Paper DS08

Optimizing Pinnacle 21 Compliance with Automated Metadata Updates

Karthik Palani, Pfizer Healthcare India Private Limited, Chennai, India

ABSTRACT

Pinnacle 21 plays a pivotal role in validating clinical trial data against CDISC standards but resolving recurring SDTM and ADAM issues and manually editing Define-XML metadata remains time-intensive and error-prone. To address this, we developed a Python-based AI powered automation tool that streamlines SDTM and ADAM issue resolution and metadata generation in P21 workflows. The tool leverages predefined corpora in Excel format to intelligently suggest fixes for commonly encountered SDTM and ADAM findings eliminating redundancy and improving accuracy. It also automates the completion of Define-XML fields in the Variables, Methods, and Comments tabs, ensuring regulatory compliance while reducing manual effort. Built for scalability, the tool integrates seamlessly with existing workflows and enhances consistency across studies. By automating repetitive tasks, it accelerates submission readiness, reduces human error, enables data managers to focus on higher-value review and QC activities. This solution represents a significant step toward modernizing clinical data workflows and improving regulatory submission efficiency.

INTRODUCTION

Artificial intelligence is increasingly reshaping clinical data workflows, offering new opportunities to reduce manual effort, enhance consistency, and accelerate submission-readiness. In the context of SDTM and ADaM metadata management, AI-assisted text generation and automated interpretation of recurring validation issues can significantly reduce the burden on programmers and data managers, who often spend substantial time drafting explanations, harmonizing metadata across spreadsheets, and resolving similar issues across multiple studies. Leveraging these capabilities, the automation system described in this paper integrates a machine-learning model to support intelligent issue interpretation and standardized comment generation within Pinnacle 21 (P21) workflows.

Pinnacle 21 converts clinical data into submission-ready outputs by checking data quality and validating datasets against regulatory standards. With built-in metadata management and continuous data checks, it speeds up timelines, reduces risk, and ensures clean, consistent data throughout the clinical workflow.

Clinical programming often requires repeatedly updating metadata in Excel files and working through many issues in external portals, which can be slow and inconsistent. This automation system reduces that effort by providing a guided interface, automatic spreadsheet updates, batch fixes from CSV files, rule-based and model-assisted text generation, and browser automation that works directly with the Pinnacle 21 tool. Together, these features make the workflow faster, more consistent, and easier to manage.

This paper documents the system's design and implementation as realized in a single Python program, one.py. All functional claims and details in the following sections are grounded in the source code, including function signatures, event wiring from the GUI, and concrete data transformation rules.

AUTOMATION TOOL COMPONENTS

The application is built around one main function, Tkinter interface and connects user actions to the backend. Tasks like Selenium automation, Excel updates, and CSVbased fixes are handled by separate helper functions, while utilities such as resource path and file upload routines support file handling and packaging. Even though everything is in one module, each feature is organized into its own easy to follow function, keeping the workflow clean and straightforward.

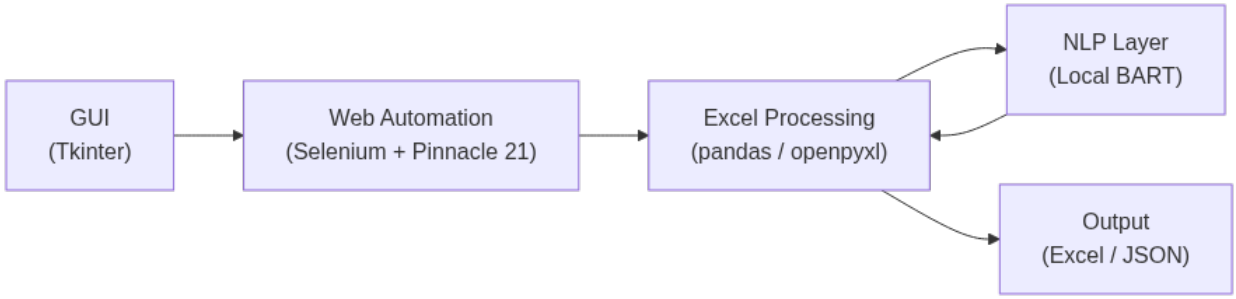


Figure 1: Components of the tool.

