PharmaSUG China 2025 - Paper AI-177

Cross-module streamlining: database and UI designs to foster ADaM spec-code synchronization

Nana Yang, Tingting Zeng, Yaohui Zhu, Yanan Sui, Mingliang Fan and You Li, BeOne Medicines

ABSTRACT

The traditional way of interacting between ADaM specification and code is to make modifications in Excel file and then notify corresponding programmers to modify ADaM code. This inevitable frequent modification of specifications and codes leads to heavy communication and risk of misalignment. Additionally, it is challenging to track historical versions of both specifications and codes. To effectively address these challenges, we have developed ACIRA (ADaM Code Intelligence for Rapid Analytics) - an AI-driven system designed to visually present real-time modifications to ADaM specifications and facilitate code generation and management. Additionally, ACIRA collaborates with the company's upstream product called SpecMaster (ADaM Specification Management) , which is a tool equipped with functionalities such as precise version control implementation and efficient specification management to enhance workflow integration. By integrating with SpecMaster, ACIRA can access the latest and most accurate specification versions. This integration ensures that the code generation process in ACIRA is always based on the most up-to-date information, enhancing the overall workflow integration and coherence between specification management and code development.

This paper aims to offer a comprehensive technical overview of the version control system and user interface (UI) design for specification modifications and ADaM spec-code synchronization. Leveraging Python's SQLAlchemy and interconnection with SpecMaster through foreign keys, database enables efficient querying, restoration, and storage of specification and code modifications. ACIRA also features a highly streamlined and visually intuitive UI, which allows users to effortlessly monitor and track any specification modification in real-time. The dashboard enables visual tracking of horizontal comparisons of various metrics across different projects. What's more, based on the updated specification content, users can trigger AI code generation with a single click. This integration eliminates manual bottlenecks, enhances data analysis efficiency, and ensures regulatory compliance through unified version control and traceability.

INTRODUCTION

In the dynamic landscape of data analysis and management, especially within the pharmaceutical and clinical research sectors, the ADaM stands as a cornerstone for structuring and analyzing clinical trial data. However, the traditional methodology for interacting between ADaM specifications and code has long been a source of inefficiency and even error, hindering the seamless progress of data-driven projects. The conventional approach to managing ADaM involves making modifications to specifications within an Excel file, followed by notifying programmers to update the corresponding ADaM code. This seemingly straightforward process is fraught with significant drawbacks.

Firstly, the iterative nature of specification and code modifications generates a substantial communication burden. Each alteration necessitates meticulous coordination between specification managers and code developers. For instance, in a large-scale clinical trial project involving multiple stakeholders, a minor change in the ADaM specification might require numerous email exchanges, meetings, and clarifications to ensure that developers or stakeholders more fully understand the update. This back-and-forth communication not only consumes valuable time but also increases the likelihood of misunderstandings, as information can be lost or misinterpreted during the transfer.

Secondly, the risk of misalignment between the specification and code looms large. Since modifications are implemented sequentially, there is a constant danger that the code fails to mirror the updated specification accurately. In a regulatory environment where precision is paramount, such misalignments can lead to incorrect data analysis results. This, in turn, may cause delays in drug development timelines and pose potential compliance risks.

Another critical challenge lies in the arduous task of tracking historical versions of both specifications and code. In an ever-evolving project, with numerous modifications made over time, it's essential to have a clear record of modifications, timestamps, and responsible parties. Without an effective version control mechanism, tracing back the development history becomes a herculean task. It becomes difficult to pinpoint when a particular issue arose, who makes modifications, how to understand the rationale behind past decisions, or track a previous, stable state of the ADaM implementation in case of errors.

To surmount these formidable challenges, our team has engineered ACIRA, an AI-driven system that ushers in a new era of the dynamic collaboration between ADaM specification and code.

THE ADVANCED AUDIT TRAIL VERSION CONTROL SYSTEM OF ACIRA

LEVERAGING SQLALCHEMY

A pivotal component of database design is its advanced version control system, which is built upon a database versioning tool for Python - SQLAlchemy, a powerful Object Relational Mapping (ORM) framework. SQLAlchemy has several features:

1. Not store updates which don't change anything.
2. Support alembic migrations.
3. Revert objects data as well as all object relations at given transaction even if the object has been deleted, transactions can be queried afterwards using SQLAlchemy query syntax.
4. Query for changed records at a given transaction.
5. Query for versions of an entity that modified the given property.
6. Query for transactions, at which entities of a given class changed, history models give access to parent objects relations at any given point in time.

So SQLAlchemy provides a high-level, abstract interface for interacting with databases, enabling efficient querying, restoration, and storage of specification and code modifications. With SQLAlchemy, ACIRA can perform complex database operations with relative ease. For example, it can query the database to retrieve specific versions of the ADaM specifications and code, allowing users to review past states of the project. It also enables the restoration of previous versions if needed, providing a safety net in case of accidental deletions or incorrect modifications. The storage of specification and code modifications are optimized, ensuring that data is stored in a structured and accessible manner, even as the volume of modifications accumulates over time.

EXCLUSIVE TABLES AND FOREIGN KEY INTERCONNECTION

ACIRA utilizes exclusive tables designed to establish a connection with the upstream SpecMaster through foreign keys. These tables serve as a central repository for all information related to ADaM specifications and code. This design allows separation of dataset definition and implementation (similar to template-instance relationship) and cascade deletion and dynamic loading through foreign key associations.

