PhUSE US Connect 2019

Paper SP09

Fitting programming processes into workflows on systems – Is the square peg in a round hole?

Sridhar Vijendra, Ephicacy, Bangalore, India
Tyagrajan Swaminathan, Ephicacy, Bangalore, India

ABSTRACT

Statistical computing environments are an integral part of statistical programming and regulatory submissions. Such environments range from simple UNIX-based systems to user-friendly web-based systems. As programmers, we are used to writing programs and creating outputs in the programming environment and recording their status in disparate project documentation. There is a lag and a possible gap between what is done in the programming environment and what is recorded in a status tracker which has scope for human error and the potential to create inconsistencies in the audit trail. There is a need for workflows that integrate with our programming environment to keep track of programming tasks and record their completion without explicit user actions. But, is it possible to exactly mirror the real-world processes within such programming environments? Are human-error-prone processes suitable for machine-compatible checks or will we be left to force-fit workflows into systems that do not want to co-operate?

INTRODUCTION

Statistical Computing Environments (SCEs) are touted as the next big thing to revolutionize day-to-day programming and data analysis activities for the various teams involved in clinical trial reporting. But what really is an SCE? Programmers, statisticians and managers are all too familiar with the complex folder structures we use in our respective organizations that are often standardized across projects and studies to allow for easier navigation. The ubiquitous SAS® is integrated into such a data server and a version control system tied to it allows programs to be created, edited, and archived. Behind the seemingly simplified interface lie a set of roles, a plethora of security settings and folder permissions that usually call for the expertise of a dedicated IT administration team. We will refer to such systems as “conventional SCEs” since they do provide an environment for creation and archival of tables, listings and figures (TLFs) and datasets and study related documentation and are veritable complex systems. But where do they fall short?

For one, such systems cannot provide a detailed audit trail of the activities in the system which is the hall mark of an SCE [1, 2]. Date and time stamps of file edits can be captured to a certain extent but for instance, if a user were to view a file or dataset that they are not supposed to, conventional SCEs cannot track such an activity, especially in a way that is easily retrievable. A simple example would be that of a QC programmer in a double programming activity taking an unintended peek at the main side programmer’s code. Secondly, conventional SCEs offer no way to track project status by integrating it with the actual tasks being carried out in the system. In order to understand this better, let us consider the way documents are created, updated, approved and archived in document management systems. Almost every document in a clinical trial goes through multiple reviews and edits by various team members until a final version is signed off and distributed for reference. This process from creation to review to re-editing (incorporating review comments from a reviewer) to a final approved version is often captured in document management systems in the form of a workflow where the artefact (the document) changes states from *draft*, to *in-review*, and then finally to *approved*. In the process, the workflow will move from one team to another or from one team member to another. A digital or wet-ink sign off that usually accompanies the approval is then archived in the Trial Master File or a similar repository. Hence, all documents go through a *document workflow*, or in many cases, a specific workflow such as *Protocol document workflow* for the study protocol, and so on for other documents. What happens in the real world is the process as per the organization’s standard operating procedure (SOP) and the process is captured in the document management system using a workflow.

Unlike document management, management of programming activity for statistical analysis is a disjoint activity in conventional SCEs that is commonly accomplished using Sharepoint-like systems or the *docs* folder of the data server as it mostly involves editing of timeline documents, status trackers, resource trackers and the like. There is a disconnect however, in what happens in the real world versus what is captured in the trackers. The ADSL dataset may be made ready-for-QC one evening but may not be marked as ready-for-QC in the tracker until the next morning because the programmer was in a hurry to leave. An instant message sent by the programmer to his QC

counterpart is obviously not captured in the SCE. The reverse also might occur in the case of an ambitious programmer intending to impress his manager. In another case, a deliverable may be sent out on Friday evening and marked in the timeline document only on Monday morning with an earlier date. This is acceptable in most cases and many such disparities can be dismissed during the audit as *findings* but is there a way to make this all foolproof and also easier for the team? Is there a way to improve traceability and reproducibility and maintain a detailed audit trail without radically changing the way we work?

Modern day SCEs, cloud-based and otherwise, are making this possible in various ways [2, 3, 4, 5]. These are largely based on user-friendly well-engineered interfaces where users are not constrained to type out cryptic commands at a lone blinking cursor on a dark screen to make things work. One of the ways to streamline project management for statistical deliverables is to have workflows in the SCE that can be checked off by team members as and when steps in an activity are completed. This could be a programming activity such as creation and QC of an ADaM dataset or it could be a project management activity such as completion of a large deliverable with various components put together by a large team. This would mean that a programmer would not need to open and update the dreaded status tracker after every dataset or table is completed. This would also mean that a project manager or team lead could stop pestering his team for status and could simply refresh his screen every few hours to see how the work on a deliverable was progressing. And status reports would be just a click away.