AUTOMATION FRAMEWORK OVERVIEW

HANDLING OF ISSUES

In typical study close-out or submission cycles, programmers spend a significant amount of time reviewing P21 findings, fixing issues, and coordinating explanations with study stakeholders. This work is highly repetitive and often tedious because many of the same ISSUE IDs appear across multiple studies with similar corrective actions. To reduce this burden, we developed a Python-based automation process that uses a custom corpus keyed to ISSUE IDs. The corpus stores standardized explanations and predefined logic for resolving common issues. For issues that require narrative text, the tool automatically inserts the appropriate explanation from the corpus.

HANDLING OF COMMON ISSUES LIKE SD, AD and TS

The automated workflow begins by allowing the user to log into Pinnacle 21 through Single Sign-On and select the appropriate study. After navigating to the Issues section, the user initiates processing by opening the first issue. The automation extracts the issue details, including the description, affected dataset, variable information, and relevant metadata. This information is then passed to the backend engine, which checks the curated corpus for a matching ISSUE ID. When a match is found, the system automatically generates the appropriate explanation and populates it into the Define or issue response fields. If no corpus match exists, the user has the option to manually add the required comment. Once the explanation is populated, the tool assists in assigning the issue status such as closing the issue and updating the assignee details.

HANDLING EXTENSIBLE CODELIST ISSUES

For issues related to extensible codelists, the automation first reads the full issue text from the P21 “Issue Details” section, including the list of offending values displayed below the description. When an extensible codelist issue is detected (e.g., *“value not found in ‘Unit’ extensible codelist”*), the system identifies the target codelist and retrieves the corresponding Controlled Terminology (CT) version referenced for that study. CT and Values will be appended in the explanation tab.

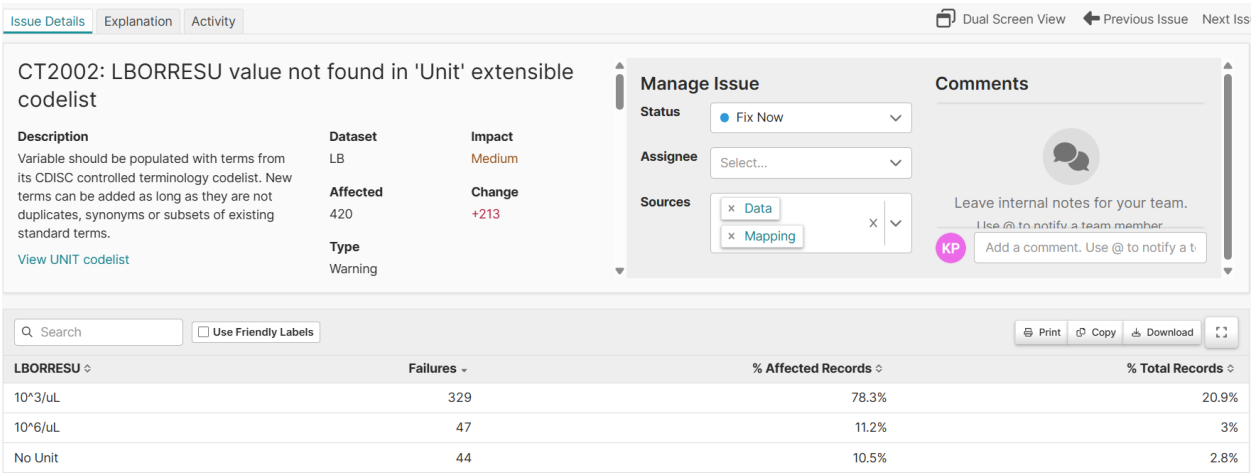


Image 1: Issue details on Pinnacle21.

