State of the SCE

A White Paper by the PhUSE CSS Emerging Trends and Technologies Working Group
Statistical Computing Environments (SCE) Project

This white paper does not necessarily reflect the opinion of the organizations of those who have contributed

Table of Contents

1. Definition of SCE

The Statistical Computing Environment (SCE) begins with receipt or access of data from the CDMS/RDC/EDC system, CROs, or external data vendors and includes development of SDTM or tabulation datasets, ADaM or analysis datasets, tables listings and graphs (TLGs), and submission components for all deliverables whether for CSRs, ad hoc analysis or other regulatory requirements such as annual reports, PSUR, DSUR, CT.GOV, EUDRA.CT, IB, data transparency, etc. The SCE takes these analyses to the point at which they are handed off to the customer, whether that be Medical Writing, Data Safety Monitoring Committee, Drug Safety, etc. Documentation that provides background, specifications and support, such as Protocol and Statistical Analysis Plan, are input into the SCE as a prerequisite to beginning working within the SCE. There could be tools within the SCE to assist with the development and maintenance of such documentation as mock TLGs and programming specifications.

2. Introduction

The SCE project began at the 2015 PhUSE CSS conference and the project team's first goal is to gather information on the current state of SCEs, identify common gaps and share best practices. With more than 21 companies and organizations represented on the project team, we have collected quite a bit of information. In order to collect more structured and consistent information as well as to open it up to the wider PhUSE community, the project team developed a survey that was opened in Q3 of 2015 The survey gathered responses from 43 unique organizations and included large and small pharmaceutical companies and CROs. The data from both the survey and the SCE project team is summarized below in order to achieve our first goal.

3. Executive Summary

In the clinical drug development world, a statistical computing environment is clearly essential for the functions of statistical programming and biostatistics. Because of the natural diversity of organizations and the way they operate, there is a wide variety of tools, processes and technology that SCEs deploy. This in turn reveals gaps in the way organizations implement their SCE, and it also reveals best practices on what works best.

All of the organizations who contributed to the data for this paper use SAS™ as their primary computing platform however no two organizations implement their SCE identically. We have seen that the most important aspects of the analytics infrastructure are ease of use, security, reliability and stability. Standards and metadata have steadily gained acceptance and taken root as the foundation of biometrics work and still continue to evolve in SCEs. Variations in both tools and business processes add to the diversity of SCEs. With the knowledge of the current state of SCEs and their individual best practices and reported gaps this information can be useful to all of the industry to help optimize SCEs.

The SCE Project Team's next goal is to develop user requirements for an ideal SCE based on the lessons learned while researching the State of the SCE white paper and with further input from a broad spectrum of PhUSE members. One possible use for these user requirements would be for industry to work together to define and develop a common set of SCE tools that

could be open source and available for adoption by any organization. Benefits of doing this would be reduced cost for adopting organizations and standardization across industry. The standardization could increase mobility among industry professionals and facilitate partnering among sponsor companies and between sponsor companies and CROs which would enhance scalability and flexibility. Ultimately one could envision a single platform which could be shared with regulatory agencies to facilitate submissions and reviews. By leveling the technical environment across the board, sponsor focus on the science, data and interpretation, and reduce the time and effort to explain to authorities what they did and why their systems and processes are reliable.

4. Summary of Gaps and Best Practices

IT Architecture

- All organizations use SAS
- Half use either Unix or Linux
- Deployment on a single global server is most common
- 57% use a system or software to access their SCE

Remote Access

- VPN and Citrix are most often used
- Connectivity problems are a common issue
- A direct URL access system which is good at handling disruptions with minimal loss in work and ease of reconnection would be a desired feature of any system

Storage Environment/File System

- Directory structures should be standard, as simple as possible, and automated

Archiving

- Leaving files on production systems risks inadvertent changes but moving them off-line is not practical because files need to be accessible. Moving to a separate system with robust controls but which can still be accessed would be ideal.

Validation

- If it is too cumbersome, it can stifle development and increase the risk that users will develop workarounds in order to complete work

Training

- Because people learn differently, it is better to use multiple delivery formats such as instructor lead, recorded, computer based, or read and understand.

Programming Lifecycle/Version Control

- Robust access controls and versioning of programs and outputs, starting after the initial development phase can achieve the desired integrity while minimizing the overhead required which can become excessive if saving and tracking each change during early development.