The use of foreign keys creates a relationship between the tables, ensuring that the data is organized in a way that facilitates easy retrieval and analysis. For instance, when a user wants to view the code modifications associated with a particular specification version, the foreign key relationships allow the system to quickly retrieve the relevant data from different tables. This structured data storage and retrieval mechanism not only improves the efficiency of the version control system but also provides a solid foundation for auditing and traceability, which are crucial in regulated industries.

DATABASE DESIGN

The database tables have the following hierarchical classification, first divided into four layers: dataset, variable, paramcd, and codelist. For the upstream SpecMaster, it stores tables of the four layers and their corresponding version tables. For the downstream ACIRA, in addition to the tables of the four layers connected to the upstream SpecMaster via foreign keys (without columns from SpecMaster), there are other auxiliary tables, such as derivation, prompt, quality control and efficancy and cross-module meta integration tables. For a specific study, the adam_setting_id (denoting the study ID) remains consistent across all relevant tables, ensuring unified identification.

Figure 1 is the database tables and connection of upstream SpecMaster and downstream ACIRA

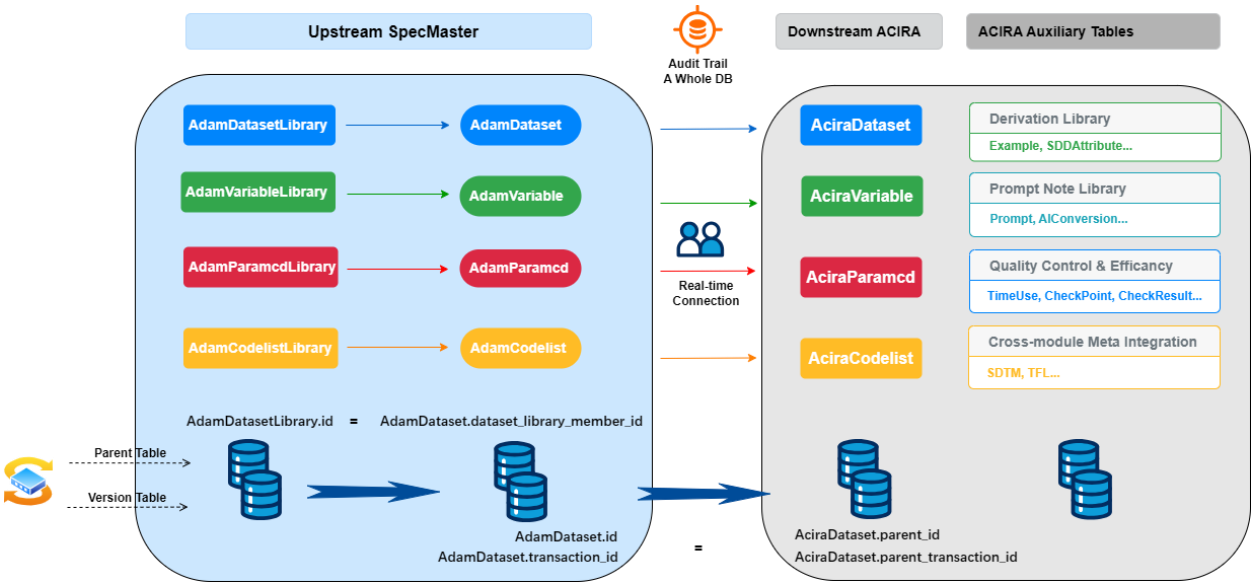


Figure 1. Database tables and connection of upstream SpecMaster and downstream ACIRA

Here is an example for implementing version control and audit trail functionality for ADaM specification and code using SQLAlchemy-Continuum, while leveraging foreign keys to establish relationships between different entities. This mechanism combines version control and foreign key associations to build a full-chain audit trail system from specification definition to code implementation.

One is for audit trail implementation mechanism,

Firstly, make_versioned(user_cls=User) is the core component initialization. It enables version control via SQLAlchemy-Continuum and specifies the User model to track change operators. Automatically generates the following audit components:

1. Transaction Table: Record timestamps, user IDs, and descriptions of all modifications.
2. History Tables: Automatically create history tables for each version-controlled model.
3. Version Records: Automatically saves the previous state to the history table each time data is changed.

Secondly, __versioned__ = {} is for model-Level version control configuration. When no fields are excluded, SQLAlchemy-Continuum enables version tracking for all fields of the model by default. Each time a record is changed, a history table is automatically created to store the previous state.

When modifying entities and committing modifications to the database, Continuum records the operations performed (INSERT, UPDATE, or DELETE) into version entities. These operation types are stored by default in a small integer field named operation_type. The "Operation" class provides convenient constants for these values, as demonstrated in the following example:

The first version's operation type is INSERT

article.versions[0].operation_type == Operation.INSERT

The second version's operation type is UPDATE

article.versions[1].operation_type == Operation.UPDATE

The third version's operation type is DELETE

article.versions[2].operation_type == Operation.DELETE

Thirdly, there are mainly three kinds of key features of audit trail.