ISSUES RESOLUTION IN DEFINE USING GenAI

The Issues Fixing tab reads a user-selected CSV and splits certain columns into “A/B” parts to disambiguate targets and values. For example, a “Codelists” update locates a row by ID and alters the Name to include contextual information (e.g., “for ”), while a “Variable” update finds a (Dataset, Variable) row and sets the Comment to a normalized identifier of the form Dataset.Variable.

To standardize issue descriptions, a locally hosted BART-based language model is used to transform raw or inconsistent issue comments into clear, human-readable explanations. Each generated description is derived by classifying the original text and mapping it to a predefined, standardized phrasing, ensuring consistency across studies. The resulting identifiers and descriptions are then automatically recorded in the Study Comments sheet, linking each machine-driven metadata change with transparent and traceable documentation. This approach preserves interpretability while reducing manual effort and variability in issue resolution narratives.

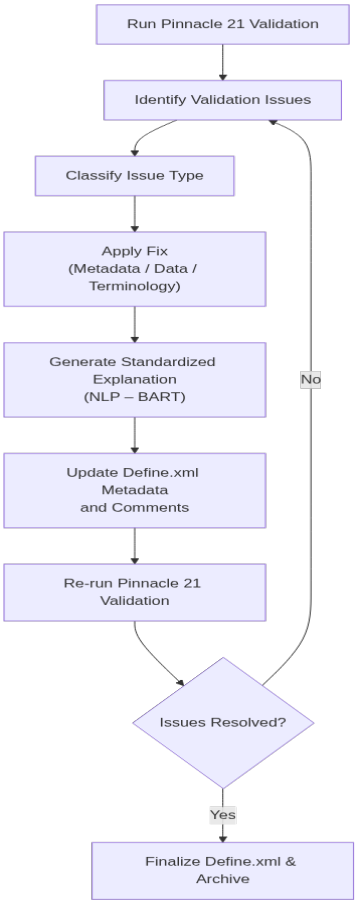


Figure 2: Flow of Issue resolution using GenAI.

CREATION OF METADATA

The creation of metadata within the system is driven primarily by Python's data-processing capabilities, using libraries such as pandas for structured tabular manipulation and NumPy for efficient array-based operations. These packages enable the tool to read, merge, filter, and transform metadata extracted from various sources with high precision. Study-specific Define.xlsx files, organizational standard metadata libraries, and exported Pinnacle 21 user-validation outputs are ingested and processed to construct a unified metadata environment. By programmatically aligning variable attributes, controlled terminology, ValueLevel definitions, and supporting components such as Methods and Comments, the tool ensures that metadata conforms to both study needs and broader company or CDISC expectations.

This automated metadata creation workflow eliminates the inconsistencies often introduced through manual curation across multiple interdependent Excel files. The system harmonizes fields from reference metadata, fills missing information in study-level files, and applies deterministic rules to ensure accuracy in areas such as variable origins, data types, lengths, controlled terminology versioning, and the structure of ValueLevel metadata. The result is a standardized, reproducible metadata output that is ready for downstream Define-XML generation and regulatory submission, improving both quality and efficiency across studies.

DEFINE FROM STUDY/STANDARDS

After uploading the P21 define study/standard spec, the user can select the needed specification and upload it into the P21 system.

