PharmaSUG 2019 - Paper PO-218

A Cloud-based Framework for Exploring Medical Study Data

Peter Schaefer, VCA-Plus, Inc.

ABSTRACT

The e-poster presents the concepts of a cloud-based framework used to implement an application for exploring medical study data. The framework allows to easily integrate the kind of scripts that are used by the FDA in their JumpStart services or scripts that create the type of data analyses and tables, listings, figures (TLFs) as suggested in the white papers published by the PhUSE working group "Standard Analyses & Code Sharing". After a brief introduction, the concepts and benefits of the proposed framework are described focusing on the metadata about the analyses and TLFs and how these metadata can drive the user interface and the output of the application. Then, the underlying cloud-based platform is described which allows for the metadata-driven implementation of the proposed framework. Finally, there are some examples based on the scripts that are available from the PhUSE "Standard Analyses & Code Sharing" working group.

OVERVIEW

A few years ago, the FDA introduced a service to support and improved the review process for drug applications called JumpStart [1]. The scripts that are used by this service were shared with PhUSE to support the joint effort of PhUSE's Standard Analyses & Code Sharing ("SACS") working group. In principle, this allows sponsors to use these scripts to review their datasets before submission, to avoid costly delays in the regulatory review process. However, using the scripts as they are will typically require expensive programming resources and will take time. The proposed cloud-based application based on a framework consisting of an existing platform and application-specific extensions will allow end users to execute and take advantage of the JumpStart analyses. The underlying platform hosts and executes scripts and provides efficient ways to create web browser accessible applications. The proposed extension supports flexibility when creating analysis results by introducing a two-step approach to generating table and graphs. An existing implementation comes with data and model repositories that can be used to share analyses and results across organizations.

FDA'S JUMPSTART SERVICE AND PHUSE WHITE PAPERS

For the benefit of their reviewers, the FDA introduced a service that explores submission data with respect to quality and usability for the standard review tools and that runs a defined set of automated analyses producing tables and figures for reviewers to use.

These analyses are selected to be universal and common across studies and submissions and include analyses of demographics, adverse events, and vital signs. The service is using a number of SAS scripts which create about 40 different tables, listings, and figures ("TLFs") covering exposure, liver labs, adverse events severity and toxicology, demographics, disposition, and a summary based on MEDdra terminology. There are two main steps of the JumpStart service:

1. Access and report on whether the submission data are fit for purpose in the FDA environment.
2. Run automated exploratory safety analyses to support the review process

The table below shows the published list of analyses that are used by the JumpStart service.

Category	Analyses
Demographics	• Demographics Overview • Age Group Counts and Percentages • Age Descriptive Statistics • Sex Counts and Percentages • Race Counts and Percentages

Category	Analyses
	• Ethnicity Counts and Percentages • Country Counts and Percentages • Site ID Counts and Percentages • Country and Site ID Counts and Percentages
AE Severity	• Adverse Events by Arm Greater than 2% • Serious Adverse Events by Arm • Adverse Events by Severity and Arm • Serious Adverse Events by Severity and Arm • Relative Risk • Odds Ratio
AE Toxicity	• Toxicity Grade Summary • Preferred Term Analysis by Toxicity Grade (V1) • Preferred Term Analysis by Toxicity Grade (V2) • Two-Term MedDRA Analysis (V1) • Two-Term MedDRA Analysis
Liver Labs	• Labs Greater than the Upper Limit of Normal • Possible Hy's Law Cases • Maximum Post-Baseline Lab Values Compared to Baseline Lab Values • Maximum AST and ALT versus Maximum TB Lab Test Results per Subject Count • Maximum Lab Test Result compared to Study Days
Exposure	• Percent of Subjects Left on Study Medication by Day of Study • Distribution of Doses by Arm • Dose Descriptive Statistics by Arm and Boxplot of Doses • Planned Arm to Actual Treatment Comparison • Dose Changes During the Study
Disposition	• Disposition Events by Arm for All Subjects • Disposition Events by Arm for Exposed Subjects • Time to Disposition Event by Arm for All Subjects • Time to Disposition Event by Arm for Exposed Subjects
MedDRA at a Glance	• The MedDRA at a Glance Analysis Panel shows the user the adverse events that occurred in the study according to their place in the MedDRA hierarchy and allows them to compare any two arms according to a number of statistics.

The benefit of the JumpStart service for reviewers ranges from better initial understanding of data to pointers to possible areas of deeper analysis and include descriptive analyses for signal detection and hypothesis generation. All in all, this will contribute to improved efficiency in the interaction between reviewers and sponsors and at the end to better and faster reviews.