1. Automatic history recording: Every create, update, or delete operation on a table automatically stores a change record in the corresponding history table. History tables include all fields from the original table, plus transaction_id (transaction ID), end_transaction_id (transaction ID when the record becomes invalid), and operation_type (operation type: INSERT/UPDATE/DELETE).
2. Transaction-Level Tracking: version_class(Model) returns the history class for the model, used to query records from specific versions.

3. Transaction Table: SQLAlchemy-Continuum automatically creates a transaction table to record timestamps, operators (via the User model), and descriptive information for each change.

The other one is for foreign key relationships and data association design. A comprehensive foreign key strategy enables bidirectional cross-hierarchy associations (SpecMaster-ACIRA), historical traceability via composite keys, efficient cascade operations and lazy loading within SpecMaster, passive multi-level deletions, and AI code traceability through ADaM dataset associations.

Here is an example.

Table 1. Example for data storage in database

AdamDatasetLibrary (upstream SpecMaster Library)

id	created_at	updated_at	created_by	updated_by
UUID-xxx001	XX	XX	XX	XX

name	description	type	structure	key_variables	comment
ADSL	Subject-Level Analysis Dataset	SUBJECT LEVEL ANALYSIS DATASET	One record per subject	STUDYID, USUBJID	XXXX

AdamDataset (upstream SpecMaster)

id	name	dataset_library_member_id	adam_setting_id
UUID-xxx002	ADSL	UUID-xxx001	xxxx

AciraDataset (downstream ACIRA)

id	code_adjust	parent_id	parent_transaction_id
UUID-xxx003	where not missing(ARMCD); test111	UUID-xxx002	7475

AciraDataset Version (downstream ACIRA)

id	code_adjust	parent_id	parent_transaction_id	transaction_id	end_transaction_id	operation_type
UUID-xxx003	where not missing(ARMCD);	UUID-xxx002	7475	7594	7698	0
UUID-xxx003	where not missing(ARMCD); test111	UUID-xxx002	7475	7698		1

HOW TO QUERY AND COMPARE DATA FOR SPECIFICATION CHANGES

Data querying needs to consider three hierarchical levels simultaneously: dataset, variable, and paramcd. This includes data from the upstream SpecMaster and its corresponding version tables, as well as data from the downstream ACIRA and its respective version tables.

The detailed steps are as follows:

Step1: Query data from current

Select current specification information from SpecMaster parent table, including dataset, variable, paramcd, type, label, length, format, codelist, origin, comment, method, time information (updated_at and created_at), etc.

Select current ACIRA metadata from ACIRA parent table, including code, operation type, time information (updated_at and created_at), link information (parent_id, parent_transaction_id, transaction_id). ACIRA metadata can link to SpecMaster metadata

with parent_id/ parant_transation_id).

Step2: Query data from version

Select history metadata from SpecMaster version table and ACIRA version table during base and compare snapshot date. ACIRA codes in the ACIRA version table can also link to history SpecMaster meta. Get the last records which are linked to SpecMaster meta if there're multiple ones. For existing records, without operation_type=2 (belong to delete), get the last record with created_at or updated_at before the snapshot date. For deleted records, keep them only if the deletion datetime is after snapshot (get deletion datetime from transaction table)

Step3: Compare data between base and compare snapshot date

For "Update": keep all the updated records

For "Delete": only for deleting variables/ parameters in existing datasets, without deleting whole data.

For "Add": only for adding variables/ parameters which were not in the initial version of dataset, without showing adding new datasets for the first time.

If sync hasn't been done (no match), highlight as "Not Sync", otherwise as "Sync".

THE INTUITIVE UI FOR SPECIFICATION CHANGES AND CODE SEAMLESS SYNCHRONIZATION AND CONSISTENCY CHECK

The UI of ACIRA is highly interactive, designed to facilitate a seamless workflow from specification modifications to code generation. Based on the updated specification content, users can trigger code generation with a single click. This simplicity in operation eliminates the need for manual intervention at multiple stages, reducing the time and effort required to move from specification updates to code implementation.

The seamless integration between specification management and code generation creates a continuous, real-time workflow. This means that as soon as a specification change is made and approved, the corresponding code can be generated and deployed quickly. By eliminating manual bottlenecks, ACIRA significantly enhances the overall efficiency of the data analysis process, enabling ADaM programmers to complete projects faster and with greater accuracy.

REAL-TIME VISUALIZATION OF SPECIFICATION MODIFICATIONS WITHIN ONE PROJECT

One of ACIRA's core advantages is its intuitive user interface centered on user experience. ACIRA features a highly streamlined and visually intuitive UI, enabling seamless real-time monitoring of specification modifications. The UI presents every ADaM specification modification clearly through color-coding, highlighting, and detailed change logs, allowing users to quickly identify version differences and understand implications without sifting through complex documents. At its core, ACIRA leverages data visualization techniques to provide an intuitive and comprehensive view of specification evolutions over time via interactive timelines and side-by-side comparisons, enhancing development team transparency and empowering ADaM programmer to make prompt, informed decisions by visualizing how modifications impact the overall project.

Figure 2 shows a screenshot of the overview of specification changes. When the "Review Changes" button is clicked, the specific content of specification modifications is presented, divided into four subpages: Dataset, Variable, ValueLevel, and Codelist. Each page uses different colors to distinguish Add, Update, and Delete operations. Records can be easily filtered by "Diff Type", and a search box allows filtering records containing any specified characters. When the "Show Details" slider is toggled on, it highlights the specific content of specification modifications compared to the latest previous version.