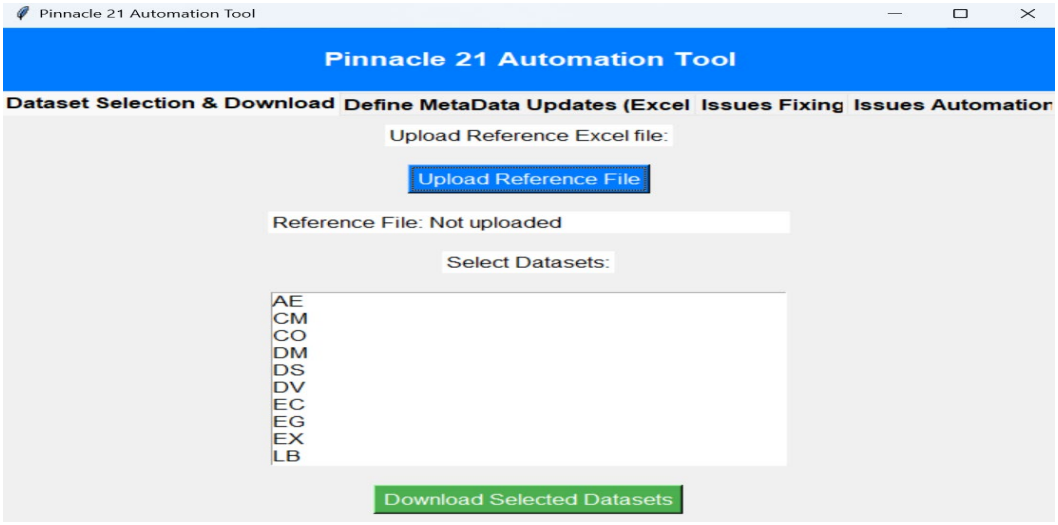


Image 2: Snippet from the tool's Data Selection & Download tab.

Figure 3: Flow of Define .xml update.

DEFINE UPDATE - GENERATED P21

The automation interface helps programmers move through the Define.xml update and issue-resolution steps in a clear and guided way. Users choose the dataset type (SDTM or ADaM) and the correct CDISC standards, then upload both the reference Define file and the study-specific Define file. The system updates all key metadata such as variables, methods, comments, value-level details, and codelists and can also remove datasets that are no longer needed. After processing, it automatically updates the study's Define.xlsx with the recommended fixes. This makes the workflow faster, reduces manual effort, and keeps the Define files consistent and submission-ready.