Note that the JumpStart service is not available to sponsors, though they clearly will benefit from the improvements of the review process. And the FDA took an additional step: In the collaboration between the FDA and the pharmaceutical industry driven by the annual PhUSE Computational Science Symposium –specifically, by PhUSE's Standard Analyses & Code Sharing ("SACS") working group – the scripts that are used by the JumpStart services were published, analyzed, and further documented [3]. With that, everyone can in principle get and run these scripts before submitting data to the FDA and pro-actively address issues that might be discovered by the JumpStart service. But as the PhUSE analysis shows, that this is not a trivial task. Typically, there is some programming work required before the scripts execute without any problems and at least, valuable SAS programming resources will be required.

Another successful approach to standardization of analyses and reports is used by the white paper project of the SACS working group [4]. This project develops white papers that define recommended TLFs for medical trial study reports and submission documents. The intent is to work towards industry standards with respect to analysis and reporting for measurements that are common across clinical trials and across therapeutic areas. Based on these white papers, developers could then create scripts consistent with the recommendations for all to use, improving efficiency and safety signal detection. At this point, it is important to recognize that writing the white papers is only the first step: Another project of the SASC working group takes it one step further: The Code Sharing and Repository ("CSR") project aims at creating and sharing scripts for the analyses and TLFs defined in the white papers and making them available for everyone to use. The list below shows some of the already published white papers.

- Vital Signs, ECGs, Labs - Central Tendency
- Non-Compartmental Pharmacokinetics
- Vital Signs, ECGs, Labs – Outliers and Shifts
- QT/QTc Studies
- Adverse Events
- Demographics, Disposition, and Medications

The overlap between this list and the table of the analyses available in the JumpStart service indicates a dilemma: The are no real standards for data analysis and reporting: In other words, even though CDISC has achieved a rather high level of standardization about how to record and tabulate data, we are still a ways from the same level of standardization for the presentation of data in TLFs.

MAKING STANDARD ANALYSES AVAILABLE TO END USERS

In the previous section, two approaches were described towards standardizing analyses of data from medical studies – the JumpStart service and the SACS working group of PhUSE – and most certainly, there are more of these attempts by other groups. It's typical, that these approaches are following a similar thought process: Define analyses and TLFs that are useful across studies and therapeutic areas, implement scripts that generate the TLFs, and make the scripts available to the public. At the core, these projects are embracing the idea of sharing scripts – and with this idea, they might be shooting too low: Sharing scripts means that two or more programmers will share code. It requires programming skills to benefit from the shared code and an end user will still require the help of a programmer to make use of the share. In what can be called "Solution Sharing", the code or script repository needs to have additional capabilities: Knowledge about what a script needs or supports as input (data and options), what kind of output it generates (list of TLFs or reports), and – most importantly – the capability to execute the scripts. With little extra effort by the programmer of a script, this concept will allow end users to directly use the scripts as a solution to their problems without being dependent on the help of an additional programmer.

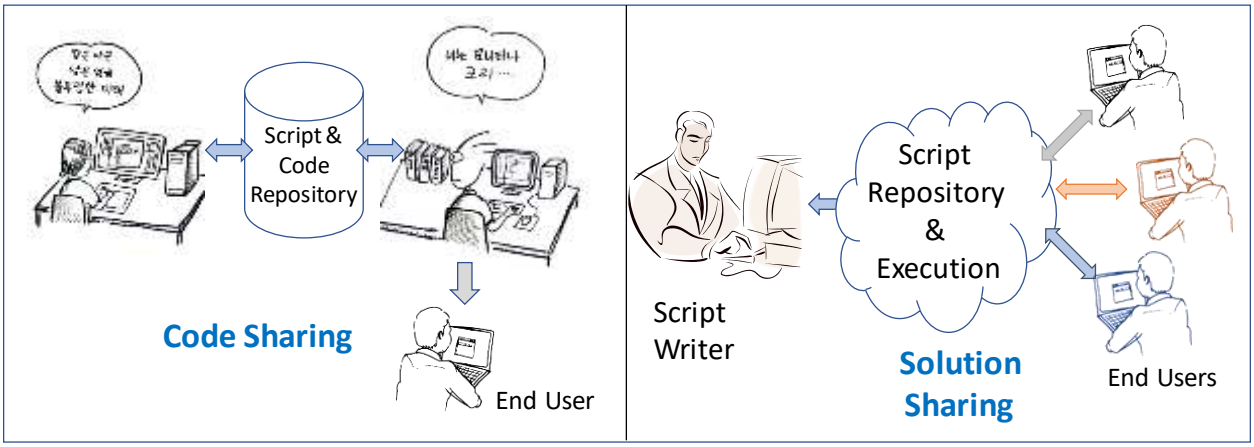

Figure 1. Code Sharing versus Solution Sharing

In fact, the PhUSE code sharing project of the SACS working group that was already mentioned recognizes the value of this additional step and makes some effort to add metadata to the script repository [5].

