finance, and IT. Holds a B.E. in Computer Science from Bangalore University and an MBA in Bioscience from NC State University. Currently leading the Service team at a SaaS company for BioPharma and Biotech, with a focus on leadership and structured problem-solving. Outside work, enjoys reading, podcasts, marathons, and hiking.



Bhuvana Venkatappa

Business Analyst



bhuvana.venkatappa@
sycamoreinformatics.com



Miami , FL , USA



[https://www.linkedin.com/in/
bhuvanavenkatappa/](https://www.linkedin.com/in/bhuvanavenkatappa/)



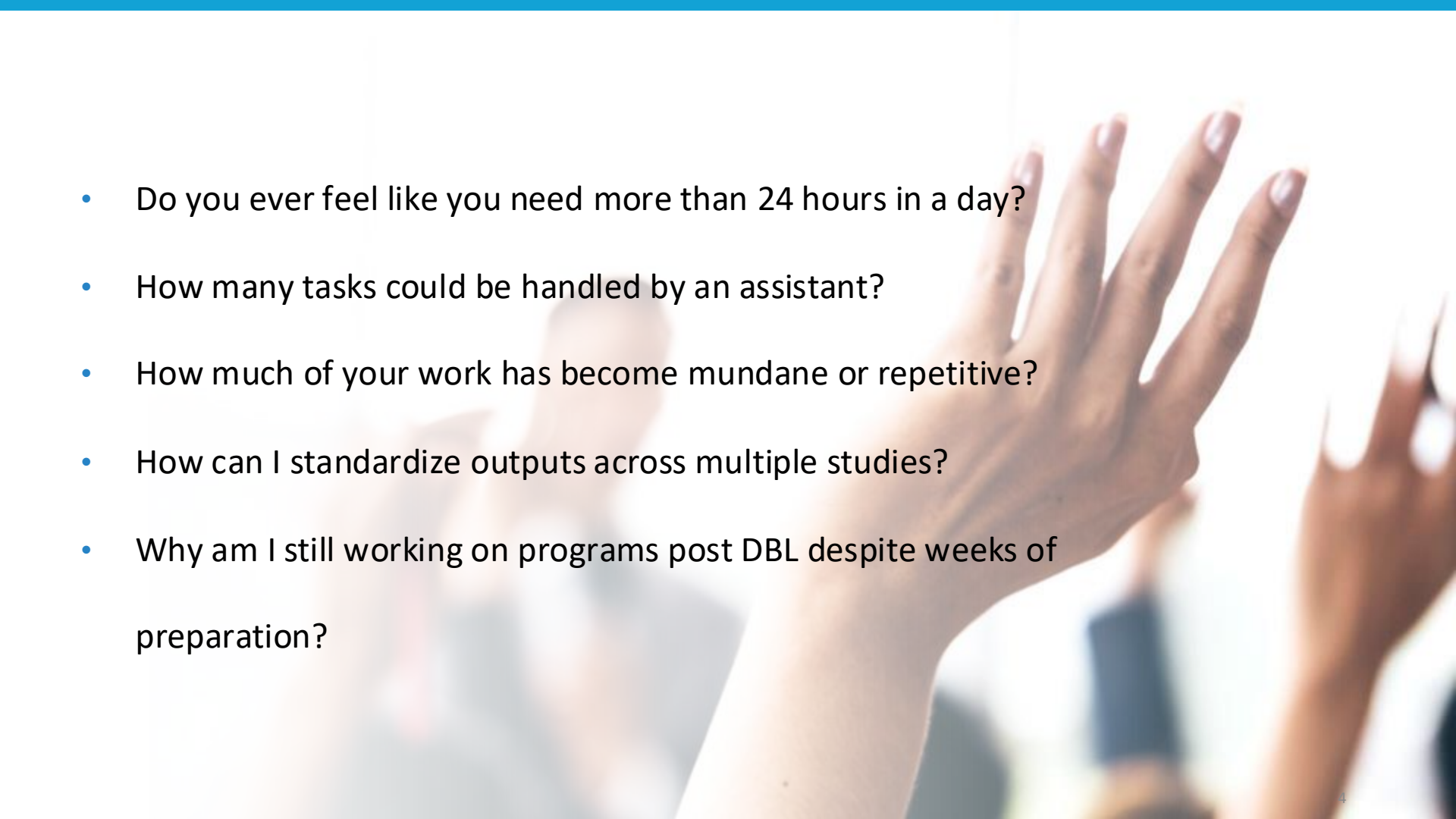
AI enthusiast and recent graduate in Electrical and Computer Engineering (AI major) from the University of Miami. Currently working as a Junior Business Analyst at Sycamore Informatics, aspiring to drive impactful projects in NLP, automation, and Generative AI. Published research on smart automotive systems and safety in electric vehicles. Experienced in client demos, product testing, and training, with contributions to client/internal training materials and mentoring.

