PharmaSUG 2023 - Paper SS-117

Consistency Checks Automation across Regulatory Submission Documents in the eCTD M5 folder

Majdoub Haloui, Merck & Co., Inc., North Wales, PA, USA
Hemu Shere, Merck & Co., Inc., Rahway, NJ, USA
Loganathan Ramasamy, MSD, Prague, Czech Republic
Jie Jiang, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

The preparation of the data reviewer's guide (DRG) and the define.xml is a key step in a regulatory submission package for clinical trials. DRGs provide regulatory agency reviewers with additional context and a single point of orientation for SDTM/ADaM datasets submitted as part of eCTD Module 5. The define.xml provides necessary information to describe the submitted datasets and their variables. High quality DRGs and define.xml files are important for a successful regulatory submission to FDA, NMPA, and PMDA agencies.

In this paper, we will provide an overview of a web-based application that leverages Amazon Web Services (AWS) serverless architecture. The application performs consistency checks across DRGs, and SDTM/ADAM defines, which are not typically covered by commercially available tools such as Pinnacle 21 Enterprise. This paper focuses on how these checks are created and an efficient approach for running them.

INTRODUCTION

Quality deliverables are important for successful submissions to regulatory agencies. Currently, there are no commercially available tools to support checking information across DRGs, define.xml and aCRF. Study teams resort to perform this function manually, usually taking days to accurately complete accurately. Due to the tedious nature of the work, such as field-by-field comparison, gaps and mismatches can easily be overlooked by statistical programmers.

Given the importance of quality submissions, our company initiated a project to deliver a system that would aid programmers in providing quality submission documents. The custom solution checks for consistency and correctness across the study data reviewers guide, define.xml and aCRF. The system identifies issues such as documentation errors and inconsistencies across documents, including missing hyperlinks or mismatches across SDTM and ADaM packages. It streamlines the submission process by removing some manual cross-checking, increasing quality and reducing the time and effort within the submission process.

This paper will highlight the need for an automated solution, the checks that are performed, key benefits and rule-based natural language processing (NLP) techniques used for cross-checking against Word, PDF, and XML files. The solution uses innovation and technology to enhance the toolkit for our users and has received very positive feedback from our user community.

NEED FOR ADDITIONAL CONSISTENCY CHECKS

Currently, there are commercially available products that can aid users with creating submission deliverables, but none that provide cross-checking functionality across PDF, Word, or XML files. Pinnacle 21 Enterprise (P21E) is one example, which is a valuable tool for SDTM & ADaM dataset and define.xml validation as well as define.xml creation, but it does not provide document content cross-checking-across aCRF, cSDRG, ADRG, and SDTM/ADaM define.xml.

Additionally, a comprehensive review of submission documents is a tremendous undertaking especially when it is done manually. Certain tasks are difficult and time-consuming to do by hand. This involves checking across various documents, finding the corresponding section in one and comparing it to another. For example, to check that define.xml has correct page number references for CRF annotations, a programmer would need to have both documents open, identify an annotation to check, navigate to the

corresponding aCRF page and verify the annotation exists. Typically, thorough comparisons can take 1 to 2 days and involve hundreds of data check points, and even as detailed and systematic a reviewer may be, findings may be overlooked and missed.

Automating the manual process with a system solution will greatly ease the burden on reviewing resources and eliminate incomplete and inconsistent findings. This will improve the quality of the submission deliverables to regulatory agencies.

TYPES OF CHECKS & DETAILS OF CHECKS

The tool implements four types of checks: SDTM, ADaM, SDTM & ADaM, and All. The SDTM check ensures consistency within SDTM submission documents, while the ADaM check ensures consistency within ADaM submission documents. The SDTM & ADaM check performs consistency checks across both SDTM and ADaM submission documents as listed below. Finally, the All check runs all checks included in the SDTM, ADaM, and SDTM & ADaM check types to provide the most comprehensive evaluation of the submission documents.

The system would benefit these 3 roles:

Role	Description
SDTM Programmer	Perform consistency checks across SDTM documents (cSDRG, aCRF, define.xml)
Analysis & Reporting Programmer	Perform consistency checks across ADaM documents (ADRG, define.xml)
Study Lead	Perform consistency checks for both SDTM and ADaM documents

The following sections provide a detailed description of each check type, including information on its use, the role of the user, and the list of checks performed.