Following the idea of solution sharing and because the existing scripts and metadata were published by the FDA, the concept of a framework-based application – dubbed with the working title “Trial Data Explorer” or “TDE” – was developed. TDE should allows end users to get the benefit of standardized data exploration – like the JumpStart service – without the need for any programmers to handle shared scripts and to support the end user. Figure 2 shows the framework-based architecture of the proposed application. The generic online platform runs on any cloud infrastructure, generates and serves the user interface, hosts scripts, templates, and metadata, and executes scripts. The platform should also include administrative features and control access rights to the content of the repository. This content – and specifically the scripts – is the part of framework that is application-specific. The metadata about the scripts are used by the platform to generate the user interfaces and to control the execution of scripts. Templates are typically used for specification of TLF content and formatting.

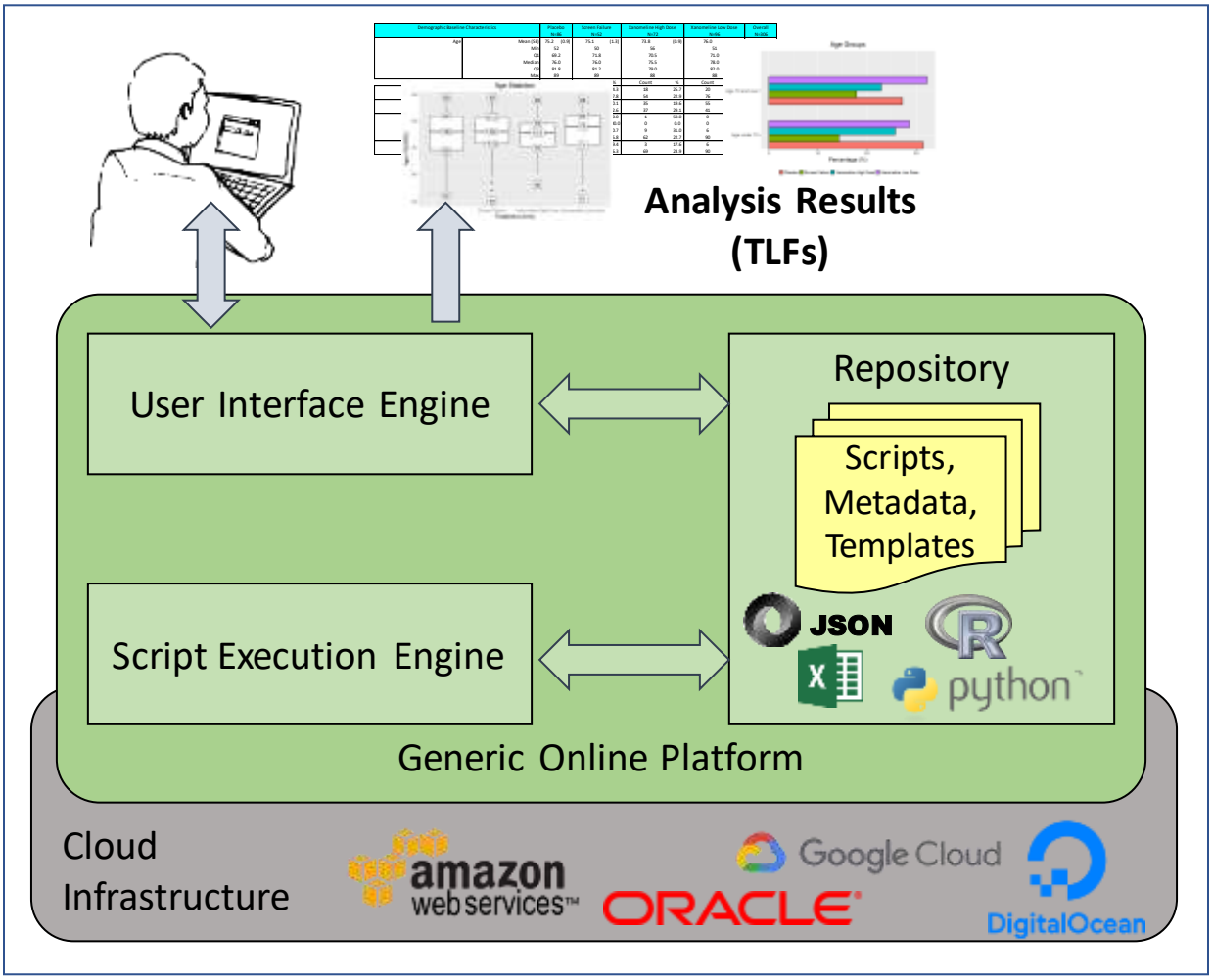

Figure 2. Framework Architecture of the proposed “Trial Data Explorer”

The proposed implementation of TDE and the split into generic and application-specific components provides developers and end users several critical benefits. By far the biggest benefit is that the implementation and maintenance effort for the application will be drastically reduced by using an existing platform. For example, the TDE user interface is generated from metadata by generic platform components. This means that the tedious UI implementation is reduced to capturing the metadata for the

scripts. In addition, the platform comes with integrated features such as script and data repositories and user authentication. Altogether, the use of a powerful generic platform allows programmers to focus their work on the core of the application – the scripts that run the data analysis and create the resulting TLFs. Because the platform is really agnostic to the application-specific scripts, it is very easy to add new features (just by adding one or multiple new scripts) or to modify feature (by replacing an existing script with a new one). This will allow for a start with a true "minimal viable product" for the TDE application and to add scripts as they are required – either from the pool of published scripts used by the JumpStart service or created by the SACS working group or from any other source.

In the remaining sections of this paper, a few implementation details of the proposed application are described together with technologies and methods that enable the implementation.

GENERATING WEBSITE USER INTERFACE FROM METADATA

A core technology of the proposed framework and the underlying platform is the concept of capturing relevant information about the scripts in metadata files. Each script is accompanied by a metadata file that contains information about the required input and how to execute the script – this will be addressed in the next section. Now first, about how to describe that required input for a script. The metadata serve a dual purpose: They are used to generate the user interface and to provide the user input to the script. Both purposes lead to a concept to organize the metadata by user interface controls as they are used on a webpage – controls such as input fields for text or numbers, checkboxes, radio buttons, or file upload control. The specification for each control will need to include a unique identifier, the type of control, and details about how to display the control on a webpage. And of course, there needs to be a way to collect the user input, so that the script can react according to the user's requests. All this information can be captured in several structured data formats, for example, XML (Extensible Markup Language), JSON (JavaScript Object Notation), or YAML (YAML Ain't Markup Language). These alternatives are pretty much interchangeable for the purpose of storing the metadata of scripts: They consist of attribute-value pairs and arrays and provide the required means to serialize data.

