Paper AS14

Consistency, Speed and Satisfaction: The Impact of TLF Mock Shell Centralisation

Jie Yang, SGS Pharma – Clinical Research, Mechelen, Belgium

ABSTRACT

In clinical trials, generation of tables, listings and figures (TLF) is traditionally guided by study-specific TLF mock shells derived from the statistical analysis plan. While well established, this approach can be resource-intensive and repetitive. To address these challenges, we implemented a central TLF mock shell library improving efficiency of TLF development.

A central TLF mock shell library provides reusable standardised templates promoting consistency in TLF formatting and structure. It covers commonly repeated analysis domains (e.g. demography, adverse events) while remaining flexible to accommodate study-specific endpoints and nuances. This approach streamlines study set-up, reduces programming and validation effort, accelerates timelines and enhances consistency across studies.

In this presentation, we will share our journey of transitioning from study-specific TLF mock shells to a central TLF mock shell library. We will highlight the key development, hurdles and governance strategies. Our experience demonstrates significant improvements in operational efficiency, cost savings and user satisfaction across teams and clients.

INTRODUCTION

TLF mock shells are templates, which define the layout, structure and style of the TLFs shown in clinical documents such as the Clinical Study Report. The shells specify which variables will be displayed in the TLFs, but do not contain any study result data yet.

The industry standard is usually creating TLF mock shells for each study either manually or automatically. While the specific needs of a study are met, the workflow inherently is sequential and resource-intensive. When the volume and complexity of clinical trials increase, this approach becomes suboptimal. To improve efficiency and scalability, we switched to a systematic approach based on a central TLF mock shell library. The library contains standardised templates applicable across multiple studies, regardless the clinical phase and therapeutic area.

This paper outlines how we create study-specific TLF mock shells and the challenges we have faced over years. We then describe the transition to a central TLF mock shell library including its development, maintenance, implementation and associated benefits.

STUDY-SPECIFIC TLF MOCK SHELLS

Study-specific TLF mock shells are usually customized for each individual study. These TLF mock shells contain study specific information like treatment names, definitions on subgroups in footnotes, etc.

PROCESS

The Statistical Analysis Plan (SAP) and the study protocol serve as the primary source for developing study-specific TLF mock shells. The study protocol provides an overview of the planned statistical analyses aligned with study objectives, while more analysis details and methods are described in the SAP. Other available sources used to create TLF mock shells are (annotated) Case Report Forms ([a]CRFs) and Study Data Tabulation Model (SDTM) metadata. CRFs describe how the clinical data is collected and mapped to SDTM variables while SDTM metadata provides more information on the data structure, format and meaning of the data variables.

When preparing study-specific TLF mock shells, statisticians specify details like layout, study populations, treatment groups, subgroup variables, footnotes, etc. The draft TLF mock shells are sent for internal review to assess layout, clarity and programming feasibility. After implementing the review comments, this process will be iterated until no further comments are raised by the internal reviewers. Then the draft TLF mock shells are shared with the client. After external review cycle(s) by the client and finalization of the TLF mock shells, the programmer can start programming. If substantial changes occur after finalization of the TLF mock shells, the TLF mock shells must be amended, resulting in additional review and validation cycles.

The entire process is illustrated in Figure 1 and examples of changes that trigger TLF mock shell updates are shown in Figure 2.