SDTM CHECK TYPE

Use	Role	Checks Performed
This option is used to ensure consistency across SDTM submission documents.	SDTM Programmer	**cSDRG and SDTM define Checks:** • All SDTM subject domains referenced in the cSDRG are present in the SDTM define.xml and have consistent labels. • The largest CRF page number referenced in the SDTM define.xml must be equal to or less than the last page number of the acrf.pdf. • When the origin of a variable in the SDTM define.xml is set to CRF, it must be annotated in the aCRF and the CRF page number referenced in the define must match the CRF page number where the variable is annotated. • All domains listed under the question ‘Were any domains planned, but not submitted because no data were collected?’ in the cSDRG (section 3.1 Overview) are annotated in the CRF but do not exist in SDTM define.xml. • At least one of the columns in the "Subject Domains" table in the cSDRG should be checked for efficacy, safety, or other/baseline subject characteristics.

		• The study name, Protocol Number, and title are consistent across the cSDRG and the define. • The MedDRA, WHODD and LOINC versions are consistent across the cSDRG and the define. **Hyperlink Checks:** • Annotated Case Report Form (aCRF) link is present in the SDTM define.xml bookmark and links to acrf.pdf. • Study Data Reviewer's Guide link is present in the SDTM define.xml bookmark under 'Supplemental Documents' and links to csdrg.pdf. Links to additional supplemental documents.

ADAM CHECK TYPE

Description	Role	Checks Performed
This option is used to ensure consistency across ADaM submission documents.	Analysis & Reporting Programmer	**ADRG and ADaM define Checks:** • All ADaM datasets referred to in Section 5.2 of the ADRG (Analysis Datasets) are present in the ADaM define.xml, and their labels and structure are consistent. • At least one of these columns should be checked for a dataset "Efficacy, Safety, Baseline or other subject characteristics, or PK/PD" in the 'Analysis Datasets' table in the ADRG. • All analysis datasets and descriptions listed in the ADaM Programs table (ADRG Section 7.1) are consistent with the Analysis datasets in section 5.2 Analysis Datasets. • The study name is consistent across the ADRG and the define. • The protocol Number and title are consistent across the ADRG and the define. • MedDRA, WHODD versions are consistent across the ADRG and the define. **Hyperlink Checks:** • Analysis Data Reviewer's Guide link is present in the ADaM define.xml bookmark under 'Supplemental Documents' and links to the adrg.pdf. • Analysis Results Metadata link is present in the ADaM define.xml bookmark under 'Supplemental Documents' and links to analysis-results-metadata.pdf. • Links to other supplemental documents.

SDTM & ADAM CHECK TYPE

Description	Role	Checks Performed
This option is used to ensure consistency across SDTM & ADaM submission documents.	Study Lead	**Checks across cSDRG and ADRG:** • SDTM and SDTM IG versions. • SDTM Controlled Terminology version. • TAUG and Other Standards. • Data cutoff date. • Dataset used for Efficacy and/or Safety. • Pinnacle 21 software version and Validation Engines. **Checks across cSDRG, ADRG, SDTM and ADaM defines:** • Study name. • Protocol Number and title. • MedDRA and WHODD versions.

ALL CHECK TYPE

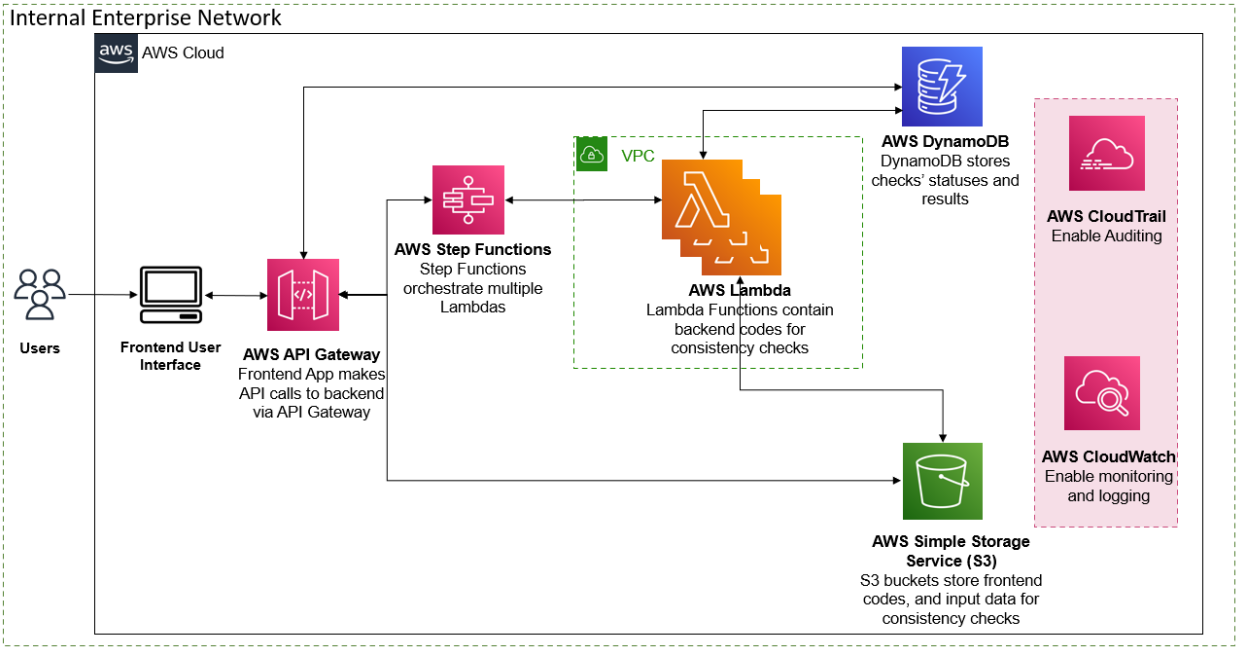
Description	Used by	Checks Performed
This option is used to execute all check types above.	Study Lead	All checks in the above check types.