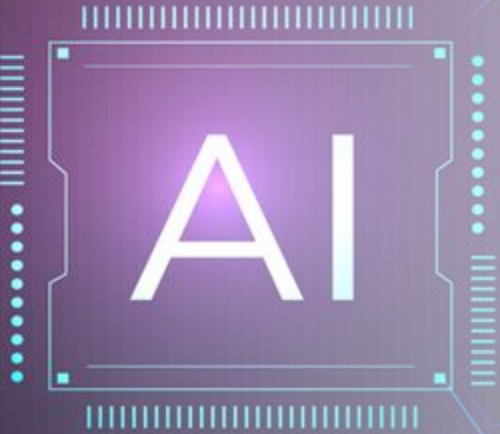


Agenda

- What's on your mind?
- The Paradigm Shift in Industry
- The Architecture - Overview
- The Code Generation - Workflow
- The Benefits and the HITL
- Conclusion



- 
- Do you ever feel like you need more than 24 hours in a day?
 - How many tasks could be handled by an assistant?
 - How much of your work has become mundane or repetitive?
 - How can I standardize outputs across multiple studies?
 - Why am I still working on programs post DBL despite weeks of preparation?

A stylized graphic of a microchip with the letters 'AI' in the center. The chip has a square shape with rounded corners and a series of small squares along its edges, representing pins or connections. The background of the chip is a gradient of purple and blue. The letters 'AI' are white with a slight glow.

AI

**Is The Solution To Automation,
Standardization, And
Efficiency Gain In Clinical
Programming**

The Paradigm Shift in Industry



Prospective Standards

Modern trends emphasize prospective planning using CDISC Analysis Results Standards (ARS).

The Analysis Results Metadata Technical Specification (ARM-TS) acts as a machine-readable "single source of truth" prior to code development.



AI Automation Leverage

Generative AI (LLMs) leverages this ARM-TS and sponsor-defined rules to automate the conversion process.

This approach moves development from manual coding to metadata-driven synthesis of GxP-compliant SAS code.

Regulatory Landscape



Regulatory bodies are publishing guidance for the use of AI/ML and LLM models
Emphasis on transparency, interpretability, human oversight, and explainability



Key Considerations for AI Adoption in Regulated Environments

- Identify top priorities for adopting AI while ensuring compliance.
- Balance innovation with risk management in regulated workflows.
- Spot high-value, low-risk use cases for early AI adoption.
- Assess organizational AI maturity — experimenting, piloting, or scaling?
- Understand major concerns or blockers to AI adoption in regulated functions.