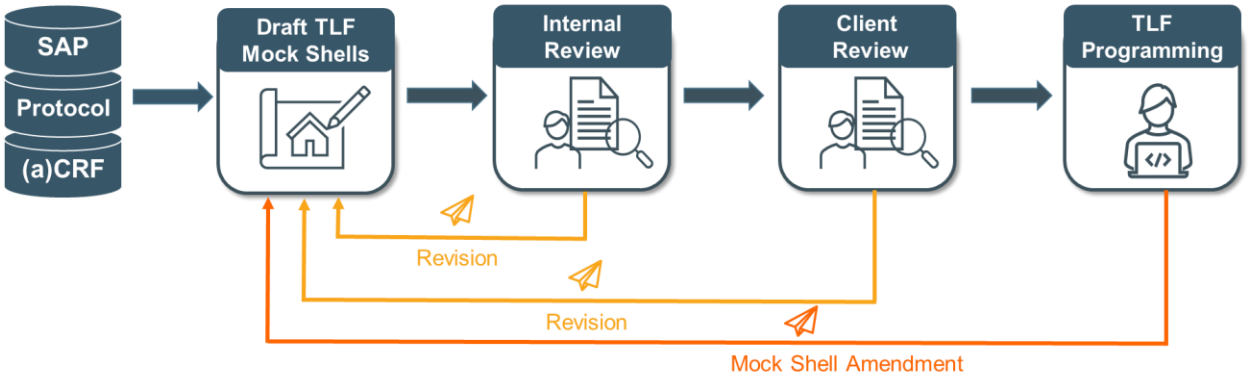

Figure 1: Creation of Study-specific TLF Mock Shells

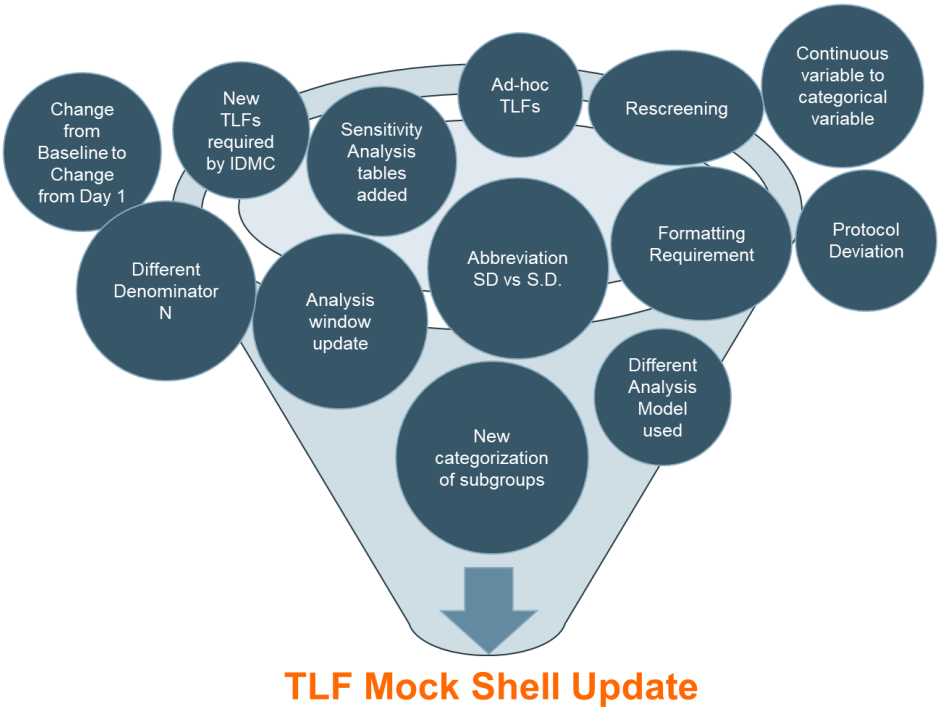

Figure 2: Changes That Lead to TLF Mock Shell Updates

CHALLENGES

Over years, we have faced the following challenges associated with the creation of study-specific TLF mock shells:

- Time consuming and costly:
  - A significant allocation of time and resources is required upfront to prepare the TLF mock shells. When creating study-specific TLF mock shells, all of the TLF mock shells are treated equally, meaning those understood TLF mock shells like adverse event tables, must be recreated and reviewed from scratch. Without pre-defined specifications, repeated discussions take place among statisticians, reviewers and client to align expectations, especially on layout and formatting. Consequently, the programmers can only create the statistical output (TLFs with study results) after the TLF mock shells are finalized, leading to a late programming onset and increased operational costs.
- High maintenance burden:
  - Usually the TLF mock shells are developed early and they require high frequent updates throughout the study lifecycle when the SAP is not finalised or when substantial changes occur. Iterative 'review-revision-validation cycles' increase the maintenance effort substantially, especially for complex studies.
- Risks:
  - The dynamic nature of the 'review-revision-validation cycles' introduces variability. High frequency of TLF mock shells modifications may lead to unforeseen programming errors and quality issues, and potentially affect delivery timelines.
- Redundancy:
  - Many tables, figures, and listings (for safety, demographics, frequency tabulation, descriptive statistics tabulation, etc.) look alike across studies. Recreating the TLF mock shells for these TLFs is redundant, resulting in unnecessary workload with little added value.
- Limited reusability:
  - Study-specific TLF mock shells are usually embedded with study-specific elements (e.g. treatment name, subgroup definitions, visit schedules, tailored footnotes), making them difficult for direct reuse in a new study.
  - Optimal design decisions from repeated discussions and validation cycles often remain in the individual projects.
- Inconsistency:
  - Without standardised specifications, TLFs can look different across studies for the same kind of output. An example is shown in Figure 3. Both tables present the descriptive statistics of laboratory test. However, the different layout and terminology abbreviations make it difficult to cross-compare the statistical results efficiently.