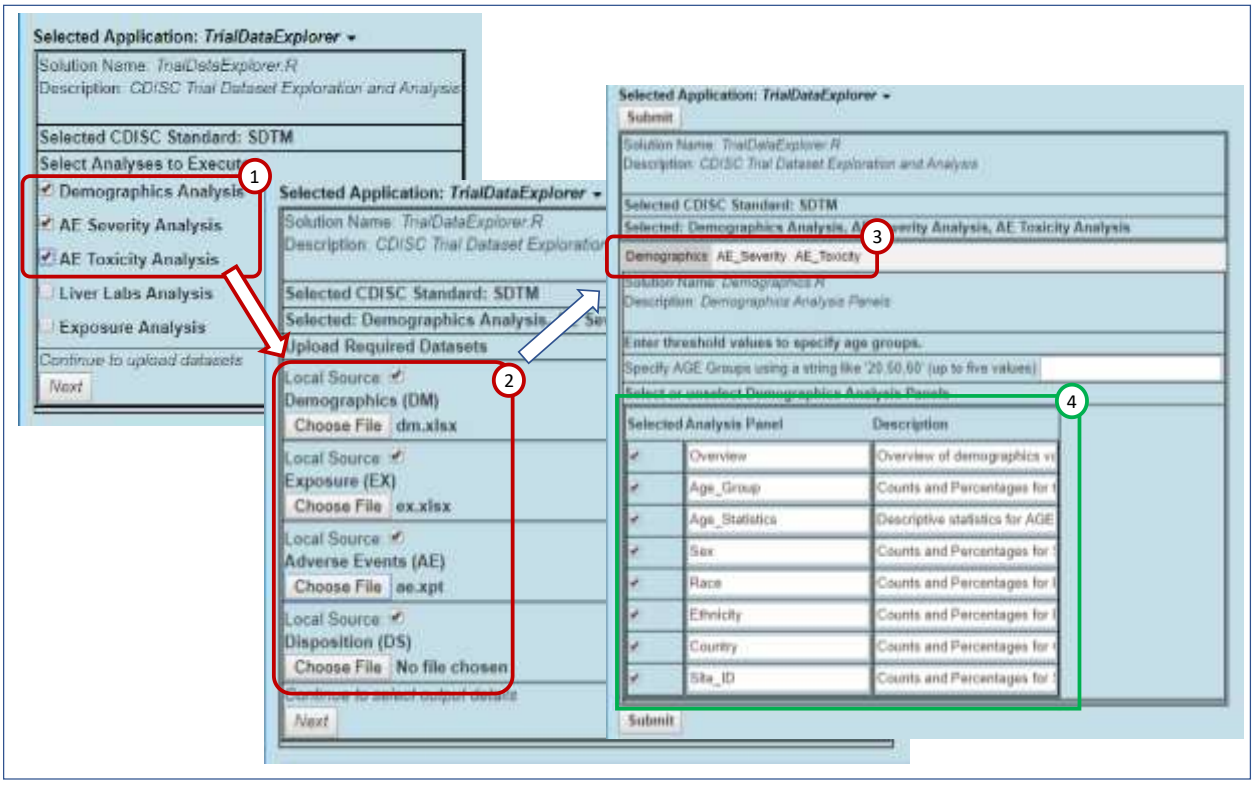

Figure 3: Screenshots of Generated User Interface for Proposed Trial Data Explorer Application

An existing platform implementation based on JavaScript uses JSON because it is easy to use, easy to read and write, and there are existing, mature JavaScript packages that support the concept of UI creation with powerful data binding to JSON structures. In this implementation, the platform presents the user with a webpage showing a list of available applications. After the user selects an application the platform will read the JSON file and generate the user interface in the web browser. The user works with the UI controls – checks checkboxes or radio buttons, uploads files, or enters numbers or text – and submits the webpage for processing to the server. On the server, the platform collects the user input in the JSON structure and starts the execution of the corresponding script. The screenshot below shows a typical user interface as generated from metadata for the TDE application with the steps ① selection of analyses, ② uploading of the required SDTM datasets for the selected analyses, ③ tabs with one entry per selected analysis, and ④ selection of analysis output on the Demographics tab.

EXECUTION OF HOSTED SCRIPTS – AGAIN THE METADATA

The metadata that are used by the platform to generate the user interface are also used to control the execution flow of scripts. In addition to the simple execution of a script using the user's input as described in the previous section, the proposed framework should include additional features for execution and user interface control. The following three concepts have been identified as very useful in real world applications.

1. Support initialization of the script or environment:

   Sometimes a script will need to be initialized before the user can even start working on the input: Good example is the creation of temporary working folders or fetching data from the repository that is displayed, so that the user can select which data she wants to work with. To support this, the metadata for the script should allow to specify whether there is some action to be executed by the platform before the UI is even generated.
   The existing platform implementation mentioned above supports this by a tag in the metadata that is interpreted as a "load action", i.e., as an action that is executed when the script is loaded.

2. Support of dynamic and interactive user experience:

   One typical drawback of many cloud-based applications is the need to successively load multiple webpages from the server.
   The concept to avoid this, are single page websites ("SPA") that will dynamically rewrite the content of the single one webpage that is loaded from the server to give the user an interactive experience that is similar to a desktop application. Essentially, this avoids the interruption of the user interaction caused by loading multiple web pages successively. Though, it might not be possible or even desirable to write more complex applications completely as a SPA, certainly using the SPA concepts and technologies for a more interactive user experience increases the acceptance of an application.
   In the platform implementation, this is achieved by using SPA technologies (JavaScript frameworks, AJAX, or other) and enriching the metadata through tags that are used by the platform to generate interactive UI elements.
   But the user interaction can go even further: Think about the steps in the analysis scripts. After the initial selection of analyses or after uploading the data, the application knows more about the options that the user has. The platform should be able to use this information to present a 'better' user interface to the user. For this end, the platform can use updated metadata with previously collected information and re-submit the metadata and the script for execution. Only this time, some user input has already been processed and the user has made progress towards the final execution of the script.