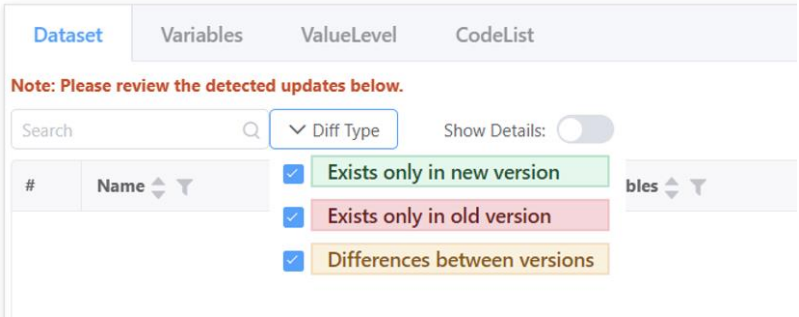


Figure 2. Overview of specification modifications

Figure 3 exemplifies this:

1. The first row shows an example of "Show Details", review can look at highlighted "Test update yy" text.

2. The second to fourth row demonstrates an example of adding PARAMCD at the ValueLevel,
3. The fifth row shows an example of deleting PARAMCD at the ValueLevel.

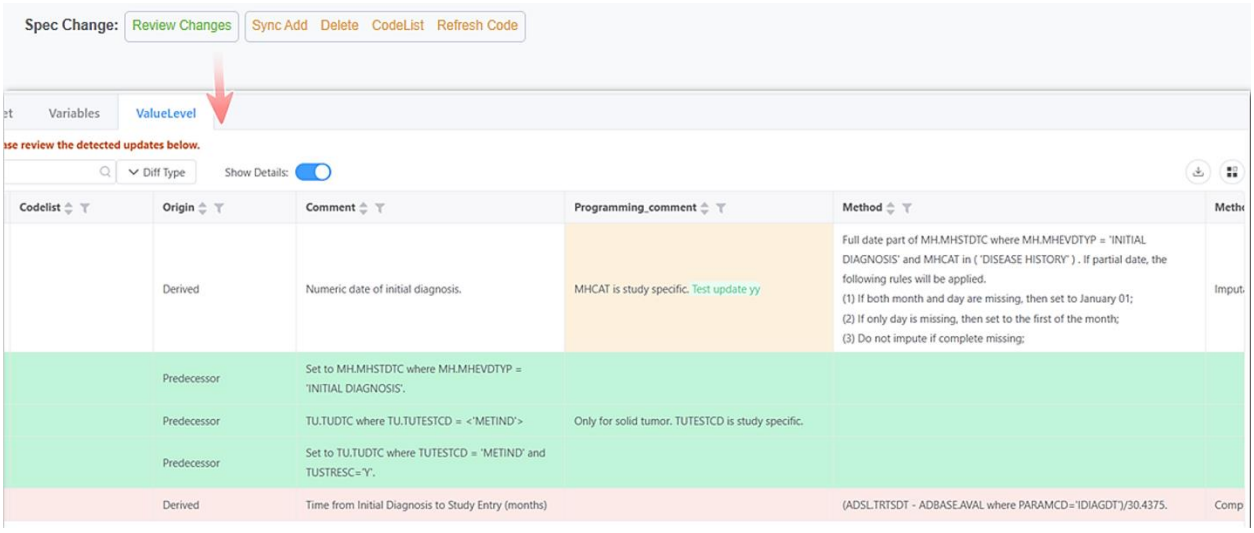


Figure 3. Example of reviewing changes of valuelevel page.

Figure 4 shows an example of ACIRA's actions when additions or deletions occur at the upstream Dataset, Variable, or ValueLevel layers. By clicking "Sync Add", "Delete", or "CodeList" buttons, users can synchronize records of additions or deletions from upstream specifications into the ACIRA part of the same database.

