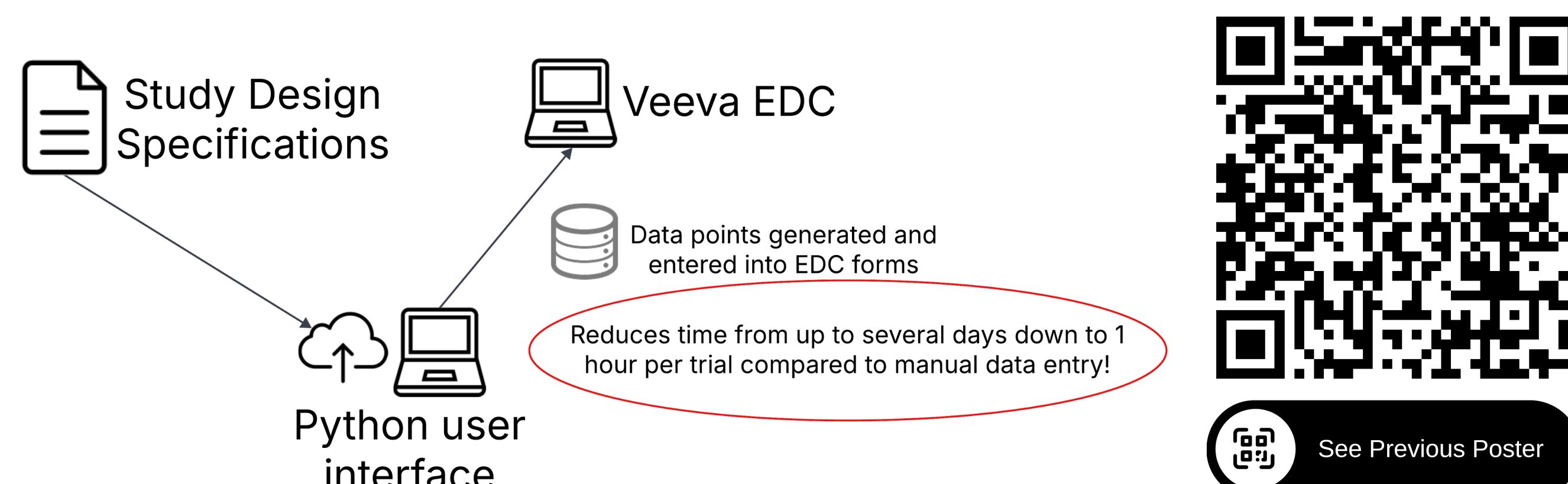




Abstract

Generating and entering test data into electronic data capture (EDC) systems is a critical yet laborious task, essential for validating study designs and numerous downstream systems and programs. We previously leveraged open-source technology to develop a tool to significantly ease the time-intensive effort of manual test data entry and notable efficiency gains have been achieved since its release; however, it falls short in effectively covering the full spectrum of test cases necessary for clinical system set-up and programming. To overcome these challenges, Takeda has sought to build on this innovative software solution to harness the power of large language models (LLMs) to expand the range of covered testing scenarios. During the development process, strategies and best practices for maximizing LLM potential in the clinical test data context have been established, along with valuable lessons learned along the way. Exciting potential future directions have been identified to further reduce clinical trial system set-up times.

Prior to Large Language Model (LLM)



Design Highlights

- Rules-based test data generation
- API connection for test data entry

Efficiency Gains: Early Testimonials

- Listings programmers: **~1 week saved per 20 listings**
- Submissions programmers: **~60 hours saved per study**
- Pipeline testers: **on-demand** test data without coordination delays
- Used on **40+ new and active trials** since April 2024

LLM Integration

Why Large Language Model?

Large Language Models (LLMs) excel at interpreting unstructured study design information and generating context-aware outputs. Unlike traditional rule-based approaches, LLMs can reason across **variability** in visit names, subject statuses, and study structures.

LLM prompts were therefore incorporated as **targeted components**, while Python remained the primary orchestration layer.

Goals and Problems Solved

Goal	Problem Solved
Integrate LLM into existing application in clinical domain as a targeted tool	Get team comfortable working with LLM and understand domain-specific prompt engineering strategies
Generate test data specific to subject status (complete, screen failure, or early termination)	Specific data points on certain forms needed for clinical programmers to simulate different subject types for applicable reports
Generate visit dates spaced in a manner consistent with spacing of visit names	Required for testing time-travel and clinical programs that compare/plot visit dates, and determining rules-based logic can be difficult

