Paper ML04

An Agentic Approach to Validation, Influencing Reporting with AI

Junibel R. De Leon, Pfizer Inc, Makati, Philippines
Karthik Palani, Pfizer Healthcare India Pvt Ltd, Chennai, India

ABSTRACT

Validating CSR summary tables remains resource intensive due to manual double programming and increasing data complexity. This work explores an agentic AI approach that automates parts of the QC process by combining large language models with retrieval augmented, stepwise code generation aligned to CDISC ADaM standards. The pilot focused on a simplified AE summary table by Preferred Term as an initial proof of concept, with future plans to extend to SOC, CTCAE and additional dimensions. Early tests across model families including T5, CodeGen and Deepseek coder showed limited text to code accuracy, prompting the creation of a curated instruction dataset, RAG based prompt matching and post processing to produce approximately 80 percent executable R code. The agentic workflow evaluates logical flow, executes generated code and supports human review of workflow. Although constrained by training data and system resources, this approach shows potential to reduce manual programming burden and support scalable, reproducible AI driven validation in clinical reporting.

INTRODUCTION

Clinical study report (CSR) summary tables play a central role in communicating safety results, yet validating these tables continues to rely heavily on manual double programming. As clinical trial datasets grow in size and complexity, this manual process becomes increasingly time-consuming, difficult to scale, and vulnerable to inconsistencies. These challenges highlight the need for approaches that can improve efficiency while maintaining the accuracy and transparency required in regulated environments.

Agentic artificial intelligence (AI) offers a potential solution by assisting with multi-step tasks such as interpreting table specifications, identifying relevant data elements, drafting reproducible QC code, and comparing results. Unlike traditional automation, agentic AI can follow structured workflows, apply rules, and generate intermediate outputs that a reviewer can inspect and validate. Importantly, human oversight remains central: the AI assists the programmer but does not replace expert review or judgment.

To ensure feasibility and a controlled scope, this pilot focuses exclusively on a simplified AE (Adverse Event) summary by Preferred Term (PT) as the initial proof-of-concept. This reflects the team's decision to start with a narrow AE PT use case before extending to SOC or other dimensions. The approach leverages a retrieval-augmented workflow, where a user prompt is matched to a predefined set of stepwise instructions that guide the model in generating R code. Among the models evaluated, the CodeGen model was selected after demonstrating the most reliable text-to-code behavior for this task.

Specifically, the pilot aimed to determine whether an agentic AI approach could:

1. interpret table requirements through retrieval-driven step instructions,
2. generate R code that reflects expected logical flow,
3. produce code that is at least partially executable with minimal post-processing, and
4. maintain traceability, data privacy, and human oversight.

This pilot does not attempt to automate full CSR generation or extend to more complex AE table families. Instead, it serves as a focused test case to understand the feasibility, limitations, and next steps of applying an agentic AI method to clinical reporting workflows.

METHODS

Overall Approach

This pilot focuses exclusively on a simplified AE summary by Preferred Term (PT) as the initial proof-of-concept, reflecting the team's decision to begin with a narrow structure before expanding to SOC or additional dimensions. The intent is to explore whether an agentic AI workflow can assist in generating draft QC code while maintaining transparency and human oversight.

To illustrate how the components of the pilot fit together, Figure 1 provides an overview of the end-to-end workflow, including the inputs supplied by the user, the retrieval-augmented instruction selection process, model-generated stepwise code, post-processing activities, and the resulting draft R script for the AE PT summary table. This visual summary reflects the current PT-only prototype and highlights the stages where human review remains essential.

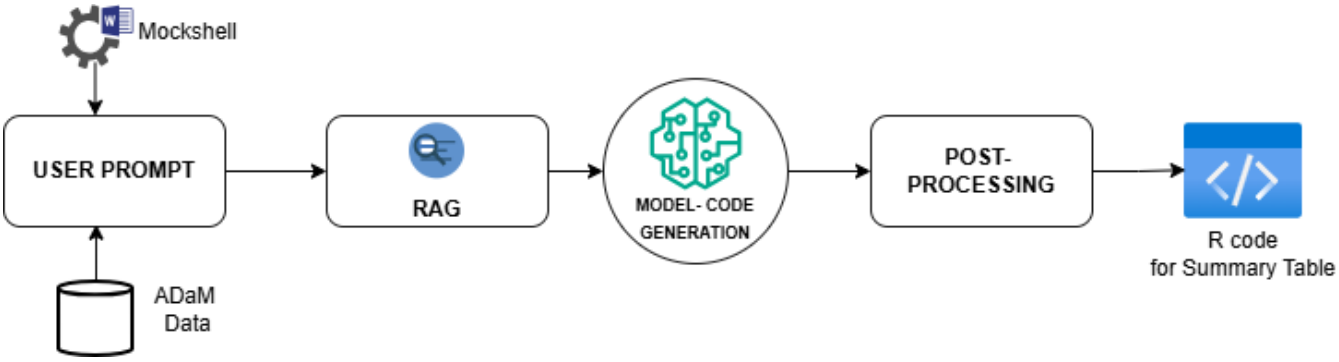

Figure 1. Agentic AI Workflow for AE PT Code Generation

Inputs and Outputs

Inputs include ADaM datasets (ADSL and ADAE), mock shells describing the expected structure, and user-provided details such as paths to datasets, table labels, and filters, as referenced during model testing. Outputs include a generated R script and intermediate diagnostic information, with post-processing applied to assemble stepwise code into an executable script, as described in the latest development update.