Dependency Tracking

- The biggest gap identified by those not using commercial systems.

Standardized Code

- Robust standardized code requires dedicated resources.

Data Extraction

- Data extraction and importing should be automated or eliminated by full integration with CDMS or data warehouse to eliminate errors and support data integrity.

Outsourcing

- Robust remote access is needed so that vendors can use sponsors' systems

5. Details

5.1. Technical Infrastructure

IT architecture:

Definition: Architecture typically defines the deployment strategy both for remote and on-site access (Web-based, Client, File System), security model, hardware, and software that are used by your SCE.

Findings:

From the survey, 100% of the 59 respondents recorded that they use SAS™ as the primary statistical analysis software for generating datasets and TLGs for regulatory submission. The platforms for production work include, PC SAS, Windows Server, Linux, Unix, SAS Drug Development (SDD), and HP server. Unix/Linux is the most common platform indicated by 49% (31% Unix and 18% Linux) of respondents, followed by Windows Server at 32%.

Of the server-based users, 73% said that they use a single global server and 27% said that they use multiple servers based on their geographical region or business unit. Access to SAS includes SAS Studio, SAS Enterprise Manager, SAS Display Manager, SDD, Oracle CDC, Unix Emulators, EntimICE DARE and custom software or system. Of the respondents, 22% said they use command line access.

When asked where SAS is hosted, 19% of the respondents said that SAS is hosted externally by a provider and 81% said that they host SAS 'in-house'. The use of SAS Grid is also in use by 12% of the respondents where 86% of those respondents host the SAS Grid internally. All but one respondent reported that they use SAS version 9.x or higher and almost half indicate that they use multiple versions of SAS. The most common versions of SAS are versions 9.2, 9.3, and 9.4 and are approximately equally split in responses. All of the respondents use at least SAS version 9.2. Other non-primary software packages that the respondents use to generate analysis datasets and TLGs were R, S+, SPSS, Excel, and SQL.

When asked about the Statistical Computing Environment interface, 22% indicated that they use a commercially available system or interface such as SAS Drug Development, Entimo, or LSH, 35% use custom software and 43% reported that each component such as SAS, the file system, versioning or tracking is independent or manual. Both those using commercially available systems and those using custom software reported the same level of satisfaction with how user friendly it is (84% somewhat or very user friendly). Those using custom software expressed slightly higher satisfaction with its performance compared to those using commercially available systems (84% vs. 75%).

Among those using commercially available systems, the most popular and common responses were: SDD, Entimo Integrated Clinical Environment, Life Sciences Hub Oracle CDC, or Rational ClearCase from IBM. Strengths reported for these include 21 CFR Part 11 compliance, full audit trail, and robust dependency tracking. Weaknesses reported include complicated & expensive interaction with other tools, overhead of access controls, and SDD's poor compatibility with other statistical analysis systems such as R.

Custom software can vary widely but we collected descriptive information from respondents which included functioning as a wrapper around SAS programs, use of SAS stored processes, macros, and AF, use of technologies including Java and PHP, and several respondents indicated that ClearCase was incorporated into their custom software. Strengths reported for these include cost efficiency, scalability, user friendliness, and the fact that changes and fixes can be made quickly. Weaknesses reported include lack of audit trail, version control, dependency tracking, uncertainty about compliance, having to take responsibility for training, availability of a small internal support group, and difficulty keeping up with the latest technology.

The average approximate number of global users on the respondents' SCEs is 426 users with a minimum of 7 and a maximum of 3,600.The average concurrent SCE users is 187 users with a minimum of 5 and maximum of 1,000 reported on the survey.

Access:

Definition: Access is the manner in which the users connect to the SCE.

Findings:

Of the responders in the survey, 98% said that the users are able to access the SCE remotely. The most common methods of remote access are VPN 50% of the time and Citrix 40% of the time. Less than 5% say the SCE is accessed through a direct URL.

Common Gaps:

The majority of respondents, 83%, indicated that they are satisfied with the performance of the remote access, whereas 17% indicated they are not satisfied with the remote access performance. The most common gaps around remote access are connectivity interruptions and connection speed. The internet is necessary with a remote connection and this tends to be the weakest link.