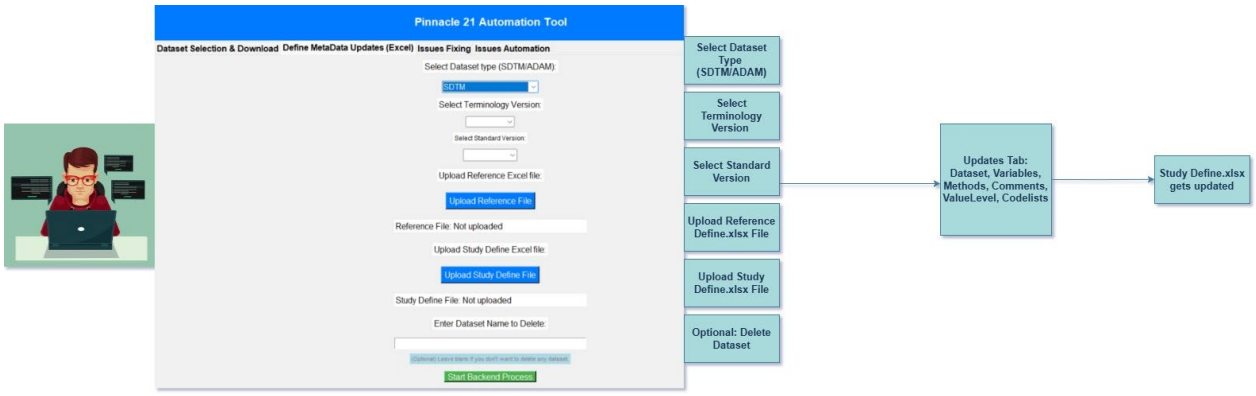
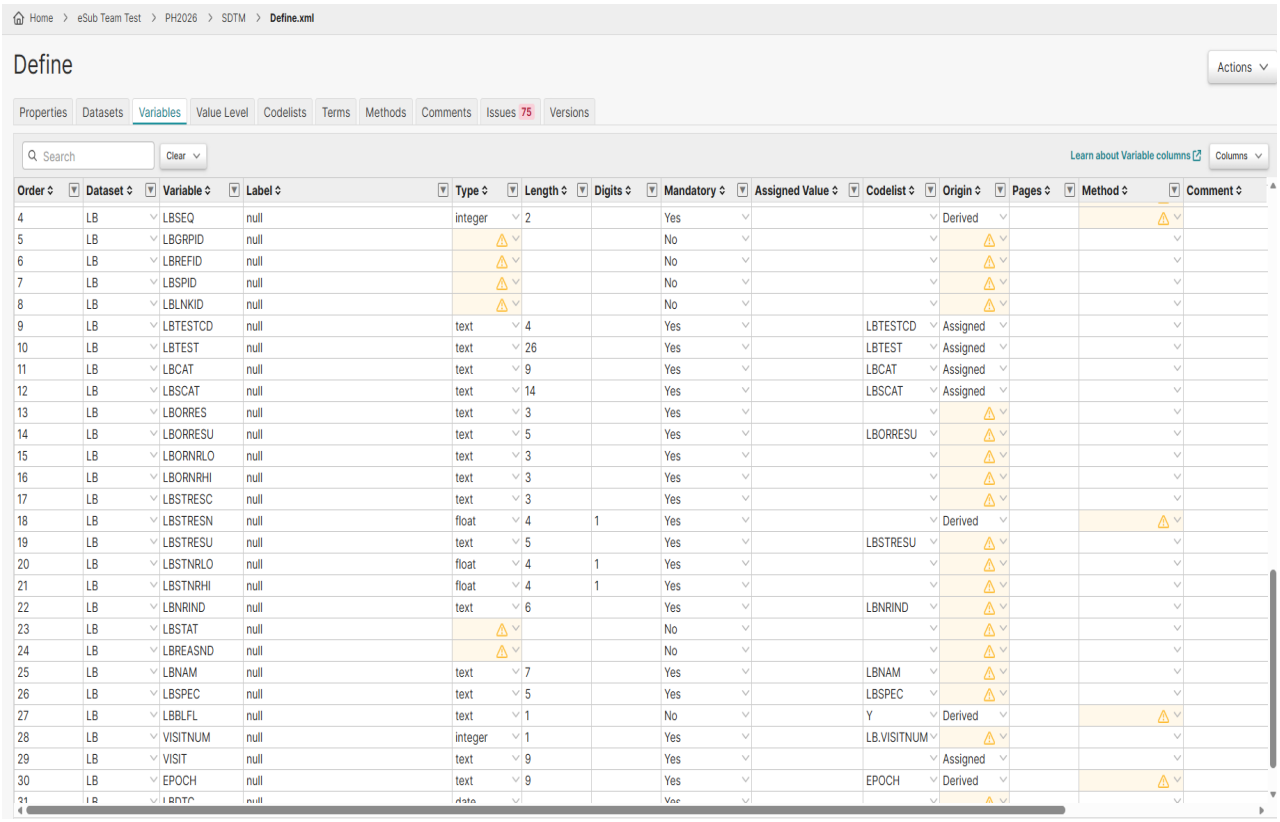
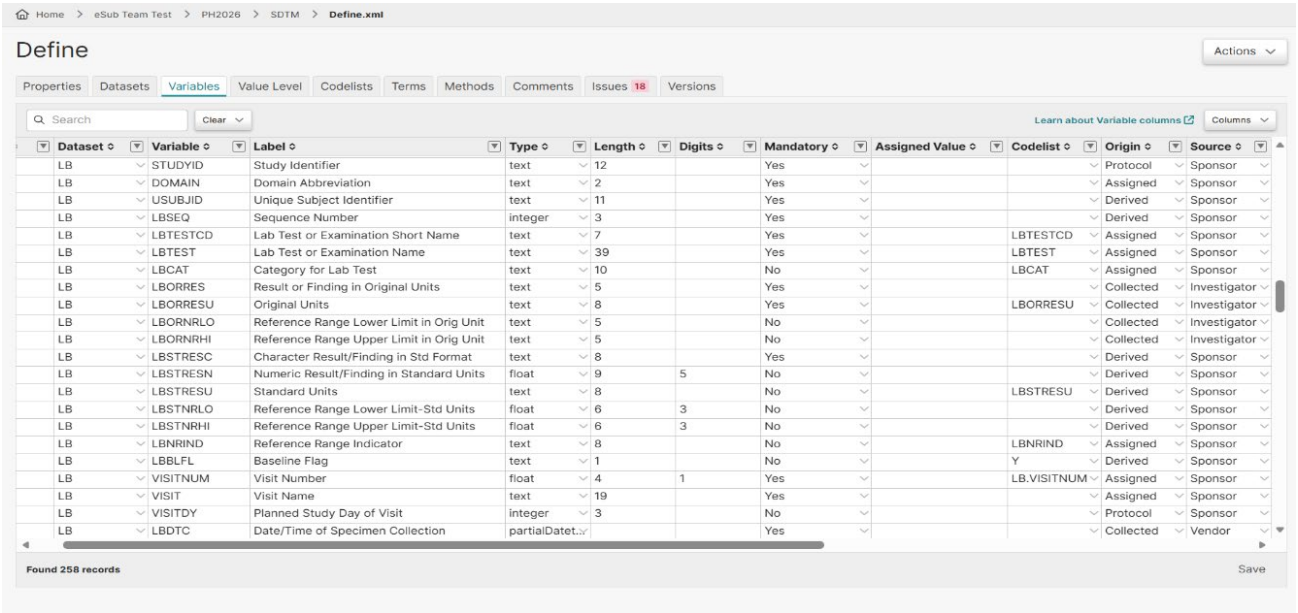