3. Support of conditional execution and chaining of scripts:

   The third useful concept allows to build applications by connecting scripts to a so-called 'chain of scripts' and supports conditional execution of scripts in this chain. The idea is very straight forward: An application can consist of multiple scripts and the metadata for the application allow to specify a sequence of scripts that the platform will execute in the order they are listed.
   The most obvious requirement regarding execution control for chained scripts is the concept of conditional execution of a script in the chain. The objective is to allow the platform to make decisions

about executing or not executing a script at run time – typically depending on some specific user input or some condition derived from input data. In the current platform implementation, the conditions are limited to settings of the checkbox and radio button controls.
An important second aspect of chaining scripts is the ability of scripts in a chain to share user input and data. Figure 4 shows the two methods that can be used to connect the scripts in a chain:

- Linking of a downstream control to an upstream control: The metadata specification for a downstream script in the chain allows to link to a control of the same type in one on the upstream scripts. In the illustration below, the arrows marked with ① and ② show this type of linkage. The platform uses this information to populate the downstream control with the user input for the upstream control.
- Linking of a file upload control to an upstream output file – which is a special case of linking controls: Instead of linking a file upload control to a file upload control in an upstream script (as shown by ②), the file upload control can be linked to an output file of an upstream script as illustrated by the arrow marked with ③. While linking to an upstream file upload control provides the linked script access to an input file that the user uploads to the application, linking to the output file allows to build chains where a script uses the output of one script as input.

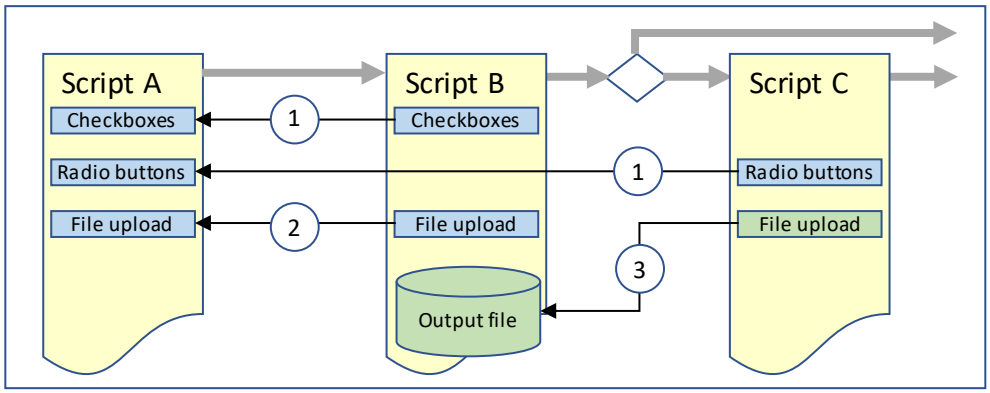

Figure 4. Linking UI Controls in Chained Scripts

FROM DATA TO RESULT FILES IN TWO STEPS

The proposed Trial Data Explorer application has really a very simple workflow as shown in the picture below. As described above the platform supports nicely the 'back and forth' between the application and the user. And because of the existing features of the platform and the concepts supported by the metadata, the implementation can focus very much on the core of the application; i.e., the analysis of the data and the creation of the TLFs.