Best Practices:

When the remote access has a uniform interface irrespective of location, it enables an easy-to-operate login functionality. The remote access is generally available 24 hours a day which is also a benefit to the users because of the flexibility of working hours. Another benefit is when the remote access is supported through their I.T. organization; this provides secure and fairly stable access. Connectivity seems to be always more robust when access is onsite at the office.

Security/access control:

Definition: SCE user authentication and authorization.

Findings:

The most common way in which organizations grant access to their CE is with requests through IT (86%) with only 13% allowing the business to directly grant access with administrative functionality. Once users have account on the SCE, 49% continue to use requests through IT to control access to content/directories/files/areas, 24% allow the business to directly control this with administrative functionality, and 20% handle through the functionality in their software. In terms of technically how access to specific content is controlled, the most common way is using Unix/Linux controls with 20% using ACLs and 20% using groups. Windows groups are, used in all cases where the platform being used is Windows. 20% of organizations have software or a system that handles the content access.

Common Gaps:

When individual access is controlled at a multi-tiered level such as the reporting/submission event instead of the study level, the system may not allow for the granular level monitoring and control that may be required. When maintaining the list of user access is manual, it is tedious and time consuming, and as a result, not always up-to-date. This is especially the case when access reviews must be done on a project by project basis. Monitoring and controlling access in multiple environments, such as Windows groups and SAS directories also adds to the complexity.

Best Practices:

Overall network access being controlled by I.T. is a good practice which can properly handle prompt removal of all access when people leave the company or vendor/CRO. When access is granted to individual SCE systems within the network by the business function with minimal I.T. intervention, then the speed and modification of access seems to be the best method as long as it is well controlled and documented by trained individuals. This can enable appropriate audit trails and security. Also when the access is managed by a role, instead of individual access, then the simplicity of granting, monitoring, and modifying access becomes favorable.

Storage environment/file system:

Definition: The standard file-system structure used by the SCE

Common Gaps:

When the folder and programming environment is a manual process, then the request for folder structures can cause typographical errors. A manual process can also create a lack of security and audit trails.

Folder structure and naming conventions, when not set up in an optimized fashion, can also cause constraints. A gap reported was a case when a user is not able to modify a standard folder for a particular need. Also, when the project/drug program level files are stored at the same level can cause a reduction of the flexibility to manage the content at that level. And finally, when the hierarchy of file organization becomes multiple, say 6 levels, then the programming environment can become complicated and more challenging to learn.

Best Practices:

Conversely to the gaps, when a standardized directory structure for all sponsors/projects is optimized, then it can aid in a fluid programming environment. An example would be when reporting events are sub-levels of each of the top three levels (Drug, Project, and Protocol). It provides the user easy access and easily viewed content, especially when comparing multiple selected files against each other, or when the user needs to search for projects. A standardized directory structure promotes common understanding and easier reference and communication among colleagues, as well as facilitates development of tools to search, catalog and navigate the directories.

Other best practices captured were a metadata driven environment and structure as well as separating the data (for data management) and the analysis (for statisticians and programmers). In this example, it has been proven beneficial to have a clear understanding what is stored as well as eliminating unnecessary duplicate copies of data.

Archiving:

Definition: Processes and procedures used to manage the storage and accessibility of non-active programs.

Findings:

Few organizations have a true archival system which involves moving files from the production area to offline/offsite storage that is not accessible (just 11% of respondents). Twenty-two percent take a snapshot copy of the project work including datasets, programs, logs, lists, and output and store it offline with the original files remaining in the production system so that they may continue to be accessed and 36% move the files off the production system to another shared area with controlled access. One respondent referred to this last option as near-line storage as opposed to on-line or off-line. Sixteen percent of respondents indicated they have no archiving process at all but most of those consider the production area to be a secure long-term repository since they do "lock" projects at completion by restricting access in some way.

Common Gaps:

The most commonly reported gap around keeping files in the production system is potential inadvertent changes, which is especially likely if the process for "locking" projects is not well developed and can be forgotten or followed incorrectly. Also reported were concerns about the ever increasing storage requirements in the production system. On the opposite side, where files are moved off-line to an archive location, respondents reported having difficulty bringing files back if they are needed for analyses.

Best Practices:

There seems to be a lack of clarity in terms of best practice with regard to regulatory compliance in this area. Best practices seem to point toward keeping the files on-line so that they can be available for ad hoc, follow-up, and exploratory future analyses. Moving the files out of the production system to a separate, restricted system which still supports access for analyses seems to be the best option as it can better protect files from inadvertent changes. This has the added advantage of taking it out of the production system, reducing the load on the system as well as potentially reducing cost if the production system uses a hosted environment.