This is fascinating and would mean the end of all spreadsheet troubles for project managers. But there are challenges. One, this requires a significant amount of setup to be done in the back end before work can be started. And in most cases, this cannot be done by hard-wiring programming workflows into SCEs. Considering that organizations tweak their processes periodically to enhance productivity and compliance, it would entail a significant amount of re-work every time a process is upgraded. This is in addition to the fact that workflows need to be tailor-made for each organization, since all our processes are somewhat different.

This means we need **integrated** and **customizable** workflows in the SCEs where we do our programing activities. **Integrated** workflows are integral to the SCE and are completed in the same system where the actions relevant to the workflow are performed. **Customizable** workflows allow for flexibility in adjusting the workflow as per the organizational SOP for each activity. Regardless of the process concerned, workflows in SCEs consist of several distinct steps, each one carried out by a specific role in the SCE [4]. The system must allow for users to be intimated (via email maybe) about their action required on an artefact (dataset or TLF) once the workflow moves to the step they have been assigned to. Overriding of specific actions is needed in all workflows, which will be recorded by the SCE as well. Undeniably, workflows can improve process compliance and enhance record-keeping. But what are the limitations? Custom creation of workflows is very much needed but just how much customization is necessary? How well can you fine-tune a customizable workflow? Will programming workflows change the way we work in the real-world? Will they change them for better or for worse? Will such workflows make documentation-averse programmers fall in line or more out?

This paper examines in detail the highlights and challenges of working with customizable workflows and what it takes to set them up and use them in day-to-day tasks. The mapping from a process to a workflow is carried out for a sample scenario typical of statistical programming activities. This paper does not identify with any SCE or any organization's processes but instead focusses on understanding the nitty-gritty of workflows as applied to managing typical programming activities in a hypothetical SCE that allows for integrated customizable workflows.

Please note that the term 'system' will be used in this paper to refer to a conventional SCE or a modern-day SCE, based on the context. The term artefact will be used to refer to a dataset and/or report (TLF).

FROM A PROCESS TO A WORKFLOW

In order to use a workflow to complete a certain activity in the system, it is necessary to understand how a workflow is used in an SCE. How does a process become a workflow? How is it integrated into the activities that are carried out in the system? This is best illustrated with a sample process such as the dataset creation and QC process. Almost every dataset in a study, including SDTM and ADaM, go through a creation and QC programming process, followed often by a review from a senior team member. One way to use a programming workflow here is to consider that every dataset that is created should go through the workflow in the system. This would require a **workflow template** to be created so that instances of the workflow can be used for each dataset that is programmed. These individual instances will be referred to as **work items** and they typically correspond one-on-one to the entries in the programming status tracker. For instance, the dataset DM will need a work item, with main side assigned to programmer A and QC side assigned to programmer B. CM will be another work item, with main side being done by A and QC side being programmed by C, and so on. To define these new terms formally:

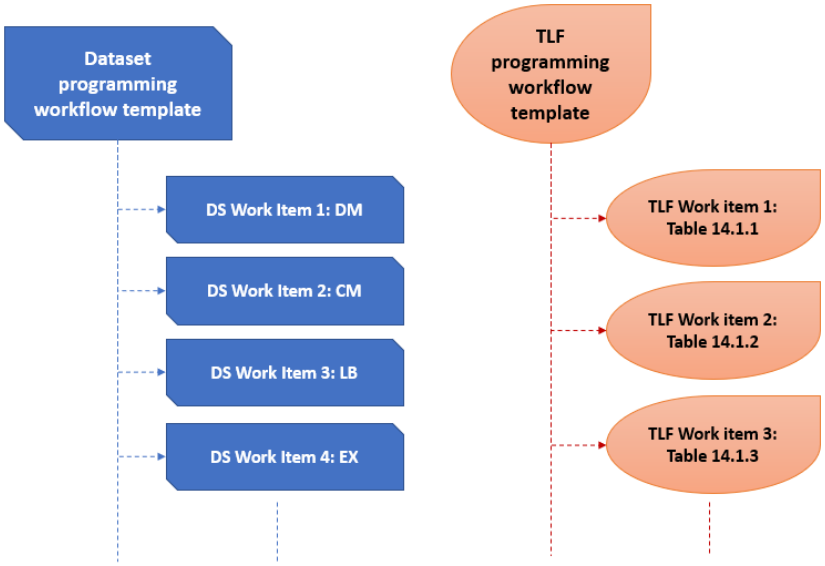
WORKFLOW TEMPLATE

A workflow template corresponds to one process as defined in the programming SOP say for instance, creation and QC of SDTMs. This template would be the blueprint of a set of work items, all of which follow the same process as configured in the template. A system might have only a handful of templates, one for each type of programming activity. These templates are used repeatedly, in the work items.