The Architecture - Overview

Orchestration Layer

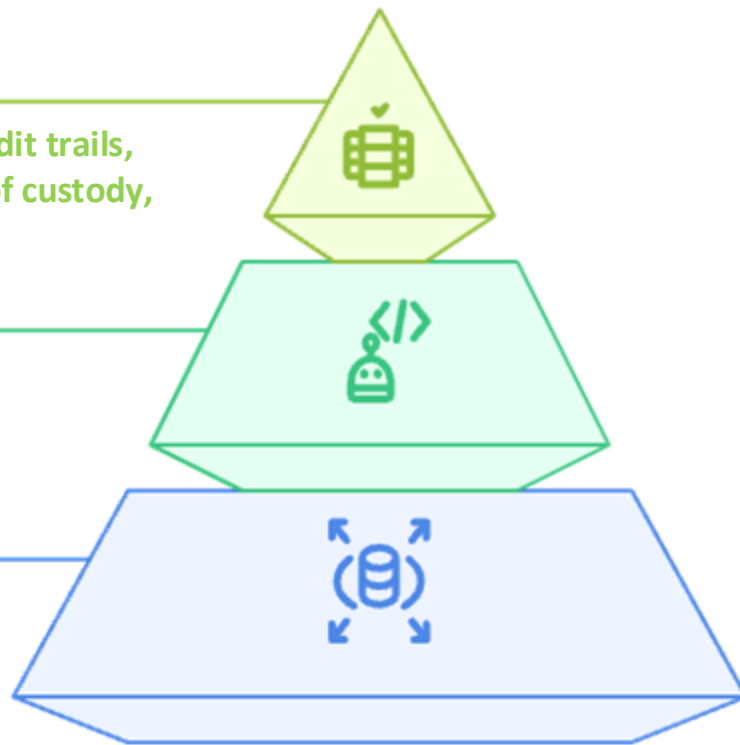
Executes exclusively within a GxP-compliant audit trails, granular access control, version control, chain of custody, traceability, and reproducibility

Generative AI Layer

Algorithmic constraints are applied to enforce reproducible, predictable code generation.