System Validation:

Definition: Processes used to validate the system components of your SCE.

Findings:

When asked the type of process used to validate the SCE and handle change control for its components, 57% of the respondents indicated that there is a robust I.T. driven

process, 12% indicated that the validation is done by the business according to their SOPs, and 7% indicated it is a combination of both. Of the respondents who answered, 98% felt their validation process meets regulatory requirements.

Common Gaps:

If it takes significant time and effort, particularly for an unplanned change, to complete the system validation, then the operational teams need to decide on taking the risk to develop workarounds. Documentation can become excessive (when paper based) and a lengthy validation process can be required for even a simple change. Moreover, changes in the system components in new versions may not always be caught during testing. Also, in the case where the validation is split between I.T. and business, this can add complexity particularly due to philosophy and interpretations of the requirements.

Best Practices:

A robust, well defined validation procedure implemented by external specialists or a combination of external and internal systems specialists seems to be the best practice. Some of the procedural steps that act on this best practice includes a UAT (user acceptance testing) that is conducted by the business specialists as well as ensuring that a disaster recovery plan or business continuity plan are in place.

System and process training:

Definition: Training for use of components of the SCE, including access to related servers.

Findings:

Fourteen percent of respondents reported having no specific training required prior to being given access to the SCE, but the rest did have required training, including ‘Read and understand’ (35.7%), Instructor lead (50.0%), and ‘Web or computer based’ (50.0%).

Common Gaps:

Web based and Read and Understand training is not as effective for many people who need the face-to-face interaction with an instructor but are definitely good for future reference and refresh. Training is often not kept up-to-date so there needs to be a clear owner who is responsible for maintaining each module. One-size fits all may not work as well as more role based training.

Best Practices:

Having only a basic training module required before gaining access can speed the time to get someone onboard and then users can complete the remaining training modules as they begin them work or when the modules are needed. Using a training management

system/audit trail to track and record training helps for regulatory inspections. Because people learn differently, it is better to use multiple delivery formats such as instructor lead, recorded, computer based, or read and understand. If live instructor lead training is used, recording it can be very helpful as it can be available to those who could not attend and for later reference. Depending on size of organization, collaborating with the training team may help identify more effective training strategy such as use of instructional software to create training modules, resources for the upkeep, training of different audiences, etc.

5.2. Programming Workflow and Processes

Programming life cycle/version control:

Definition: Development processes/flow used to develop both standard and non-standard programs for datasets and TLGs.

Findings:

Most respondents said that programs are developed within their system or interface from the beginning, but a third said they either developed them initially or entirely outside the system. The advantages of developing entirely within the system are greater control and the ability to generate accurate metrics to track work but the advantages to at least developing initially outside of the system are greater flexibility and potentially quicker development.

Almost half of respondents said they use software for version control such as RCS, VCS, or ClearCase. Of those that use version control software, RCS, ClearCase, and VCS were used about equally but almost half reported using other software such as EntimICE, Github, and MS Visual Source Safe. Version control was universally used for programs but also used for datasets and TLG outputs by close to half.

Most respondents have a lifecycle that involves development, then QC or review, and then final or validated where statuses are used to indicate the state of the program. While in development, programs generally have no protections and when they are promoted to a QC or review status, access is restricted and most reported the files being physically moved into a new production location. One respondent who did not have separate locations for development and production indicated they are working to implement in order to be able to restrict access. It is possible to restrict access without moving files to a separate location but it requires fairly sophisticated access controls.

Common Gaps:

Detailed and structured status flows can be time consuming and cumbersome. Thus, ensuring that the flow is as simple and streamlined as possible is important. Audit trails can become very large if all transactions are tracked which is an argument for not tracking during initial development and only starting once validation begins. Allowing

custom or informal statuses, even if there are standard ones available can become messy as custom statuses quickly multiply, making it difficult to generate meaningful metrics for tracking.

Best Practices:

Assigning statuses and moving programs to production areas and restricting access allows a system to track detailed history of file changes including status, date, user, and any comments. Having access controls in production allows confidence in the status of programs at any point in development. Version control allows easy recovery and comparison of files from previous versions as well facilitating audit readiness.