Workflow

1. Retrieval-Augmented Instruction Selection

The workflow uses a RAG-based bridge, where the user prompt is matched through cosine similarity to a curated data frame of "report names" mapped to stepwise instructions. These instructions represent the logic required to generate an AE PT summary and are fed sequentially to the model to produce corresponding code snippets.

2. Model-Based Code Generation

Multiple models were evaluated, and the CodeGen model was adopted after demonstrating more reliable text-to-code behavior than the other alternatives (e.g., Flan-T5 base). For each matched instruction, CodeGen produces an R code fragment reflecting that step in the logic.

3. Post-Processing of Generated Snippets

Because raw model output is not fully executable, post-processing is performed to combine stepwise fragments, reorder ADSL and ADAE operations, correct syntactic issues (such as variables beginning with underscores), and assemble the content into a single structured QC script.

4. Execution and Review

The combined R script is evaluated for logical flow and executability, consistent with the team's emphasis on these metrics in the current phase. Full value-level comparison against production outputs is planned for later, once scope expands beyond PT-only logic.

5. Traceability and Oversight

All generated logic is subject to human review, and model outputs are verified manually.

Technical Environment

- The process uses automated code to read files, interpret shells, and run comparisons.
- A locally hosted AI model helps draft the specification and code but does not access the internet and does not run any data outside the secure environment.
- All data stays within the organization's protected network.
- The programmer retains full control and reviews all outputs.

Pilot Scope and Limitations

The pilot is intentionally limited to the AE PT summary because the training data and RAG repository were built from only one AE PT report, with expansion to SOC, severity, or additional AE domains requiring further generalization of the instruction set and training data.

Metrics and Evaluation

The team evaluates model performance using code-focused metrics, including:

- the proportion of generated code that is directly executable without modification,
- the extent to which the generated logic follows the expected step-by-step sequence, and
- whether essential operations (such as grouping, sorting, filtering, and variable derivation) appear in the correct order and reflect the intended logic.

These metrics reflect the current emphasis on assessing logical flow and code structure, rather than final table-level concordance, which will be explored in later phases as the workflow expands beyond the initial PT-only prototype.

RESULTS

The pilot demonstrated that an agentic AI workflow can support the creation of first draft QC code for a simplified AE summary by Preferred Term (PT), while reflecting the model-behavior challenges.

Early attempts using prompt-only generation were unreliable, and models such as Flan-T5 base and Deepseek coder produced incomplete or inconsistent logic that did not reflect the expected sequence of steps. This led the team to adopt a retrieval-augmented, stepwise generation process, where a user prompt is matched to a curated set of instructions that the model processes one step at a time. Among the models evaluated, the CodeGen model produced the most reliable text-to-code results and was therefore used for the current PT-only prototype.

The resulting workflow generated code snippets for each step, which were then combined through post-processing to address syntactic issues and ordering problems. When executed, the consolidated script was partially executable, with several steps aligning to the intended logic, though additional refinement was needed for variable handling, filter integration, and syntax corrections such as handling identifiers beginning with underscores.

The team also observed variability across runs, where identical prompts sometimes produced different outputs. This inconsistent behavior reinforced the need to evaluate performance based on logical flow and executability rather than value-level concordance, which will be addressed in later phases.

Overall, the system did not achieve full automation, but it provided a traceable, step-aligned draft of the intended QC logic and reduced the initial effort required to assemble a baseline script. These outcomes are consistent with the team's view that the work is an early research-stage prototype, with further development needed before evaluating broader table types or more complex AE structures.

DISCUSSION

This pilot explored whether an agentic AI workflow could meaningfully support the validation of a simplified AE summary by Preferred Term (PT). The results show clear potential, but also confirm the limitations raised by the team during development.

The main strength of the approach lies in its ability to provide a structured, stepwise starting point for QC code, generated through retrieval-driven instructions and model-based code creation. The system produced fragments that reflected the intended logic and allowed reviewers to assess the overall flow, which offered value during early stages of validation where initial code construction is time consuming.

However, the pilot also revealed important constraints. As noted previously, prompt-only methods were not reliable, and the models evaluated generated incomplete or inconsistent steps, especially when logic required variable derivation, ordering, or preprocessing. Flan-T5 base and Deepseek coder models did not meet expectations, while the CodeGen model delivered comparatively better text-to-code behavior, though still requiring post-processing to assemble executable code. These challenges were expected, given the limited training data, the use of only one AE PT report for fine-tuning, and the complexity of clinical programming logic.