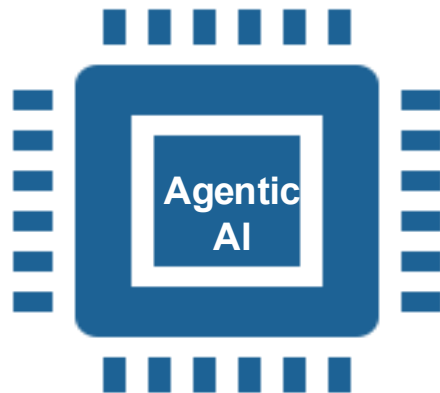
Metadata Layer

Structured metadata aligning with pre-approved specifications

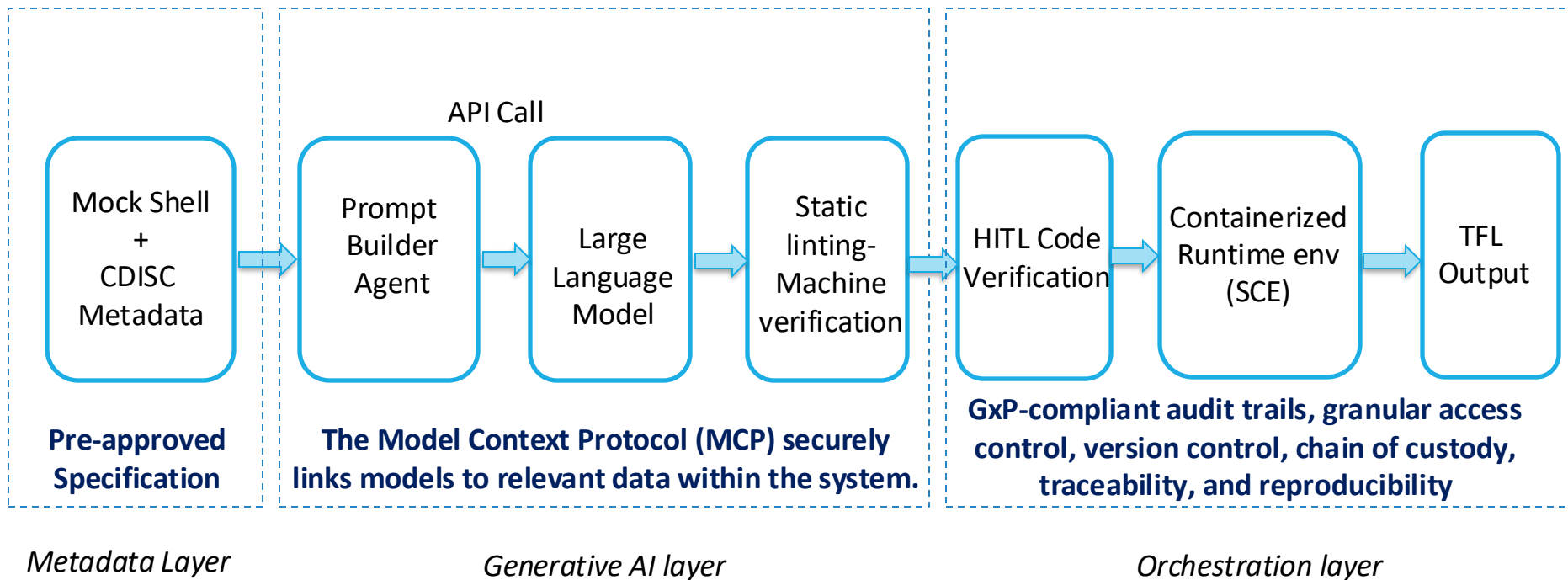


Agentic AI for Code generation

- Mock Shells
- Macros
- Data
- Meta data



Agentic AI for Code generation - Workflow



Benefits and the Human-in-the-Loop (HITL)



Key Benefits

- Speed: Eliminates the 100% overhead of dual-programming.
- Quality: Ensures consistent adherence to analytical intent.
- Compliance: Built-in, automated audit trails.
- Reproducibility: Guaranteed identical output from identical inputs.
- Scalability: Rapidly produce hundreds of standardized TFLs.



Human-in-the-Loop (HITL)

Programmers serve as the ultimate Validation Gateway, reviewing AI-generated code to ensure analytical intent was correctly translated.

This frees specialized teams to focus on complex scientific questions and statistical methodology rather than repetitive code generation.



Gen AI – How it can help

- Key Objective: SAS program generation
- It functions as an expert "co-pilot" for SAS programmers in the driver's seat
- You can make LLMs think "only as much as the GxP rules allow."
- Unlike Open source Chatbots, no hallucination or creativity



Future Directions & Conclusion

The transition to this framework sets the stage for continuous improvement and strategic expansion:

- ▶ **Self-Correction:** Integration of Reinforcement Learning from Human Feedback (RLHF) to continually refine compliance and accuracy.
- ▶ **Strategic Expansion:** Moving to a Cross-Study Metadata Repository for portfolio-level automation (e.g., ISS/ISE).
- ▶ **Advanced Orchestration:** Evolution toward Multi-Agent Systems for collaborative, end-to-end study reporting.

This framework enables the pharmaceutical industry to transition clinical reporting to a modernized, scalable, and error-resistant process, securing quality and accelerating insight simultaneously.



Questions?

Thank You

Summary

- ✓ Manual TFL coding causes variability, rework, and slow reviews.
- ✓ Executable shells and CDISC metadata drive automated, guard-railed SAS generation.
- ✓ Programmers focus on review and QC instead of repetitive coding.
- ✓ The approach can extend to:
 - Reinforcement Learning from Human Feedback (RLHF)
 - Portfolio-Level Automation
 - Multi-Agent Systems



The Traditional vs The AI Enabled Approach

Dimension	Traditional Approach	AI-Enabled Approach
Speed	Manual coding and review cycles	Automated code generation and validation
Quality	Subjective interpretation	Consistent adherence to standards
Compliance	Manual traceability	Built-in audit trails, GxP compliance
Reproducibility	Study-specific manual execution	Fully reproducible runs across studies
Scalability	Resource-intensive	Parallel generation across multiple TFLs



Architecture Layer 1: Data & Metadata Context

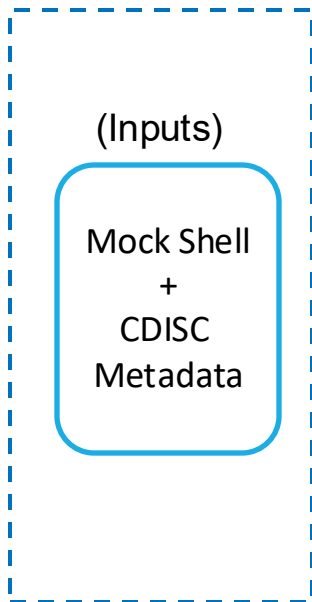
Purpose: Establish the authoritative single source of truth and enforce data security boundaries.