QC/review process:

Definition: Processes used to verify that datasets and TLGs are complete and accurate

Findings:

Eighty-eight percent of respondents answered that in their organizations, QC programmers are able to see the production code as they are doing their QC. There is disagreement on whether it would be preferable for QC programmers to be restricted from seeing production code but in most systems this would be complicated to implement due to the fact that most programmers serve in both QC and primary programming roles simultaneously.

Common Gaps:

Allowing the QC programmer to see the production code, although very common, was seen as a gap by many because it can lead to copying of code and not truly independent verification. On the other hand, people who reported not being able to see the production code while performing QC, indicated that this was also a gap because it did not allow code review. One respondent said that they were developing a system where a person performing programmatic verification would not be able to see the production code but a third person might be able to separately perform code review. Not having a well-defined process for QC was also reported as a gap.

Best Practices:

Commonly reported best practice includes having QC activities done within the system so that there are permanent records of the QC and it is linked to the production program. Also having a clear and well defined process to follow can help ensure QC is done consistently. Finally, full parallel programming is considered the gold standard but using a risk based approach to determining what level of QC to perform for each program is a very common way to ensure quality while reducing resource requirements.

Audit trail /traceability:

Definition: The survey asked the respondents to describe the components which they feel comprise an audit trail, if any, in the SCE. This could include revision history/comments for dataset and TLG programs, logs from programs, and date/time stamps and permissions on programs and datasets/outputs.

Findings:

Not surprisingly, those who have a system or interface as part of their SCE whether it is commercially available or developed in-house, reported a more robust audit trail and the commercially available systems seem to be the most robust. Having a system which runs analyses and tracks QC and review provides a wealth of metadata which can be used to generate audit trail reports.

For those not using a system or interface, common elements include the program, log, lst, output and dataset files themselves along with their date/time stamps, but also program change history in the program headers. Another common element is a manual QC tracking spreadsheet. One person reported having changes documented in the program files and in a system with the comments being linked together with a common numbering system.

Other elements thought to comprise an audit trail with or without a system are raw data files with a log or record of extractions, and comparison results when changes are made to files.

Common Gaps:

For users with systems, the common complaint is that the audit trail can be too comprehensive, saving every version of every file, full details of every transaction, and detailed comparison results for each file, including logs and lists as well as programs, outputs and datasets.

For users without systems, the problems are that the process is manual and can then be skipped or followed inconsistently leaving gaps in the audit trail. Also information regarding directory and file permissions (as well as those who had access to what and when), may only be periodic snapshots at best, not necessarily capturing every change.

Best Practices:

It is clear that systems or software offer a huge benefit in insuring a detailed audit trail which is automatically generated as programs move through development. Even when not using a system, having detailed process documentation can help to ensure that the correct steps are taken and all pertinent information is captured along the way. Effort should also be taken to define “what” to audit, record and trace and “when” to start doing it – for example, when entering testing stage, or production.

Dependencies:

Definition: Mechanism to flag or trigger rerun of datasets and TLGs based on changes to meta-data, data, programs and software.

Findings:

The most common method for tracking dependencies is manually documenting within a spreadsheet or the program header itself (47%). Twenty-nine percent have automated dependency tracking capability within their systems. A quarter of respondents indicated that they have no dependency tracking process at all.

Common Gaps:

Dependency tracking has a lot of overhead and can slow the system down.

Best Practices:

Dependency tracking is another clear benefit to using a system. This provides the most detailed, consistent, and reliable dependency information with little extra effort. The most robust systems will require programs to be rerun if any changes are made upstream to a dependent file and will not allow a rerun unless there has been a change upstream. Systems can also prevent programs from being finalized until all upstream files have been finalized. If no system is used, it is still possible to compile dependency information with the use of a macro which can write information to a dataset each time a program is run, making that information available for reporting.

Reproducibility:

Definition: The ability to reproduce the same results at a future time point such as years later.

Findings:

Almost all respondents indicated that they can reproduce analyses (83%). This is a key capability as it goes to the heart of traceability and data integrity. The ways in which people achieve reproducibility differ and, as is the case with other capabilities, vary by whether or not a system is used. The most robust systems ensure reproducibility by having full versioning on all files (datasets, programs, logs, lists, and outputs), allowing no files to be deleted, and tracking all version information for each deliverable. Less integrated environments achieve reproducibility primarily by keeping all files, including raw datasets, and making them read-only once finalized. One respondent added that they make a copy of global macros in the production area. One interesting comment, which is probably shared by many, is that they have never had to reproduce analyses but that they feel confident that they could if they had to.