HIGH-LEVEL ARCHITECTURE

ARCHITECTURE AND TECHNOLOGY

The tool is hosted in an enterprise managed AWS cloud environment and has a multi-tier serverless architecture. Serverless approach allows us to run application without managing servers. The tool uses AWS services for its business logic and hosting of front-end content without the need of servers. Comparing to traditional server-centric infrastructure, serverless architecture eliminates the infrastructure maintenance tasks such as server provisioning and patching, allows for scalability based on the needs, and lowers the costs because of pay-per-use model.

High-level architecture diagram is presented as below:



The tool's architecture consists of three tiers: presentation, logic, and data:

- **Presentation Tier**

  The application's front end is a static web application hosted in Amazon S3 bucket, and it connects to the backend through APIs. The front end is a static website that leverages ReactJS framework.

- **Logic Tier**

  The application's back end uses Amazon Lambda with Amazon API gateway to handle the core business logic. There are multiple Lambda Functions which handle different aspects of the consistency checks. The Lambda Functions are behind Amazon API gateway, and accessible by the front end through API URL paths. The back end also uses Amazon Step Functions as an orchestrator to sequence the Lambda Functions' execution.

- **Data Tier**

  The application uses Amazon S3 bucket to store the files temporarily while running the consistency checks. The application also uses Amazon DynamoDB to store the status of the consistency checks and results.

INFORMATION EXTRACTION

We use rule-based natural language processing (NLP) techniques to accurately extract information from the DRGs and the aCRF documents. As a first step in dealing with the DOCX version of the reviewer guides and define.xml files, we parse the documents using an XML library. Although the content of the reviewer guides is unstructured, the underlying DOCX document is an XML content which makes certain information extraction tasks easier. For example, extracting tables, headers/footers from DOCX are much more reliable than extracting from the PDF version of the reviewer guide.

When it comes to extracting actual content from reviewer guides, the document layout plays a crucial role in the information extraction. We rely on various contextual cues, such as the texts surrounding paragraphs, fixed identifier formats, identifying tables using pre-defined list of columns, and we apply pattern matching using regular expressions to further extract relevant content.

In ADRG/cSDRG, we extract information from document headers, tables, and other sections of the reviewer guides. We compare the information we extract from ADRG/cSDRG against their respective define.xml files. The key information we extract from each document for comparison are listed in table below:

aCRF	DRGs	define.xml
Annotations (domain names and variables)	• Study name from headers and from other sections • Protocol number/title • Subject Domains table (cSDRG) • Analysis Datasets Table (ADRG) • ADaM Programs Table (ADRG) • Information from Study Data Standards and Dictionary Inventory • Data cutoff date	• Bookmark links • Datasets table • External dictionaries from Controlled Terminology • aCRF page numbers (SDTM) • aCRF variables (SDTM)

PROCESSING XML AND PDF SOURCE

There are two steps when extracting information from the XML document: reading the XML document using Python library such as *lxml* and extracting relevant information from the XML data. Most of the information from SDTM/ADaM defines can be extracted directly from the single XML tags. For example, study name, study description and protocol number can be extracted directly from the relevant tags in <GlobalVariables> section in the XML data:

```
<GlobalVariables>
   <StudyName>P011V01MK1111</StudyName>
   <StudyDescription>A Phase III, Multicenter, Randomized Trial ...</StudyDescription>
   <ProtocolName>MK1111-011</ProtocolName>
</GlobalVariables>
```

In other cases, such as tables from DRGs, we parse the xml section <w:tbl>...</w:tbl> to further obtain the table data from the xml source. We use a predefined list of columns to identify the right table, and once identified, we extract the content from <w:tbl>...</w:tbl> section. The figure below shows the flow of the table extraction from the XML source:

Dataset – Dataset Label	Efficacy	Safety	Other	...	...
AE-Adverse Events		X			
CM-Concomitant Medications		X			
DA-Drug Accountability		X			
DM-Demographics			X		
DS-Disposition			X		
EG-ECG Test Results	X				

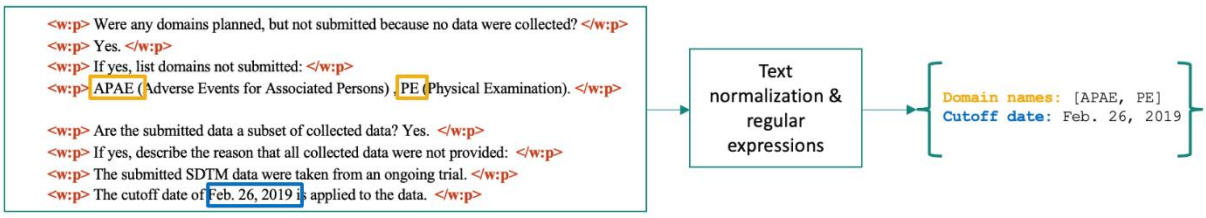
1 Read XML

```
<!-- Subject Domains Table -->
<w:tbl>
    <!-- Header row -->
    <w:tr>
        <w:tc>
            <w:p>
                <w:r>
                    <w:t>Dataset – Dataset Label</w:t>
                </w:r>
            </w:p>
        </w:tc>
        <w:tc>
            <w:p>
                <w:r>
                    <w:t>Efficacy</w:t>
                </w:r>
            </w:p>
        </w:tc>
        <w:tc>
            <w:p>
                <w:r>
                    <w:t>Safety</w:t>
                </w:r>
            </w:p>
        </w:tc>
        <!-- Rest of the column headers -->
        <w:tc></w:tc>
    </w:tr>
    <!-- Remaining rows of the table -->
    <w:tr></w:tr>
    <w:tr></w:tr>
    <w:tr></w:tr>
</w:tbl>
```

2 Extract table In structured format

```
["Dataset-Dataset Label", "Efficacy", "Safety", "Other", ...]
["AE-Adverse Events", "", "X", "", ...]
["CM-Concomitant Medications", "", "X", "", ...]
["DA-Drug Accountability", "", "X", "", ...]
["DM-Demographics", "", "", "X", ...]
["DS-Disposition", "", "", "X", ...]
["EG-ECG Test Results", "X", "", "", ...]
```

In order to extract unstructured data within the XML source, we iterate over XML paragraph elements <w:p>...</w:p> and further apply standard NLP techniques such as text normalization (eliminating variations of quotes, hyphens, and other special characters, removing extra spaces), pattern matching using regular expressions to obtain relevant information.



We follow similar approaches when extracting information from the PDF source such as the aCRF document. We use PyMuPDF Python library to process and extract information from the aCRF document. The only information we extract from the aCRF are annotations in each page from which we extract domain names and variable names.

BENEFITS OF USING THE TOOL

The tool has proven to be a valuable addition to our process as it improves document quality and saves time compared to manual checking. On average, tasks that took 1-2 days per study can now be completed in 5 minutes or less. The tool can be especially useful in situations where multiple studies are included in a single submission (more time saving is realized). This allows programmers to focus on other tasks, which increases productivity and efficiency.

Moreover, the tool reduces the risk of human error and ensures consistent and reliable results every time, especially when dealing with partners and external vendors' deliverables. Manual checking cannot be thoroughly performed every time the partner updates their deliverables due to time constraints, but with this tool, this is no longer an issue.

CONCLUSION

In conclusion, the use of this tool as part of the document review process can greatly enhance the quality and speed of the process while reducing the risk of errors and non-compliance. Only by leveraging innovative technologies such as natural language processing, which enables the transformation of unstructured data into structured data, and the application of text normalization techniques, could such a solution be achieved.

REFERENCES

"Welcome to PyMuPDF — PyMuPDF 1.21.1 documentation". Hosted on GitHub and registered on PyPI. https://pymupdf.readthedocs.io/en/latest/.

"lxml - XML and HTML with Python". Hosted on GitHub and registered on PyPI. https://lxml.de/index.html#support-the-project.

"Serverless on AWS". Serverless Computing – Amazon Web Services.

"AWS Lambda". Serverless Computing - AWS Lambda - Amazon Web Services.

ACKNOWLEDGMENTS

The authors would like to thank Martyna Kołacka for her contribution to the project.

CONTACT INFORMATION

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

Majdoub Haloui
Majdoub.haloui@merck.com

Hemu Shere
hemu.shere@merck.com

Loganathan Ramasamy
loganathan.ramasamy@merck.com

Jie Jiang
jie.jiang2@merck.com