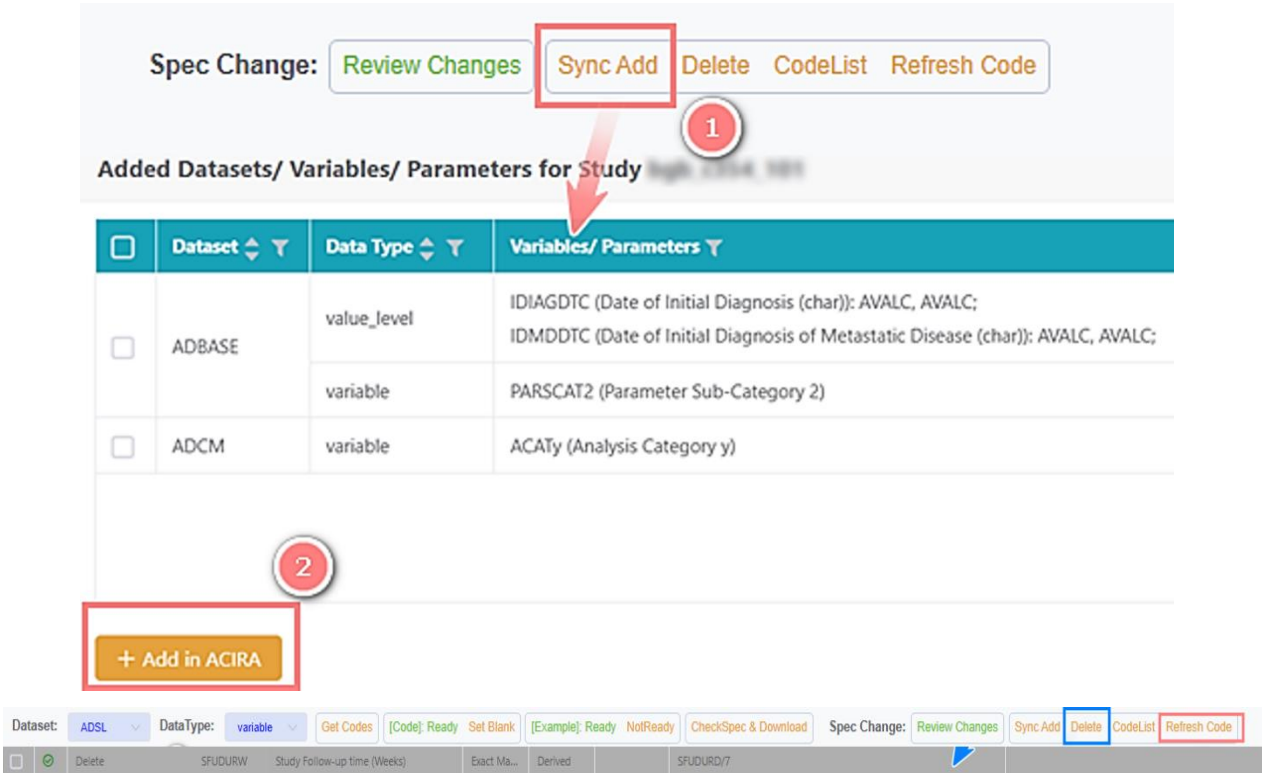


Figure 4. Example of synchronizing add, delete and codelist.

CODE SEAMLESS SYNCHRONIZATION, AUTOMATED GENERATION AND MANAGEMENT

Another core advantage of ACIRA lies in its one-click code synchronization and AI-driven code automation. In contrast to

traditional methods where significant delays often exist between specification modifications and code updates, ACIRA ensures codes are always synchronized with the latest specifications through real-time AI processing. Once ADaM specifications are updated, the system automatically analyzes modifications and user can confirm modifications and generate corresponding AI code, eliminating manual coding efforts, reducing repetitive task time, and minimizing human error—AI algorithms strictly adhere to updated specs to enhance accuracy. This synchronization brings far-reaching value: it avoids misusing incorrect codes due to delays, while shortening coding time and improving efficiency by reducing iterative communication and manual update processes. Particularly in complex projects with numerous code modules, this feature replaces time-consuming, error-prone manual coding with automated precision, ensuring seamless alignment between specifications and generated code.

For scenarios where code refresh is required due to specification modifications, there are two operation modes:

One is batch processing, as shown in Figure 5. and the other operation mode is non-batch processing with confirming one-by-one in figure 6.

Figure 5. Example of all-batch processing.

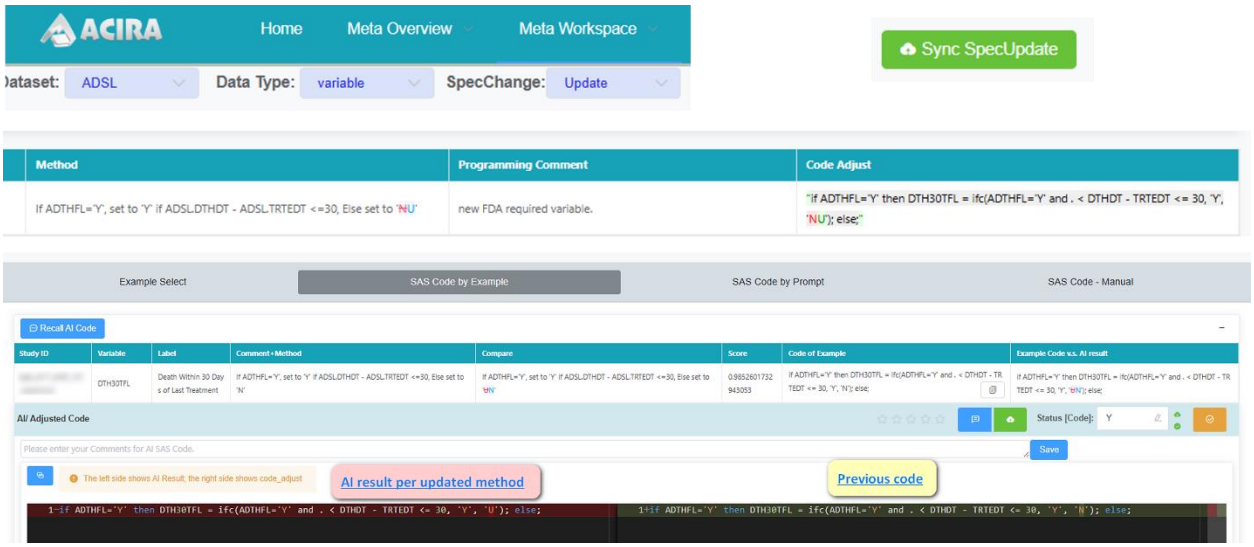


Figure 6. Example of one-by-one batch processing.