Common Gaps:

With the most robust systems that save every version of every file, it was noted that this takes a lot of memory and clogs up the system. Two potential problems which may be

shared by many are that SAS version will change over time and may create problems with rerunning code, and that most people have never had to reproduce analyses so may not be familiar with the procedures for doing so.

Best Practices:

This is a case where having robust version control, traceability/reproducibility information and a solid project locking functionality, providing tight access restrictions are clear best practices. It is also important not to forget about the global macros used in the analysis and having a way to ensure that the correct version can be accessed whether with robust version controls or by simply making a copy of them at the time of analysis finalization.

Standardization/code reuse:

Definition: The extent to which standardized code is used and the mechanism to support the development and validation of code, and also the extent to which code is reused, copied or borrowed and any mechanisms in place to assist with standardization and reuse.

Findings:

When asked what percentage of tabulation (i.e., SDTM) datasets can be produced with standard code, requiring little or no modification for a typical study report, 18% said none and of those who did have standard code, 43% said less than half of tabulation datasets are produced with it, 57% said more than half, and 32% said greater than three quarters.

When asked what percentage of analysis datasets can be produced with standard code, requiring little or no modification for a typical study report, 12% said none and of those who did have standard analysis dataset code, 59% said less than half of analysis datasets are produced with it, 41% said more than half, and 21% said greater than three quarters.

When asked what percentage of TLGs can be produced with standard code, requiring little or no modification for a typical study report, just 6% said none and of those who did have standard TLG code, 50% said less than half of TLGs are produced with it, 50% said more than half, and 28% said greater than three quarters.

A large portion of the respondents have dedicated resources to develop and maintain standard code. All respondents indicated that they are able to copy code from one project to another and easily modify for use in the new project. This is a major part of code reusability, but only 46% said they have software that facilitates this copying, and the rest said it is a manual process.

Common Gaps:

Copying code can lead to errors if programmers are not careful to update study specific components such as headers, labels and treatment groups. Using fully automated

standard code can be risky because the programmer can become complacent and may not realize issues that arise. Contract Research Organizations (CROs) needing to match sponsor standards cannot always create internal standards. Companies must weigh the savings against the considerable cost involved in dedicating resources to create robust standard code.

Best Practices:

Macros are a form of standardized code and are handled separately in the next section. When there is software that facilitates copying of code, errors can be reduced by having the software automatically update the program header so that it is pointing to the new data source. SDTM automation is often facilitated and driven by metadata. It is hard to develop standard analysis dataset code unless you have standardized tabulation data, facilitating it. It is then necessary to have standardized analysis datasets in order to have truly standard TLG code, so many companies focus on developing their ADaM standards. When standard code is programmed in a simple way with very clear organization and comments to allow it to be easily modified, it can allow flexibility in handling non-standard input data. Systems where the standard code is fully automated and cannot be modified are not as good at dealing with non-standard input data but can be more efficient and can reduce programming errors. This is an area where having dedicated resources can make sense.

Macros:

Definition: Use of macros and how they are developed and deployed at each level in the hierarchy; including demotion.

Findings:

All but one of the respondents reported having standard global SAS macros. Some companies have dedicated resources for developing macros, some use a full I.T. driven validation process for macros and some develop macros outside the system altogether. There is a split between the best way to handle updates to macros with some versioning them, and others keeping the version static but making changes backwards compatible.

Common Gaps:

Without versioning, major updates may require creating a new macro because backward compatibility cannot be maintained. Very robust macros are often so complex that they are hard for new programmers to learn and costly to develop and maintain. Heavy use of standards and macros can make it difficult to integrate two programming groups after a merger. If there is not full buy-in by all users and stakeholders, work put into standards and macros can go to waste. Macros must be revalidated each time there are upgrades to SAS or server.

Best Practices:

Developing only simple macros for basic core functions can be done without dedicated resources. Simple macros which just do one thing are easy for programmers to learn and use, and easy to modify for a project if needed. A “building block” approach to macro development focused on small, simple “blocks” makes it easier for a broad group of people to develop, document, use, and maintain the macro system. Placing a snapshot of global macros in project directories can ensure reproducibility if macros are subsequently updated.