STUDY IDENTIFIER - ANALYSIS VERSION

TABLE 1: DESCRIPTIVE STATISTICS OF LABORATORY TEST ACTUAL VALUES

ANALYSIS SET: SAFETY

LABORATORY CATEGORY: LABORATORY PARAMETER 1 (unit)

ANALYSIS VISIT	TREATMENT A (N=XX)	TREATMENT B (N=XX)
BASELINE		
n	XX	XX
Mean (SD)	XX.X (X.XX)	XX.X (X.XX)
SE	X.XX	X.XX
95% CI	XX.X; XX.X	XX.X; XX.X
Median	XX.X	XX.X
Q1; Q3	XX.X; XX.X	X.X; X.X
Min; Max	XX; XX	XX; XX
VISIT 1		
n	XX	XX
Mean (SD)	XX.X (X.XX)	XX.X (X.XX)
SE	X.XX	X.XX
95% CI	XX.X; XX.X	XX.X; XX.X
Median	XX.X	XX.X
Q1; Q3	XX.X; XX.X	XX.X; XX.X
Min; Max	XX; XX	XX; XX

STUDY IDENTIFIER - ANALYSIS VERSION

TABLE 1: DESCRIPTIVE STATISTICS OF LABORATORY TEST ACTUAL VALUES

SAFETY ANALYSIS SET

LABORATORY CATEGORY: LABORATORY PARAMETER 1 (unit)

	TREATMENT A (N=XX)	TREATMENT B (N=XX)
BASELINE		
n	XX	XX
MEAN (S.E.)	XX.X (X.X)	XX.X (X.X)
S.D.	XX.X	XX.X
MEDIAN	XX.X	XX.X
95% CI	XX; XX	XX; XX
MIN; MAX	XX; XX	XX; XX
Q1; Q3	XX; XX	XX; XX
VISIT 1		
n	XX	XX
MEAN (S.E.)	XX.X (X.X)	XX.X (X.X)
S.D.	XX.X	XX.X
MEDIAN	XX.X	XX.X
95% CI	XX; XX	XX; XX
MIN; MAX	XX; XX	XX; XX
Q1; Q3	XX; XX	XX; XX

Figure 3: Example of Inconsistencies Between Study-specific TLF Mock Shells

CENTRAL TLF MOCK SHELL LIBRARY

In order to improve efficiency without sacrificing the quality, we came up with an alternative approach. This approach starts with a central TLF mock shell library. The central TLF mock shell library is a version-controlled file with a collection of standardised TLF mock shells, accompanied with user instructions, layout and formatting specifications and parameterised elements to fit study-specific needs. It does not contain any study-specific information. For each study, we do not create TLF mock shells based on the templates of the TLF mock shell library. Alternatively, we map the desired TLF output to the specific template of the central TLF mock shell library using reference ID in the SAP.

The rationale of the transition is based on the patterns we observed when we review the past studies:

- Approximately 80% of TLF mock shells are commonly used in the majority of clinical trials and can be standardised regardless of the therapeutic area and clinical phase (for example, demographic and baseline characteristic tables, adverse event tables, summary tables, shift tables, basic figures like line plots and box plots, and basic listings). No 'review-revision-validation cycles' are needed for those TLFs that are already well understood and repeatedly used across studies.
- The other 20% of TLF mock shells are usually linked to novel endpoint, statistical modelling, regulatory related analysis and so on. Some of them can be partially standardised with flexible components.