- ✓ **Study Metadata Repository (SMR):**
Centralized source for all specifications.
- ✓ **Structured Input/Data Model:** Provides the foundation for LLM grounding.
- ✓ **Compliance Gateway:** Enforces the Model Context Protocol (MCP).



Security: Ensures only non-sensitive metadata is used for LLM context, while Protected Health Information (PHI) remains segregated in the SCE.

The Workflow – Step 1 : Metadata Layer



Metadata Layer

- **Study Metadata Repository (SMR)**
 - Single source of truth for study specs and standards.
 - Stores ADaM/SDTM definitions, derivations, and ARM-TS
- **Structured Input & Data Model**
 - Inputs are machine-readable and traceable
 - TFL shells in JSON/XML; ARD - reusable results repository
- **Secure Data Control**
 - Controlled access to ADaM/SDTM and ARD datasets.
 - PHI remains isolated; only metadata is shared with the model.



Architecture Layer 2: Generative AI & Constraint Mechanisms



Prompt Builder Agent:

Transforms specifications into a constrained instructional payload for the LLM.



Deterministic Injection:

Performs Fixed Seed and ultra-low Low-Variance Decoding to enforce run-to-run reproducibility.



SAS Function Guardrails:

Applies whitelists to restrict the LLM to pre-validated procedures/macros, preventing high-risk commands (e.g., X command).

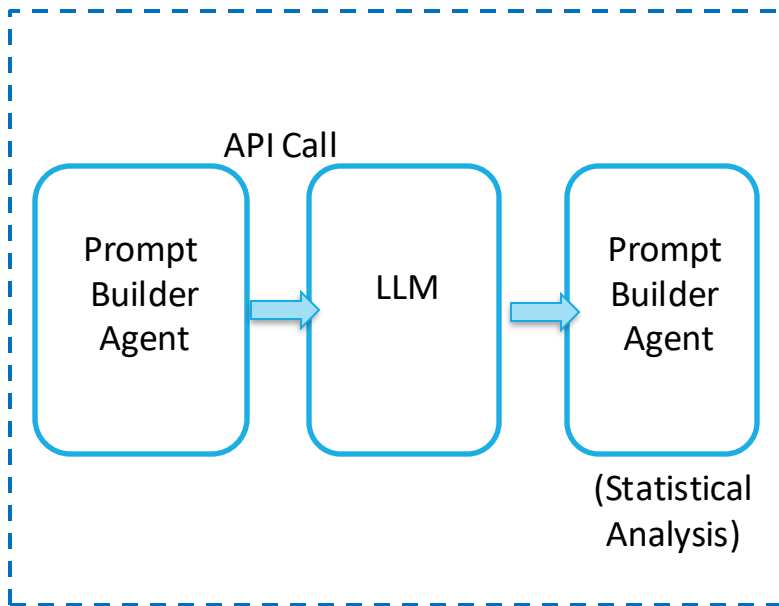


Static Validation:

Immediately checks the generated code for GxP markers and adherence to coding standards before execution.