Reproducibility was also a significant issue. Team members observed that identical prompts sometimes produced different outputs, which is inconsistent with established QC expectations. While post processing and human review improved stability, the model's variability confirmed that deterministic behavior cannot yet be assumed.

Finally, the current implementation remains PT only and does not yet cover SOC or additional AE structures. Extending the workflow to more complex table types will require a generalized instruction repository, broader training examples, improved metadata interpretation, and further tuning of the generation pipeline.

Overall, the pilot demonstrated that agentic AI can augment early QC code development by providing a transparent, step aligned logic draft. At the same time, it highlighted important limitations, particularly around training data sufficiency, reproducibility, and scope, that must be addressed before expanding to more complex reporting scenarios. Continued iteration, expanded examples, and sustained human oversight will be essential for future development.

FUTURE WORK

Future work will focus on improving the reliability and generalization of the current PT only prototype.

First, the instruction repository needs to be expanded and generalized beyond the single AE PT report used for training so that the retrieval process can support additional AE table types and eventually SOC and severity-based summaries.

Second, the model requires more diverse fine-tuning examples to strengthen its understanding of clinical programming patterns and improve the consistency of stepwise logic generation.

Third, the retrieval augmented workflow should be refined by enhancing the clarity and coverage of step instructions, ensuring that key operations such as grouping, filtering, sorting, and derivations are captured accurately.

Fourth, post processing should be improved to reduce manual corrections by embedding common syntax rules and formatting expectations into training data so that generated code is closer to fully executable form.

Finally, the team will formalize logic focused evaluation metrics, since value level accuracy will become meaningful only after expanding beyond the PT only prototype.

CONCLUSION

This pilot demonstrated that an agentic AI approach can provide early support for generating QC code for a simplified AE summary by Preferred Term (PT). The workflow was able to interpret stepwise instructions, apply a retrieval driven process, and produce draft R code that reflected the intended logic, although additional post processing was required. These results show that the approach can assist with the initial stages of QC programming, especially when time and resources are limited.

The work also highlighted several limitations. The model outputs were not fully reliable, and the CodeGen model, while the most effective among those tested, still produced incomplete or inconsistent steps that required manual review and correction. The team also observed variability across runs, confirming that consistent and deterministic behavior is not yet achievable.

Looking ahead, future progress will depend on expanding the instruction repository beyond the single AE PT report, increasing training examples, refining post processing, and improving the clarity and coverage of the stepwise instructions used in retrieval. Broader evaluation across additional AE structures, including SOC and severity, will be necessary in later phases before value level comparisons become meaningful.

Overall, the pilot offers an early demonstration of how agentic AI can augment QC code development by providing a traceable and structured starting point. Continued iteration, additional data, and sustained human oversight will be essential to advance this approach toward more complex reporting scenarios.

REFERENCES

Microsoft. (2024). *Microsoft Copilot* [Large language model]. Microsoft. https://www.microsoft.com/copilot

DeepSeek AI. (2024). *DeepSeek Coder 1.3B* [Large language model]. DeepSeek. https://www.deepseek.com

Nijkamp, E., Pang, R., Hayashi, H., et al. (2022). *CodeGen: An open large language model for code with multi-turn dialogue capability* [Large language model]. Salesforce Research. https://github.com/salesforce/CodeGen

Chung, H. W., Le, Q. V., & Google Research Team. (2022). *FLAN-T5* [Large language model]. Google. https://github.com/google-research/t5x

ACKNOWLEDGMENTS

The authors thank all members of the Agentic AI working group for their time, openness, and collaboration throughout this pilot. We acknowledge the contributions of Muhammed Farzan K, Mayur Mishra, and Alisha D, whose experimentation with different model families and prompt structures helped shape the direction of this work. We also thank Periasamy Karuppannan and Priya Venkatesan Iyer for their guidance on scope, expectations, and refinement of the study approach.

We are grateful to the broader programming and data sciences teams for their feedback during discussions and for sharing practical insights related to QC workflows, reproducibility considerations, and the needs of CSR validation. Their input was essential in assessing the usefulness and limitations of agentic AI for clinical reporting.

Finally, we acknowledge all colleagues who supported coordination, documentation, and planning activities during this research and development phase. Their collaboration ensured that the pilot progressed with clarity and shared purpose.

CONTACT INFORMATION

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

Author Name: Junibel R. De Leon
Company: Pfizer, Inc.
Address: 19/F 8 Rockwell Building, Rockwell Center, Hidalgo Drive, Makati City, Philippines 1200
Work Phone: +639178039578
Email: junibel.deleon@pfizer.com
Website: https://www.linkedin.com/in/junibel-de-leon-4973522a/