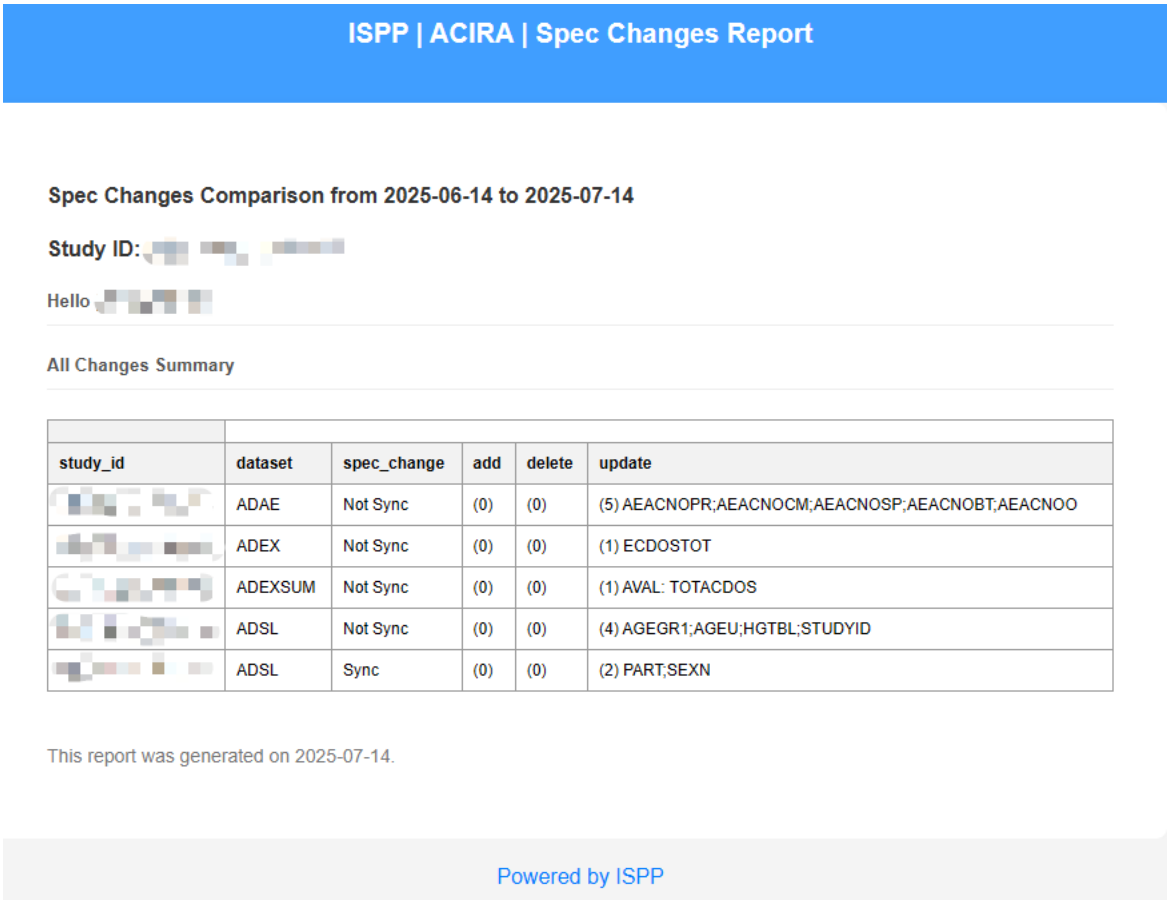
PROACTIVE COMPLIANCE OVERSIGHT: REAL TIME EMAIL ALERT

In the context of ADaM specification and code management, email alerts serve as a critical proactive tool to mitigate compliance risks and streamline workflow efficiency. Traditional manual communication channels often lead to delayed responses to specification changes, increasing the likelihood of code-specification misalignment. Both alert types are generated based on real-time data from ACIRA's version control system, leveraging SQLAlchemy-Continuum to track modification timestamps and user actions. Email alert function addresses this by:

1. Enforcing real-time accountability: Ensuring ADaM programmers are promptly notified of specification modifications to prevent lag in code updates.
2. Enhancing audit trail transparency: Providing documented records of notification timelines for regulatory compliance.
3. Reducing communication bottlenecks: Automating alert distribution to replace manual follow-ups, especially in large-scale projects with multiple stakeholders.

Enable two-way workflow:

1. Bi-Weekly Study Summary Alert: Consolidated Spec-Code Status Update. Provide a periodic overview of specification changes and corresponding code synchronization status per study, facilitating proactive issue resolution. Email Structure Example as following in figure 7:



ISPP | ACIRA | Spec Changes Report

Spec Changes Comparison from 2025-06-14 to 2025-07-14

Study ID: [redacted]

Hello [redacted]

All Changes Summary

study_id	dataset	spec_change	add	delete	update
[redacted]	ADAE	Not Sync	(0)	(0)	(5) AEACNOPR;AEACNOCM;AEACNOSP;AEACNOBT;AEACNOO
[redacted]	ADEX	Not Sync	(0)	(0)	(1) ECDOSTOT
[redacted]	ADEXSUM	Not Sync	(0)	(0)	(1) AVAL: TOTACDOS
[redacted]	ADSL	Not Sync	(0)	(0)	(4) AGEGR1;AGEU;HGTBL;STUDYID
[redacted]	ADSL	Sync	(0)	(0)	(2) PART;SEXN

This report was generated on 2025-07-14.