The Workflow – Step 2 : Generative AI Layer



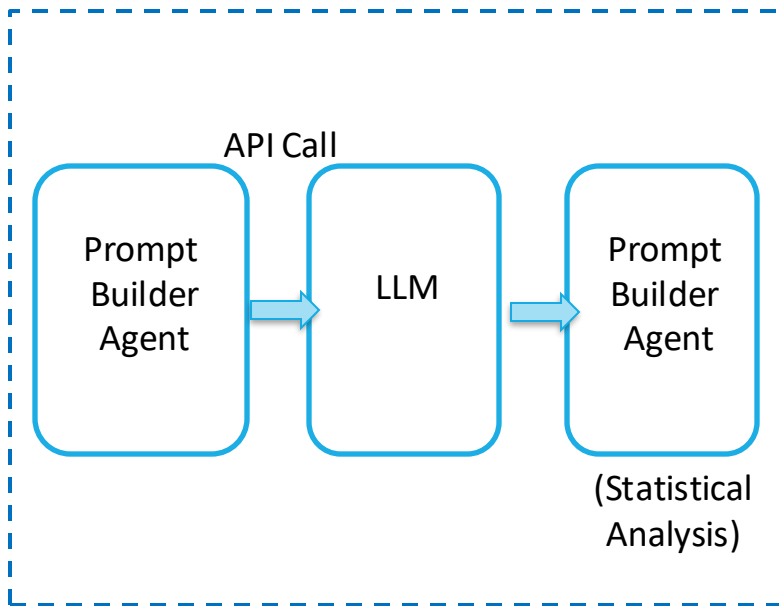
Generative AI layer

The Prompt Builder Agent

- i. **Core Request:** *What to create* — derived from shell columns.
- ii. **Contextual Data:** *What data to use* — based on CDISC variable names.
- iii. **Institutional Constraint:** *How to code* — must use approved macros
- iv. **Deterministic Control:** *Auditability* — uses fixed seeds and SPEC_ID injection.



The Workflow – Step 2 : Generative AI Layer



Generative AI layer

The LLM's Generation Process

- Fixed Seed Control
- Low-Variance Decoding
- Constrained Output Schema
- Macro Function Guardrails

Static Validation: Automatically verifies syntax, standards, audit markers, and output format before execution.



Architecture Layer 3: Execution & Orchestration



Execution

Code runs in a secure, versioned SCE environment with a Containerized SAS Runtime, guaranteeing identical execution every time (Reproducibility).



Governance & Audit

Finalizes Audit Trails by applying Cryptographic Hashing (SHA-256) to all output and log files for non-repudiable logs.



Traceability Linkage

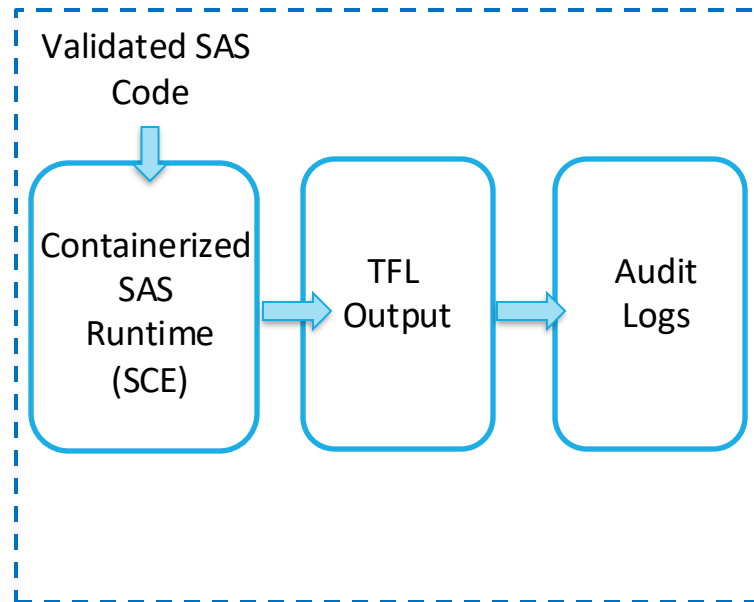
The system automatically connects the final TFL output version back to the generated code version and the original ARM-TS input.



The Workflow – Step 3 : Orchestration Layer

Validated Execution:

- Quality Control
- Containerized Runtime
- Reproducibility
- TFL Generation



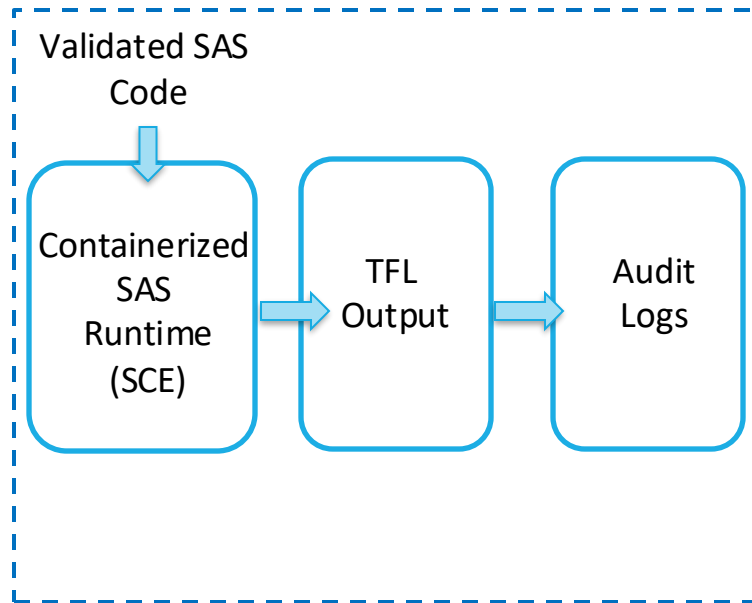
Orchestration layer



The Workflow – Step 3 : Orchestration Layer

Governed Execution:

- Audit Trails
- Traceability
- Data Lineage



Orchestration layer



Now Is the Right Time !!!

- Industry Standardization
 - Examples: USDM, CDASH, SDTM, ADaM, ARM, ARS, ARS-TS
- Technology Enhancements
 - Examples: R, Python, AI, LLMs, Agentic AI
- Regulatory Landscape
 - Regulatory bodies are publishing guidance for the use of AI/ML and LLM models
 - Emphasis on transparency, interpretability, human oversight, and explainability



Generative AI Framework: Core Technical Pillars

Grounded Inputs



Code generation is based on structured metadata (e.g., ARM-TS, CDSIC) ensuring outputs align with pre-approved specifications.

Deterministic Control



Algorithmic constraints (Fixed Seed, Low-Variance Decoding) are applied to enforce reproducible, predictable code generation.

Secure Orchestration



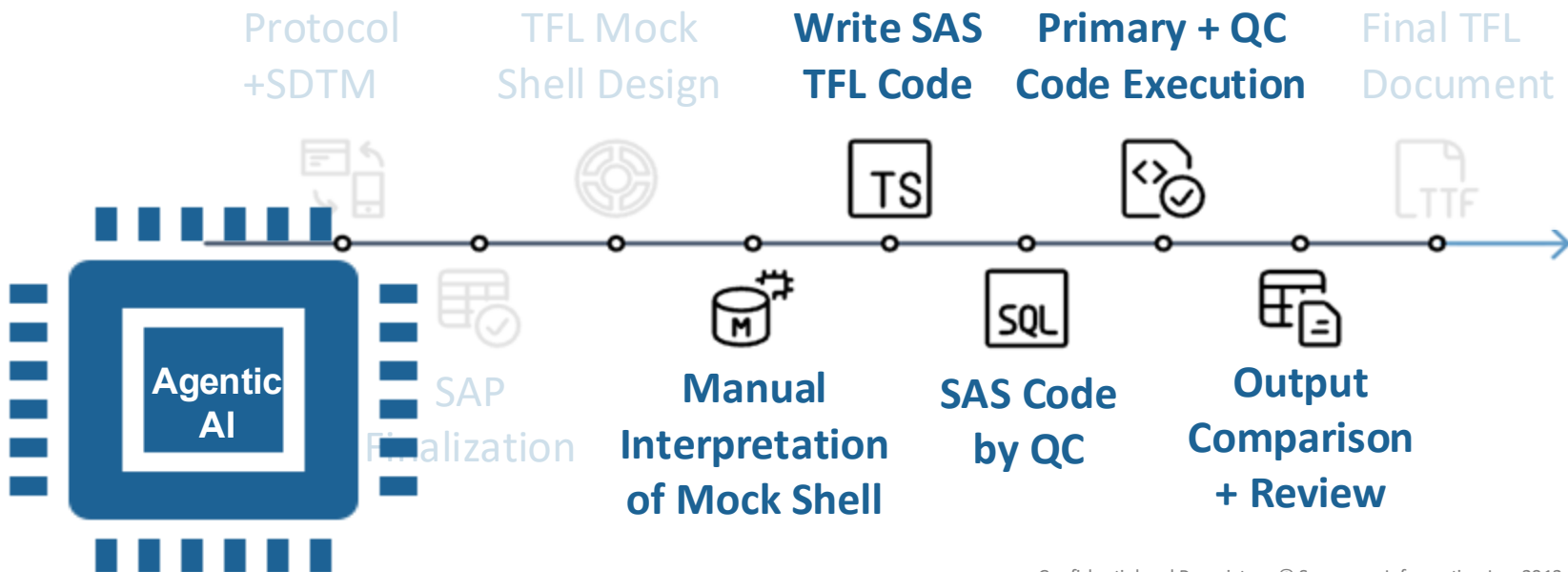
Execution occurs exclusively within a GxP-compliant, secure Statistical Computing Environment (SCE) with immutable audit trails.