Figure 4: Flow of Define Metadata update with the tool interface.



Order	Dataset	Variable	Label	Type	Length	Digits	Mandatory	Assigned Value	Codelist	Origin	Pages	Method	Comment
4	LB	LBSEQ	null	integer	2		Yes			Derived			
5	LB	LBGRPID	null				No						
6	LB	LBREFID	null				No						
7	LB	LBSPID	null				No						
8	LB	LBLNKID	null				No						
9	LB	LBTESTCD	null	text	4		Yes		LBTESTCD	Assigned			
10	LB	LBTEST	null	text	26		Yes		LBTEST	Assigned			
11	LB	LBCAT	null	text	9		Yes		LBCAT	Assigned			
12	LB	LBSCAT	null	text	14		Yes		LBSCAT	Assigned			
13	LB	LBORRES	null	text	3		Yes						
14	LB	LBORRESU	null	text	5		Yes		LBORRESU				
15	LB	LBORNRLO	null	text	3		Yes						
16	LB	LBORNRHI	null	text	3		Yes						
17	LB	LBSTRESC	null	text	3		Yes						
18	LB	LBSTRESN	null	float	4	1	Yes			Derived			
19	LB	LBSTRESU	null	text	5		Yes		LBSTRESU				
20	LB	LBSTNRLO	null	float	4	1	Yes						
21	LB	LBSTNRHI	null	float	4	1	Yes						
22	LB	LBNRIND	null	text	6		Yes		LBNRIND				
23	LB	LBSTAT	null				No						
24	LB	LBREASND	null				No						
25	LB	LBNAM	null	text	7		Yes		LBNAM				
26	LB	LBSPEC	null	text	5		Yes		LBSPEC				
27	LB	LBBLFL	null	text	1		No		Y	Derived			
28	LB	VISITNUM	null	integer	1		Yes		LB.VISITNUM				
29	LB	VISIT	null	text	9		Yes			Assigned			
30	LB	EPOCH	null	text	9		Yes		EPOCH	Derived			

Image 3(a): Snippet from the Define variable tab showing the issues (75) with the variables.