Powered by ISPP

Figure 7. Example of email alert.

2. Real-Time Spec Modification Alert: Immediate Notification for Critical Changes. Trigger instant notifications when new specification modifications are detected, enabling ADaM programmers to act without delay.

Both ways allow recipients to act on alerts by navigating to the "Spec Change" UI for specification modification review.

COMPARE SPECIFICATION CHANGE AND CODE CONSISTENCY CHECK

The UI for comparing the consistency verification within specification modifications and code is similar to the UI of "Spec Change", but its purpose is to overall review of modifications in both specification and code during selected time period, such as before delivery. It allows reviewers to select different dates freely and conduct a phased, retrospective overview of whether specification modifications and corresponding code updates within a specific period are as expected and double confirm whether the synchronization is correct. This provides a "double safeguard" against operational errors by cross-validating textual specifications with generated code.

Figure 8 shows the UI, where "Old Version" serves as the base date for comparison, and "New Version" serves as the compare date. The timeline selector supports granular date range adjustments, enabling reviewers to focus on critical milestones. After selecting the dates and clicking the "Compare" button, all specification modifications and corresponding codes from the base date to the compare date are displayed at four levels: Dataset, Variable, ValueLevel, and Codelist.

Additionally, a new column "spec_change" is added to record whether users have synchronized the specification modifications in ACIRA system. The UI also includes a batch export function, allowing users to generate detailed excel reports with new version specification content and freely select columns for exporting, which can be attached to quality assurance documentation for audit trails. This comprehensive verification workflow ensures that no specification-code discrepancies go unnoticed throughout the project lifecycle.

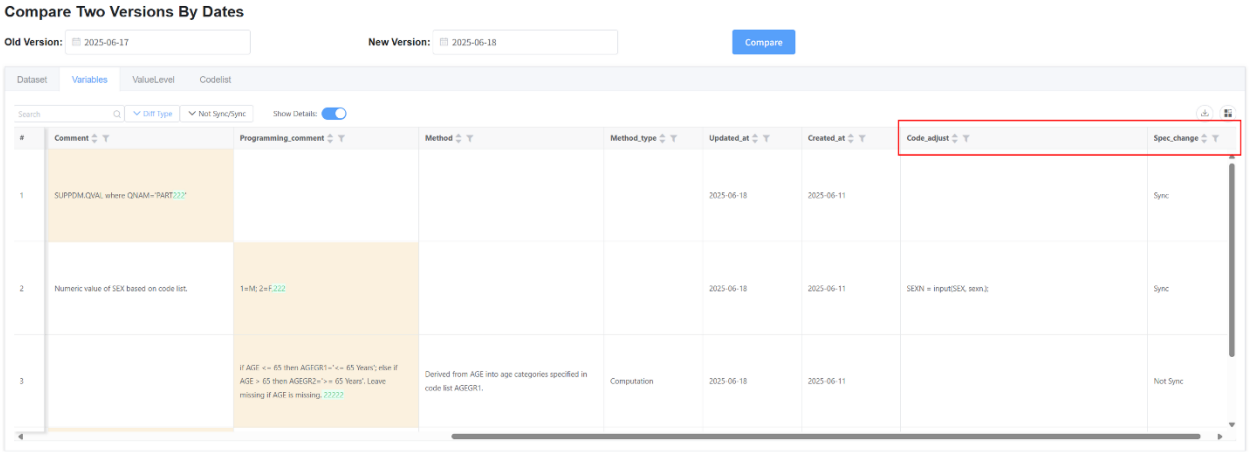


Figure 8. Example of consistency check

THE DASHBOARD: A WINDOW TO CROSS-PROJECT INSIGHTS

Dashboard is different from specification compare/ synchronization in develop console, it empowers users to visually track horizontal comparisons of diverse metrics across projects, offering a comprehensive overview of progress, productivity, and quality. Customizable to prioritize metrics aligned with specific needs, it facilitates real-time monitoring of project timelines, operational efficiency, and quality control workflows. Reviews can oversee the number of modifications by study-version-dataset, and can filter modifications by category, such as total/ open/ closed or row add/ update/ delete. For example, "Spec Change Not Sync/Sync/Tot. #" is project-wise statistics on not sync/sync specification modifications (with percentage counts), top-frequency dataset distributions at the "Dataset" level per project, aggregated summaries categorized by project, dataset, and variable. For selected projects, developers can analyze specification change synchronization between a base date and comparison date.

The system continuously enhances metadata validation checks, enabling developers to identify specification discrepancies during code generation—streamlining communication and quality assurance. leveraging multi-dimensional visibility that allows reviewers to gain holistic cross-project perspectives to aid decision-making. All data is downloadable for advanced analytical workflows.

Figure 10 is an example of dashboard spec change overview. You can customize various visualization perspectives according to specific requirements. When the mouse hovers over the graph, the summary presented by the study can be viewed.