LLM must be proficient in SAS syntax, CDISC contextual awareness, and adherence to institutional macro logic.

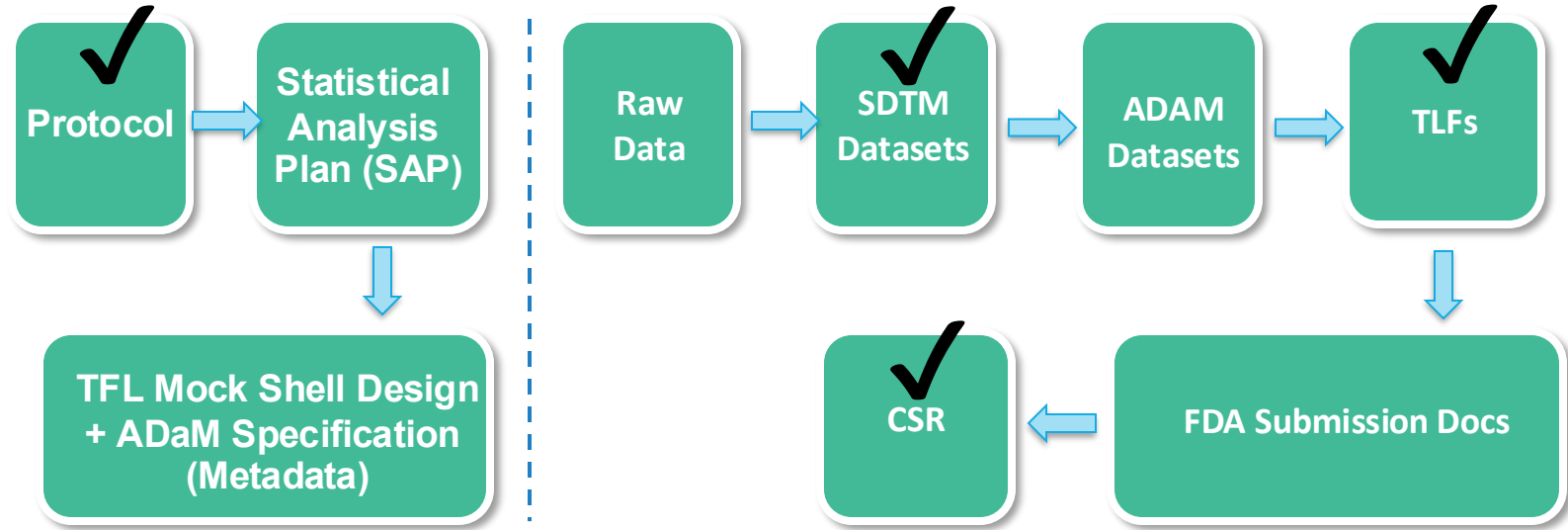


Agentic AI for Code generation

- Let's Identify where AI can add the most value today.
- Many intermediate steps can be reduced to just a few clicks? – Why not!



Opportunities to Automate in Clinical Trials



Data and Standardization

Data Program execution



AI Integration Point in the TFL Process

- Let's Identify where AI can add the most value today.
- Many intermediate steps can be reduced to just a few clicks? – Why not!