Dataset	Variable	Label	Type	Length	Digits	Mandatory	Assigned Value	Codelist	Origin	Source
LB	STUDYID	Study Identifier	text	12		Yes			Protocol	Sponsor
LB	DOMAIN	Domain Abbreviation	text	2		Yes			Assigned	Sponsor
LB	USUBJID	Unique Subject Identifier	text	11		Yes			Derived	Sponsor
LB	LBSEQ	Sequence Number	integer	3		Yes			Derived	Sponsor
LB	LBTESTCD	Lab Test or Examination Short Name	text	7		Yes		LBTESTCD	Assigned	Sponsor
LB	LBTEST	Lab Test or Examination Name	text	39		Yes		LBTEST	Assigned	Sponsor
LB	LBCAT	Category for Lab Test	text	10		No		LBCAT	Assigned	Sponsor
LB	LBORRES	Result or Finding in Original Units	text	5		Yes			Collected	Investigator
LB	LBORRESU	Original Units	text	8		Yes		LBORRESU	Collected	Investigator
LB	LBORNRLO	Reference Range Lower Limit in Orig Unit	text	5		No			Collected	Investigator
LB	LBORNRHI	Reference Range Upper Limit in Orig Unit	text	5		No			Collected	Investigator
LB	LBSTRESC	Character Result/Finding in Std Format	text	8		Yes			Derived	Sponsor
LB	LBSTRESN	Numeric Result/Finding in Standard Units	float	9	5	No			Derived	Sponsor
LB	LBSTRESU	Standard Units	text	8		No		LBSTRESU	Derived	Sponsor
LB	LBSTNRLO	Reference Range Lower Limit-Std Units	float	6	3	No			Derived	Sponsor
LB	LBSTNRHI	Reference Range Upper Limit-Std Units	float	6	3	No			Derived	Sponsor
LB	LBNRIND	Reference Range Indicator	text	8		No		LBNRIND	Assigned	Sponsor
LB	LBBLFL	Baseline Flag	text	1		No		Y	Derived	Sponsor
LB	VISITNUM	Visit Number	float	4	1	Yes		LB.VISITNUM	Assigned	Sponsor
LB	VISIT	Visit Name	text	19		Yes			Assigned	Sponsor
LB	VISITDY	Planned Study Day of Visit	integer	3		No			Protocol	Sponsor
LB	LBDTC	Date/Time of Specimen Collection	partialDatet...			Yes			Collected	Vendor

Image 3(b): Snippet from the Define variable tab after the issue is fixed and reduced to 18.

CREATION OF NEW SPECIFICATION USING GenAI

We operationalize an automated pipeline that transforms a Statistical Analysis Plan (SAP) into an analysis-ready ADaM specification using a large language model (LLM) with human-in-the-loop oversight. The SAP is first parsed with layout-aware preprocessing to normalize narrative text and extract structured content such as endpoint definitions, population rules, visit windows, and tabular schedules. The document is then segmented into semantically coherent "chunks" (e.g., Endpoints, Analysis Populations, Methods), which improves downstream inference. An LLM, optionally augmented with retrieval from a program-level knowledge base (prior SAPs/specs and company standards), performs information extraction to identify the required ADaM datasets (e.g., ADSL, BDS domains) and proposes dataset- and variable-level specifications. This includes declaring variable names, labels, types, controlled terminology, origins, and explicit derivation rules with traceability to SDTM inputs and SAP citations. Automated conformance checks (naming, types, key integrity, codelists) are applied prior to human review.

To illustrate, consider a subject-level analysis flag derived from SAP language such as "the primary efficacy analysis will be performed on the intent-to-treat population." From this instruction, the system proposes an ADSL flag variable such as ITTFL (Intent-to-Treat Population Flag) and drafts a derivation that assigns ITTFL='Y' to all randomized subjects meeting the SAP-defined criteria and leaves the flag blank otherwise. The draft specification records the variable's origin (e.g., SDTM DM and disposition data), documents the derivation logic, and links it to the corresponding SAP section defining the analysis population. A subject-matter expert then validates or refines the rule for example, by incorporating protocol deviations or eligibility exclusions after which the finalized specification is exported to Excel or JSON (and optionally define.xml-ready metadata). This approach demonstrates how free-text SAP requirements can be consistently translated into auditable, standards-conformant ADaM specifications while preserving expert oversight.