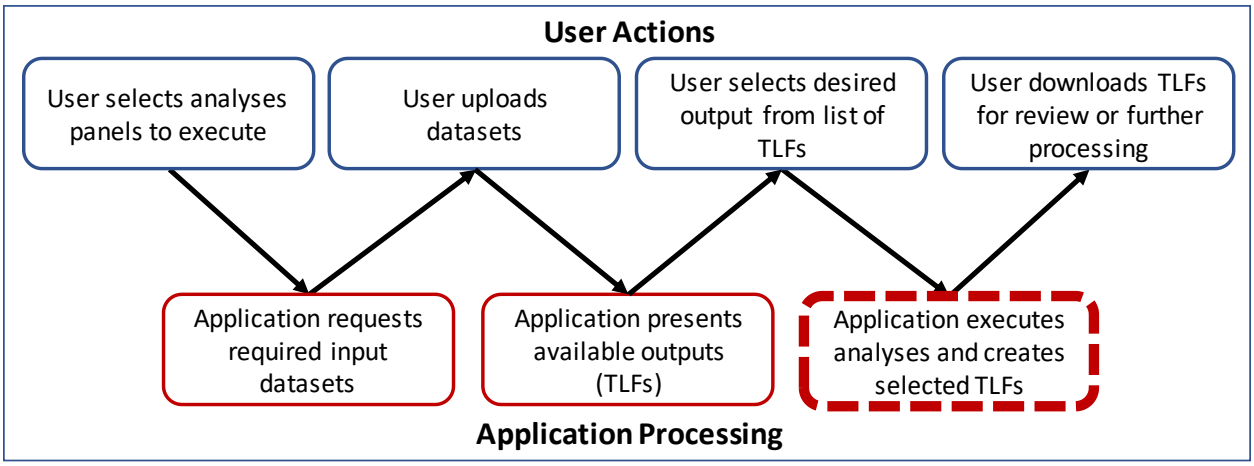

Figure 5. User and Application Interaction

As a basic reporting features, the proposed Trial Data Explorer application will create final output as Excel files (for tables) and image files (such as JPEG or PNG). Open source programs such as R or Python offer several powerful and efficient packages to create these types of output files. But the proposed framework concept for the TDE implementation goes a step beyond this: To support additional flexibility and easy customization of the output objects the concept for output creation is using a two-step approach (Figure 6): In the first step, the analysis scripts will use the input data and templates that describe the content and formatting of the output objects. Driven by the template, the script will generate the data for the output object – but not using the final file format (such as Excel or JPEG) but an abstract file structure independent of the final file format. An existing implementation uses JSON for this intermediate format. Then in a second step, scripts that are agnostic to the specific analysis or content of the table will convert these intermediate output files into the final formatted result files.