TRANSITION FROM STUDY-SPECIFIC TLF MOCK SHELLS TO A CENTRAL TLF MOCK SHELL LIBRARY

Creating a central TLF mock shell library rather than separate study-specific TLF mock shells enables more efficient and sustainable development of statistical outputs while maintaining high quality.

DEVELOPMENT OF A CENTRAL TLF MOCK SHELL LIBRARY

We followed the steps below to develop the central TLF mock shell library:

- Engage stakeholders (statisticians, programmers, project coordinators, Analysis Data Model [ADaM] standard experts) during project initiation.
- Review study-specific TLF mock shells from past studies and identify common elements across studies.
- Select representative studies by study design, study phase (I-IV), therapeutic area, and statistical models.
- Define standard requirements for font size, page orientation, alignment, footnotes, etc.
- Iterative development, review and validation of the draft TLF mock shell library until consensus was achieve.
- Finalise the central TLF mock shell library and release it internally.

A senior statistician is assigned as the central TLF mock shell library admin. The library admin is responsible for maintaining and updating the library as needed. User feedback is collected regularly for projects using the library. New TLF templates are incorporated into the library when, during the course of a project, a need for an additional TLF mock shell is identified, for example to accommodate a novel endpoint. Whenever a TLF mock shell is introduced into or withdrawn from the library, it undergoes a formal submission and review process, and version control is applied. Internal trainings are provided to our statisticians and programmers on the proper use of the library.

IMPLEMENTATION OF THE CENTRAL TLF MOCK SHELL LIBRARY

For a new study, the central TLF mock shell library is consulted to determine whether all required TLF mock shells are available in the library based on the information provided in the SAP, study protocol, and a(CRF). Each TLF mock shell has a unique ID in the TLF mock shell library, which is referenced in the SAP appendix to keep traceability.

New TLF mock shells are created only when no suitable TLF mock shells are available in the library (because of novel endpoints/methodologies, specific regulatory requirements, etc.). The programmer can already proceed with programming once the common TLFs are linked to the templates in the library. Meanwhile, focused review will be conducted for the newly created TLF mock shells. The finalized new TLF mock shells will be shared with the programmers later on. For simple clinical phase 1 studies, almost all of them TLFs can be referenced to the TLF mock shells in the library. Figure 4 shows the process of using the central TLF mock shell library.