Prompt Output Validation and Early Accuracy Indicators

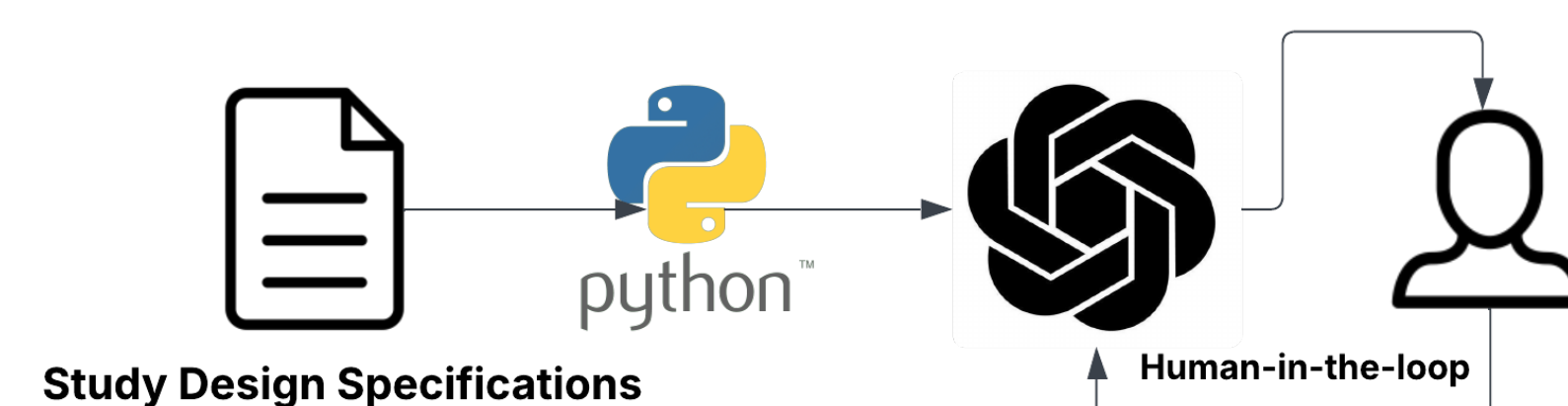
LLM Prompt	Validation Checks	Accuracy Criteria	Accuracy ²
Visit Selection	Format and controlled terminology	Consistency across multiple iterations	>90%
Visit Date Generation	Format and length	Spacing and chronological order	100%
Test Data Generation	Format and controlled terminology	Varies by subject status and form	>90%

1 - In accordance with Takeda's recommended artificial intelligence best practices

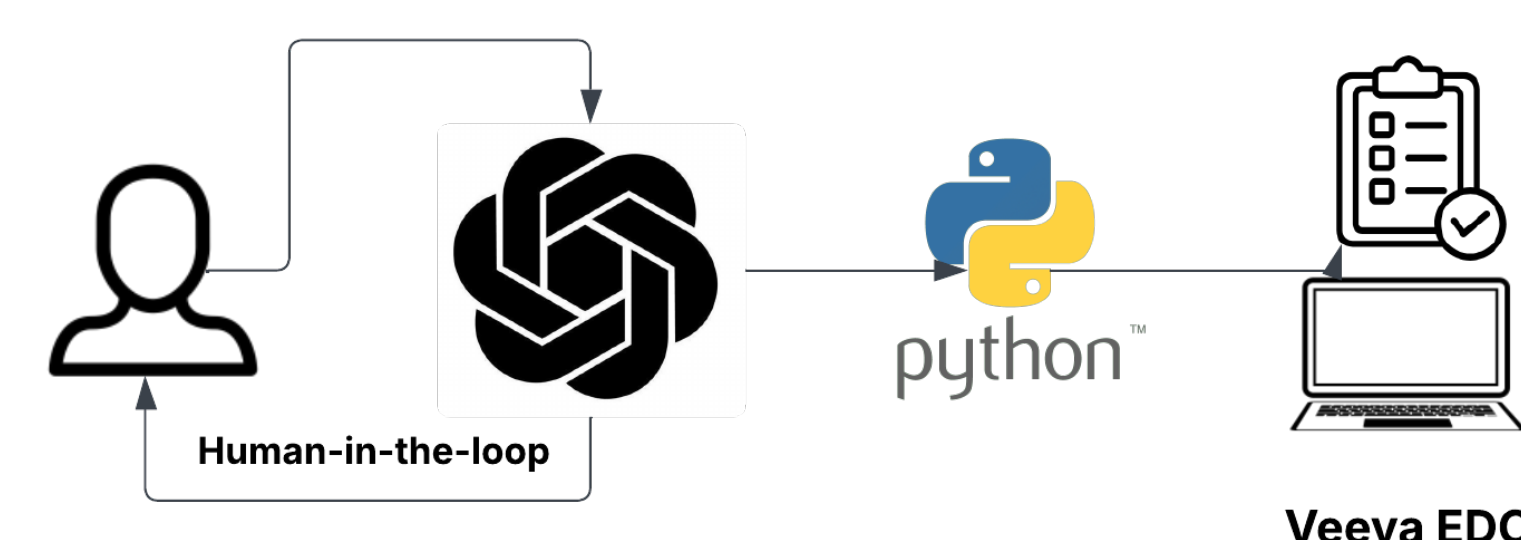
2 - Prompts iteratively tested on 13 trials spanning various therapeutic areas and phases. Accuracy percentages reflect results of testing each trial 5 times per subject status after 15+ previous rounds of testing and refining.

Design

This design balances automation with domain expertise and aligns with the **human-in-the-loop** principle¹. A **Takeda-approved** LLM was selected that does **not** learn from input prompts and is accessed via a **private** endpoint to help protect Takeda's data.



LLM for visit selection: prompted for selection based on subject status, and user evaluates results and re-prompts if deemed necessary



LLM for visit date and test data generation: user may re-prompt if LLM output fails programmed validation checks, and once successful then test data is entered

Prompt Engineering Lessons Learned

- System prompts are critical for consistency
- Strict output formatting improves reliability
- Validation + retry logic reduces error rates
- Concise prompts outperform verbose ones, filter and provide only small chunks of SDS within prompt

Potential Future Directions

- LLM-generated data for edit checks and other EDC rules
- Report-specific test data (i.e. data review listings)
- Explore other LLM strategies such as tool calling and structured extraction from specifications
- Explore leveraging historical data

Conclusion

Combining Python automation with targeted LLM reasoning holds potential to significantly improve the realism, coverage, and efficiency of clinical test data generation. This hybrid approach enables scalable system validation while preserving human oversight.

Acknowledgements

Rajkumar Poosarla, Sona Ashok Kumar, Keshari Gupta, James Lin, Abhilash Sreenivas, Kelsey Sheak, Allison Prey, Amber Hannah, Aviad Adlersberg, Aarti Sanadi