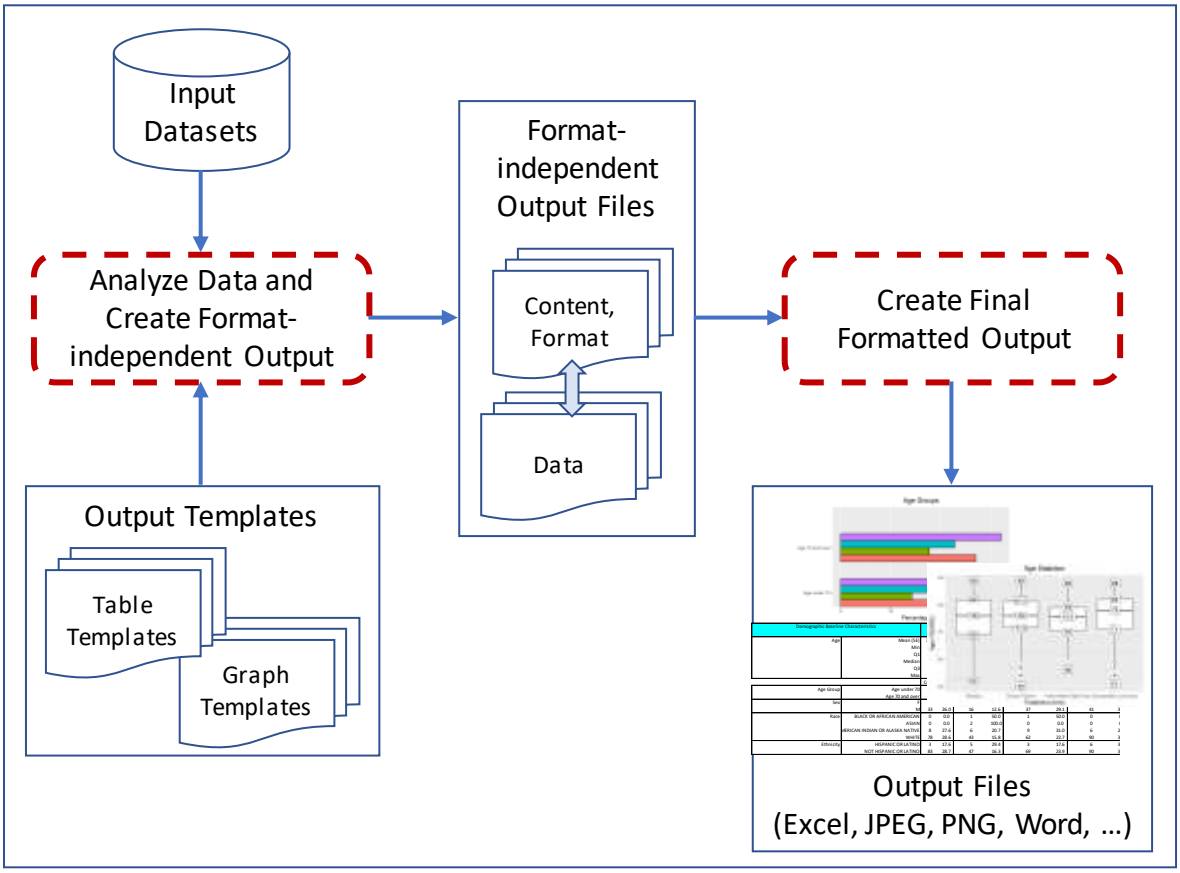

Figure 6. Template-based Two-step Process to Create Result Files

The concept for this two-step approach is based on the following:

- There is one template for each of the result files. The template specifies the type of output, the content (variables, statistical summary, etc.), and the formatting.
- Table templates specify the table to be generated row by row and for each row, every cell of the row is described. The table specification needs to be generic because the number of rows as well as the number of columns in the final output table will depend on the data.
    - A specific 'header row' includes information about required columns. The underlying assumption is that all rows will consist of the same number of columns, though the specification allows for merged cells in a row.
    - A so-called 'dynamic row' specification includes the metadata about when additional rows are required.
    - The intermediate output files for tables contain all information, i.e., formatting and data, in one file.
- Graph templates specify the type of plot (such as time-series, box plot, bar chart, etc.) and the applicable x- and y-axis variables. Additional entries provide plot-type specific formatting information

as required. Contrary to tables, the intermediate output for plots consists of two files per plot, one file containing the plot specification and one containing the data.

Note that the proposed concept works well because the input data are structured and assumed to comply with CDISC standards (SEND, SDTM, or ADaM). The main advantage of this two-step approach is the increased flexibility:

- In general, it is rather easy to make formatting changes because no code changes are required.
- Changes to the final output have limited impact, because only the few scripts that execute the second step need to be modified.
- Adding additional output files could be easier, because it will require the creation of a template and very limited – if any – code changes.

CONCLUSION