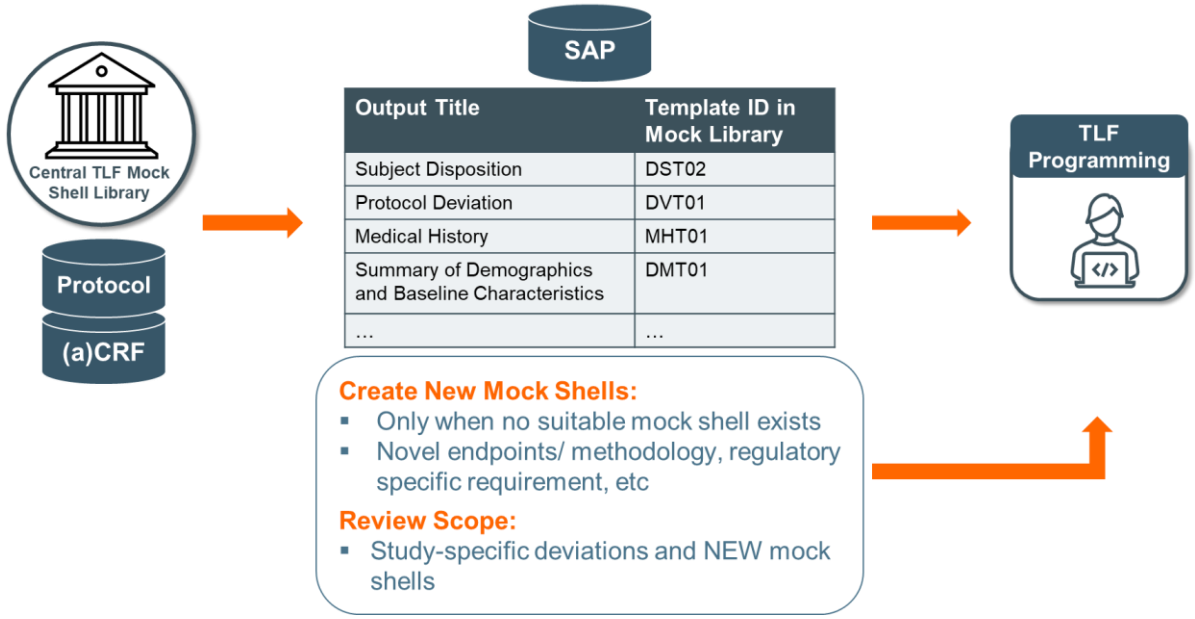

Figure 4: Using the Central TLF Mock Shell Library

BENEFITS OF TLF MOCK SHELL CENTRALISATION

The central TLF mock shell library approach has many advantages:

- Streamlined process and accelerated timelines:
  - Approximately 48 working hours saved by leveraging the central TLF mock shell library instead of building study-specific TLF mock shells. Redundant time spent on layout, formatting and style alignment is eliminated.
  - Thanks to the pre-validated TLF mock shells in the library, the programmer can start generating TLF outputs once the ADaMs datasets are ready and the appropriate TLF mock shells of the library are referenced in the SAP. For complex studies, programmers can continue developing common TLFs while statistician focus on creating new TLF mock shells. This parallel workflow enables a compressed timeline.
- High quality:
  - The central TLF mock shell library is built on top of collective best practices and is governed through development, review and validation process.
  - Standardisation spares more time for statisticians/programmers to focus on analytical quality and study-specific nuances
- Consistency across studies:
  - Improved readability and cross-study comparison.
  - Facilitates study pooling and creation of regulatory submission documents (e.g., Integrated Summary of Safety [ISS]) and Integrated Summary of Effectiveness [ISE]).
- High clients and internal satisfaction:
  - This approach was well received by our clients, who benefit from lowered costs, reduced validation effort and consistent high quality.
  - Statisticians, programmers and validators value the reduction in tedious, repetitive tasks.
- Scalability:
  - This transition enables us to manage more studies simultaneously with fewer resources.
  - Centralized maintenance and version control of the central TLF mock shell library, eliminating the need to manage these activities separately for each study.

KEY TO SUCCESS

As a CRO (Contract Research Organization), we were uniquely positioned to drive the successful implementation of the TLF mock shell library centralisation due to our broad exposure across clients, clinical phases, study designs and therapeutic areas. The experience with regulatory submissions (to the US Food and Drug Administration [FDA], the European Medicines Agency [EMA], etc.), and correct translation of CDISC implementation guides by our standards experts also facilitated the process. The breadth of experience allowed us to identify common and well-understood TLF mock shells patterns for standardisation, while preserving flexibility for study-specific analysis.

Formal change management system and a dedicated library administrator ensure the stability and quality of the library in the long run. Finally, comprehensive trainings and feedback sessions are critical to support correct and consistent use of the library. Unlike the study-specific TLF mock shells, the library is intentionally designed to be clinical phase agnostic and therapeutic area agnostic. Therefore, it requires the users to apply analytical judgement. Our internal trainings focus not only on how to reference the appropriate TLF mock shells in the library, but also on the underlying design principles and expectations. The users are trained to recognise study-specific elements in the SAP and critically assess if additional clarifying information should be added as footnote. Lessons learned sessions are organised for the users to further support the correct usage and evolvement of the library.

CONCLUSION

The adoption of a central TLF mock shell library represents more than a procedural change, it is a conceptual shift with long-term benefits, including reduced budget, streamlined workflows, accelerated timelines, high-quality deliverables and increased satisfaction among clients and staff. Developing a central TLF mock shell library is not the end point, it establishes a foundation for future enhancements, including opportunities for further automation and potential integration with AI-driven processes.

ACKNOWLEDGMENTS

Many thanks to Kristel Gysemans for her help in writing this paper and to Rukun Hinz, José Fernandez, Pieter Coppens and Annelies Van Zeveren for their help creating the abstract, presentation, and this paper.

CONTACT INFORMATION

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

Author Name: Jie Yang
Company: SGS Pharma – Clinical Research

Address: Generaal De Wittelaan 19A bus 5
2800 Mechelen, Belgium
Work Phone: + 32 15 27 32 45
Email: clinicalresearch@sgs.com
Website: www.sgs.com/pharma