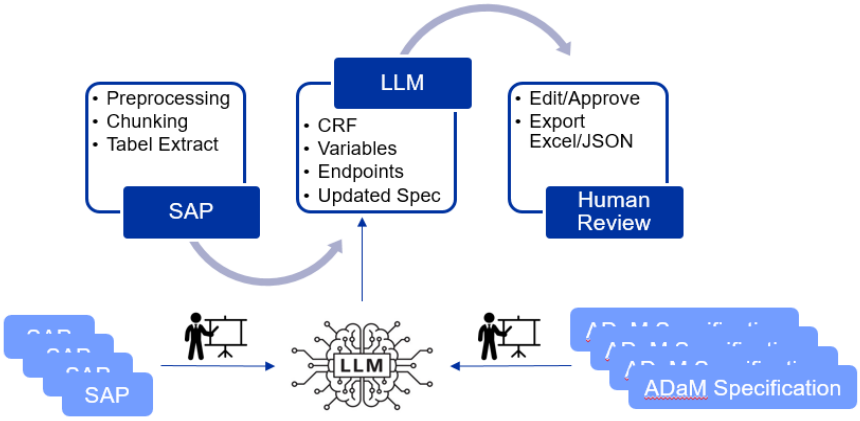


Figure 5: Flow of generating new specification using GenAI.

FUTURE ENHANCEMENTS

Looking ahead, several opportunities exist to expand the system and further support the evolving needs of clinical data management. One key direction involves strengthening the tool's ability to adapt to diverse therapeutic areas, study designs, and global regulatory expectations. As standards continue to mature, future versions of the system could incorporate broader controlled-terminology libraries, therapeutic-area extensions, and emerging regulatory frameworks. Enhancing interoperability with upstream and downstream platforms, statistical programming environments, and submission-authoring tools would also create a more seamless end-to-end workflow for clinical teams.

Another important avenue for enhancement lies in expanding the tool's intelligence and automation capabilities. Future versions may offer more advanced AI-driven assistance, supporting tasks such as automated interpretation of validation findings, and context-aware metadata suggestions. This could help reduce inconsistencies across studies and further streamline quality-control processes. There is also potential to incorporate interactive dashboards, audit-ready metadata traceability, and collaborative features that allow study teams to review and approve metadata changes more transparently. Collectively, these enhancements would position the system as a more comprehensive, adaptable, and future-ready solution for modern clinical data operations.

CONCLUSION

This automation framework enhances the SDTM and ADaM validation lifecycle by accelerating issue identification, standardizing resolution pathways, and improving the precision of Define.xml metadata updates. By reducing manual intervention and mitigating error propagation, it strengthens the overall reliability and traceability of clinical data deliverables. Furthermore, its modular and scalable design allows seamless incorporation of study-specific logic, sponsor conventions, and programmable comments, ensuring adaptability across diverse therapeutic areas and protocol designs. As the framework evolves, its capacity to integrate advanced natural language processing and scalable metadata governance positions it as a future-ready solution capable of supporting modern, high-volume clinical data workflows with improved efficiency, quality, and compliance.

REFERENCES

- https://github.com/phuse-org/phuse-scripts/tree/master/data/sdtm/cdiscpilot01
- https://github.com/phuse-org/phuse-scripts/tree/master/data/adam/cdiscpilot01
- https://huggingface.co/facebook/bart-base
- Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, et al. Transformers: State-of-the-art natural language processing. *EMNLP: System Demonstrations*. 2020.
- Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, et al. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ACL*. 2020.

ACKNOWLEDGMENTS

I would like to thank Surabhi Sundar, Mayur Mishra, and Abhinav Srivastava for their collaboration and support in helping develop this paper. We also extend our gratitude to Periasamy Karuppannan and Priya Venkatesan Iyer for their guidance.

CONTACT INFORMATION

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

Karthik Palani
Pfizer Healthcare India Private Limited
New No.237, Anna Salai, Thousand Lights, Chennai, Tamil Nadu 600014
Karthik.p2@pfizer.com