5.3. Interactions with other systems

Data Loading and Transformation:

Definition: Processes and mechanisms used to extract, transform, and load data so that it conforms to a standard structure and is accessible for analysis and reporting as well as to what extent is it automated.

Common Gaps:

If there is no tool to automate the data extraction then Programmers rely on Data Management to send them the data which can result in waiting and not having the most current data when it is needed. Since the folder where data extracts are saved is most often read-only, there needs to be a way to remove datasets which are no longer needed.

Best Practices:

Whether with commercially available software or custom developed tools, most companies automate the process of extraction from Data Management Systems. This reduces the possibility of error and creates a good audit trail. Since data extraction can be time consuming, having extracts run nightly after business hours can eliminate the need to wait for them to run during the day.

Importing:

Definition: The way in which external data sources, such as laboratory data or data from an external vendor such as a CRO, gets in to the SCE.

Findings:

Almost all respondents use a secure data transfer system/portal such as sFTP, but since those can be cumbersome to setup, there are still cases where data is ‘imported’ by email receipt. Robust systems have the ability to make an area of the SCE accessible to outside vendors so that they can deposit files directly.

Common Gaps:

Very robust systems can often be inflexible and require very standard file names and structures. If data receipt is not automated in some way, the manual process can be time-consuming and open to errors.

Best Practices:

Having secure data transfer system/portal or data transfer functionality in SCE software is the best way to have secure data transfer. Having a system such as sFTP, which lends itself to interacting programmatically so that receipt of data and logging of transactions can be automated, supports data integrity. There are many secure web based secure data transfer systems available which can be easy to setup and use, but many do not support programmatic interaction. The ideal would be for all external vendor data to be loaded into the Data Management System or Data Warehouse first so that Statistical Programmers can receive all data from a single source.

Unblinding:

Definition: Mechanism used to ensure that unblinded data is controlled and only accessible to appropriate personnel working on a program. Steps needed to apply unblinded randomization codes.

Findings:

Forty-six percent said that Statisticians or Statistical Programmers handle unblinding with a manual process, 21% have software that automates it, and 30% said that Data Management or the data management system handle unblinding.

Common Gaps:

The most common gap was that the Data Management System lacked the proper access controls to handle unblinding. This is not within the scope of the SCE but it could be advocated for by Biostatisticians and Statistical Programmers. Having an entirely manual process as many do, can introduce errors or at least add several steps in order to ensure that errors are not made.

Best Practices:

One good practice is to restrict access to the area where randomization work is done to just the unblinding Statistician. Another one is to have a separate area for the unblinded team to work in order to minimize the possibility of inadvertent unblinding. Creating dummy data to work with during development so that programs will run seamlessly once actual treatment codes are available was commonly reported. Because randomization/blinding/unblinding is such a critical component to the integrity of the clinical trial, if no other place has the complete set of robust controls describe earlier (audit trails, documentation, versioning, validation etc.), then this area must have these.

Publishing:

Definition: The way TLGs are moved from the SCE into reporting documents.

Findings:

Most respondents reported having a completely manual process for getting TLGs from the SCE into reporting documents. Often this involves providing RTF files to Medical Writing so that they can paste into MS Word documents. About a third did report having an automated process.

Common Gaps:

The biggest gap is having a manual process which cannot ensure that TLGs are not altered after they leave the SCE. With automated processes to insert TLGs into reporting documents, the gap is that there is no flexibility and TLGs must meet exact size and format requirements in order to incorporate into the reporting documents, including such things as fonts and use of headers and footers.

Best Practices:

Having at least some automation in this process is advisable as it is preferable to be able to ensure that no changes can be made to TLGs once they leave the SCE. This is achieved by some with SAS macros and others by the Medical Writers pulling TLGs with a system or MS Word macros and then rendering the document into a read-only PDF. Another common practice is just to provide publication ready PDF documents which are used as appendices instead of inserting individual TLGs into their publications. Programmers and statisticians should become involved in process and content discussions with medical writing and publishing colleagues. This leads to establishing a set of expectations and facilitates the development of useful tools.