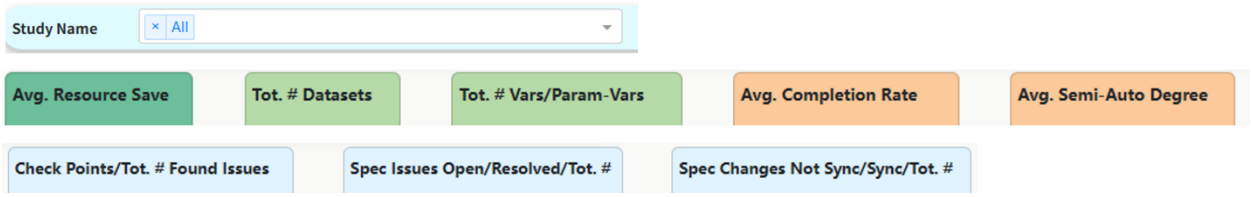
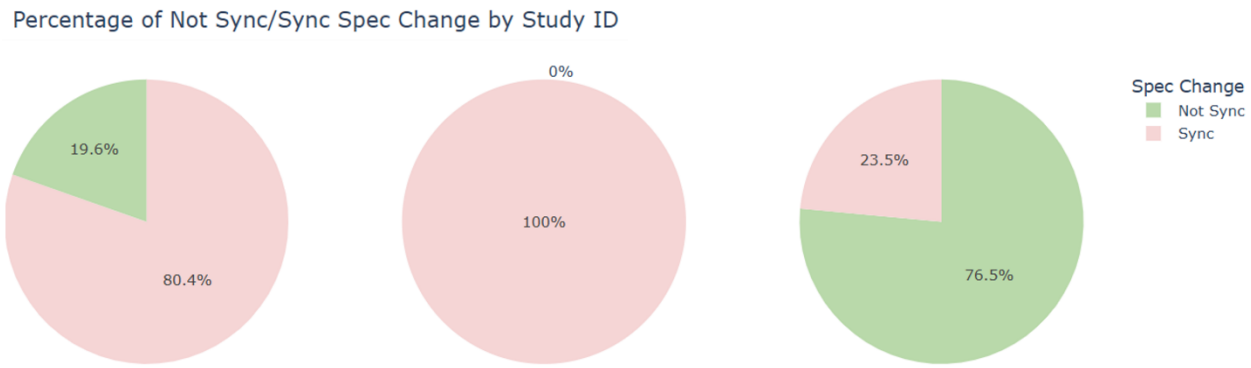


Figure 9. Dashboard items category



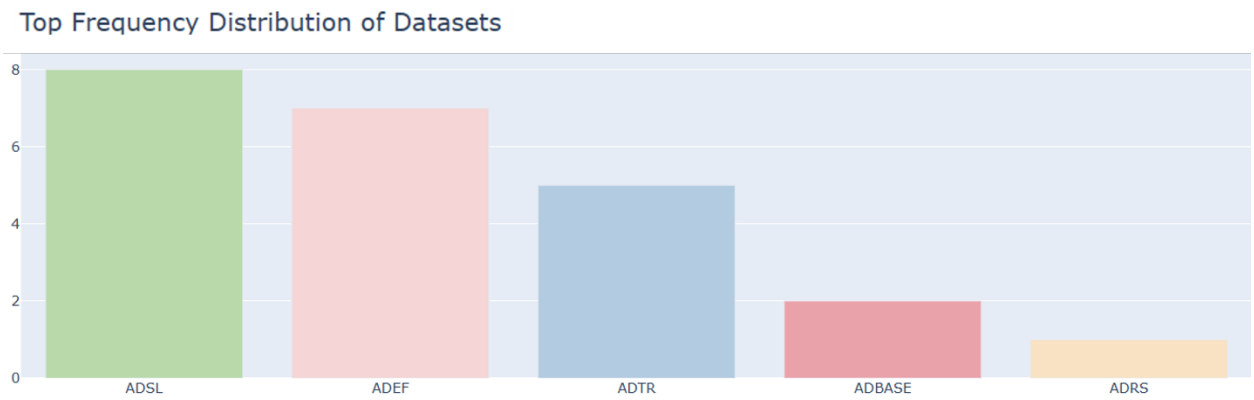


Figure 10. example of dashboard spec change overview

CONCLUSION

ACIRA revolutionizes ADaM specification and code management by addressing traditional inefficiencies through an AI-driven, SQLAlchemy-powered framework, such as communication overload, alignment risks, and version tracking gaps. By integrating with upstream SpecMaster via hierarchical foreign key relationships, ACIRA enables real-time version control, seamless code generation, and cross-validated consistency checks. Its intuitive UI facilitates one-click code synchronization, color-coded change visualization, and phased retrospective reviews, while the dashboard offers cross-project metrics for resource optimization and quality assurance. Leveraging SQLAlchemy-Continuum for transaction-level auditing, ACIRA ensures traceability from specification modifications to AI-generated code, creating a unified workflow that minimizes manual errors and enhances regulatory compliance and audit trail. This integration of advanced database architecture, AI automation, and user-centric design establishes ACIRA as a transformative solution for clinical data management, driving efficiency and accuracy in pharmaceutical research.

REFERENCES

SQLAlchemy-Continuum. SQLAlchemy-Continuum — SQLAlchemy-Continuum 1.3.6 documentation

CONTACT INFORMATION

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

Nana Yang

BeOne Medicines (formerly BeiGene)
nana.yang@beigene.com