The proposed Trial Data Explorer application allows end users to explore clinical or non-clinical trial data based on scripts that are used by the JumpStart service. By using the concept of a generic cloud-based platform, extended through application specific features, no programming expertise is required which makes the TDE application very efficient. A proof of concept implementation shows that the proposed implementation approach is feasible. A completed application would provide the industry with a tool that can help to streamline the review process further. And beyond this one application, applying the same concept to other areas would enable the parties involved in drug development to share tools and analysis results more easily – and as a result after the JumpStart for the review process, continue the race at full speed.

REFERENCES

[1] JumpStarting Drug Review, FDA Website Information for Consumers (Drugs), https://www.fda.gov/Drugs/ResourcesForYou/Consumers/ucm397921.htm, accessed in March 2019.

[2] Get a Jump Start on Clinical Data Analysis and Visualization with Standard Scripts, Jared Slain, Hanming Tu, PhUSE Annual Conference 2016, Paper CS07.

[3] PhUSE White Paper about Scripts: https://www.phuse.eu/documents//working-groups/deliverables/screen-shots-of-the-display-created-using-scripts-contributed-by-the-fda-version-11-23-may-17-11811.pdf , accessed in March 2019.

[4] Analysis and Display White Papers project Website, Project 08, PhUSE Standard Analyses & Code Sharing Working Group, https://www.phusewiki.org/wiki/index.php?title=WG5_Project_08 https://www.phuse.eu/standard-analyses-code-sharing, accessed in March 2019.

[5] PhUSE Working Group “Standard Analyses & Code Sharing”; Project “Script Metadata for Sharing”, https://www.phusewiki.org/wiki/index.php?title=Script_Metadata_for_Sharing, accessed in March 2019.

[6] XML on Wikipedia, https://en.wikipedia.org/wiki/XML, accessed in April 2019

[7] JSON on Wikipedia, https://en.wikipedia.org/wiki/JSON, accessed in April 2019

[8] YAML on Wikipedia, https://en.wikipedia.org/wiki/YAML, accessed in April 2019

ACKNOWLEDGMENTS

Much of this work is inspired by collaboration with colleagues and friends on conferences organized by PhUSE, FDA, SAS, and CDISC. Without the support and encouragement of many people, the proposed framework would not exist, and this paper would not have been written. There is no single person that deserves special credit, but it goes to everyone in these organizations.

CONTACT INFORMATION

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

Peter Schaefer
VCA-Plus, Inc.
pschaefer@vca-plus.com