5.4. Outsourcing and collaboration

Definition: How remote contractors, functional service providers, full service providers/CROs, and other groups within your organization such as Medical Writing, Medical Affairs, Drug Safety, Clinical, Research, etc. interact with your SCE, and how the SCE supports them.

Findings:

The majority (58%), but not all of the respondents said they have external parties such as CROs, FSPs, other vendors or collaboration partners that use their SCE. Some said that they have external parties doing work for them but only on the external party's own systems. For those who do have external parties using their systems, Citrix was the most common method, used by 60%, followed by VPN access at 45% and direct URL access at 30%. It is likely that those who have direct URL access capabilities are using a robust cloud based system.

Common Gaps:

Without a robust system, it may be difficult to appropriately restrict access when external parties are given access to the system. External parties will not be familiar with custom systems, so the learning curve may be higher. Citrix has some limitations such as not being able to fully support some common tools.

Best Practices:

Having external parties use the sponsor's systems when working on their projects gives greater control to the sponsor and increases consistency across projects. Having the ability to let external parties use the sponsor's systems offers scalability because resources can be brought in for any project at any time. Important elements should be captured in contractual agreements with external entities to avoid misunderstandings and disagreements. For example, capture the list of trainings to be taken and kept current, any specific processes the CRO personnel should follow, whether files can be stored on CRO systems or solely on sponsor system, whether CRO supporting software can be used, expected quality standards, etc.

5.5. Metrics/Project Management

Definition: Key performance indicators used to measure the productivity of vendors/resources, quality and timeliness of analysis datasets and TLGs. How the metrics are collected and are there systems that collect them or are they manually tracked.

Findings:

Few respondents (20%) feel that their software has good metrics capability built into it. Most (57%) do have some metrics manually tracked and only 14% feel they do not have good metrics available.

Common Gaps:

One respondent indicated that some commercial software does not offer good metrics. Others complained that using excel to collect metrics can be a problem due to occasional file corruption.

Best Practices:

The types of metrics vary widely so it is hard to establish agreement on what constitutes good metrics. Some respondents have SAS macros that collect metrics in datasets or excel for reporting so that is not quite built into their software but is also not entirely manual. One person reported that the information in their time and labor system, which tracks time spent on projects and tasks, is used to calculate project tracking metrics. The ultimate use of the metrics should dictate the type and amount of metrics to collect. For example, if metrics are to be used to provide project status updates, collecting SAS program completion status might be sufficient. If metrics are used to calculate detailed costs per type of analysis, then detailed time data is needed.

5.6. Regulatory Compliance

Definition: Discussion of which elements/features are important for regulatory compliance/what are the regulators really looking for.

Findings:

85% of the respondents to the survey consider their SCE to be compliant with regulatory requirement, however there were many comments indicating that they are not clear on what it means for an SCE to be compliant.

Common Gaps:

When software does not enforce compliance, it is dependent on personnel following processes. Without robust file protections, it is hard to convey confidence in integrity of data and analyses.

Best Practices:

Here is a selection of typical comments regarding best practices for compliance:

- Documented processes for development, maintenance, and access control
- Very good audit trail and access controls
- Validated and version controlled
- Traceability
- End-to-end Documentation
- Passed many audits over the years

6. Thank you to all of the organizations who contributed information

Accovion GmbH	d-Wise	IRIS	PPD
Adept Analytical, Inc	Eisai	Janssen R&D	PRA HealthSciences
Astellas	Eli Lilly	LEO Pharma	Quartesian
AstraZeneca	Emergent BioSolutions	LEO Pharma A/S	RHINOSYS
Bayer Healthcare Pharma	GCE Solutions	Merck & Co. Inc.	Roche
Bayer Pharma AG	Grünenthal GmbH	Merck Serono	SANOFI
Biogen	GSK	Navitas	S-cubed
Boehringer Ingelheim	GSK Vaccines	Novartis	Takeda Development Center
Celgene Corp	ICON Clinical Research	Parexel	
Cmed	Infinity Pharmaceuticals	Pfizer	
Covance	inVentiv Health Clinical	Phastar	

Note: There were some individuals who provided information independent of any larger organization and we would like to thank them as well.

7. Acknowledgements

The core team who drafted the paper include: Art Collins, Mark Matthews, Stephen Matteson, John Hirst, Nigel Montgomery, and Olivier Leconte
Additional acknowledgement:
Geoff Low, Emerging Trends and Technologies working group Chair