WORK ITEMS

Work items are instances of the workflow template. A system will have many hundreds of work items, one for each artefact created in every study, possibly for every deliverable. This is illustrated in **Figure 1**.

Figure 1: Workflow templates and work items



COMPONENTS OF A WORKFLOW

Before we proceed to dissect a sample programming process to map and create a workflow for it, let us become familiar with some additional nomenclature related to workflows. This nomenclature is neither universal nor standardized but will simply be used to explain the concepts in this paper:

STEPS

A step is the basic building block of a workflow and can be a sub-process within a process. In the most simplified version of a dataset creation and review process shown in **Figure 2**, step 1 of the workflow would involve creation (assuming no double programming involved) of a dataset and step 2 would involve review of a dataset by a statistician. Steps in a workflow may be sequential or parallel. As will be described later in this paper, the dataset programming and QC activity happens to be a distinctly parallel step while most review steps are sequential.

ASSIGNING STEPS TO USERS

Just like an entry is made for each programmer assigned to an artefact in the programming status tracker, similarly, users are assigned to work items in the SCE. Specifically, users are assigned to specific steps in the workflow in an SCE. This is illustrated in the example shown in **Figure 2**.

STEP STATES

A step exists in one or more states. Once a work item is created, a diagram of states can be shown for all the steps in a work item at any point in time. For example, when a dataset creation process is in step 1 with the state of *Programming in progress*, the dataset review step is in the state of *Not started*.

STEP RESETS

A step state can be reset from a *Completed* to a *Not started* state by a specific trigger. This trigger will most probably come from another step from which a feedback exists.

STEP NOTIFICATIONS

When a step is completed, and the workflow moves to a subsequent step, a notification is triggered to the user assigned to the next step. For instance, in the example shown in **Figure 2**, when the programmer creating the dataset completes their programming, a notification is sent to the user assigned to the subsequent step that the dataset is ready for review. Notifications can be sent in various forms, including email or a dialog pop-up within the SCE.

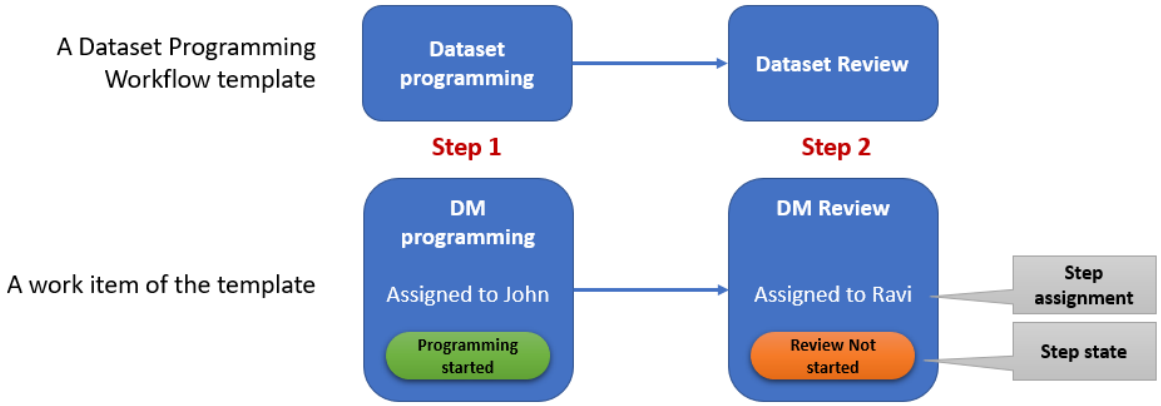
STEP REGISTRATIONS

Completion of a step in a workflow can be registered (recorded) in the system either manually or automatically. This is the essential part of the audit trail in the system. Manual step registrations are simpler but involve additional user action. Configuring the SCE for automatic step registration necessitates the overhead of the setup involved and is also prone to inadvertent step completions.

STEP OVERRIDES

Override of a step means that a user marks a step as complete or as skipped without performing the action required of the step. Override configuration will need to be done prior to step assignment. An example of a step override is that of a dataset review that is assigned to a lead programmer but is overridden by the statistician, because they find it unnecessary for certain reasons. The override is also captured by the SCE.

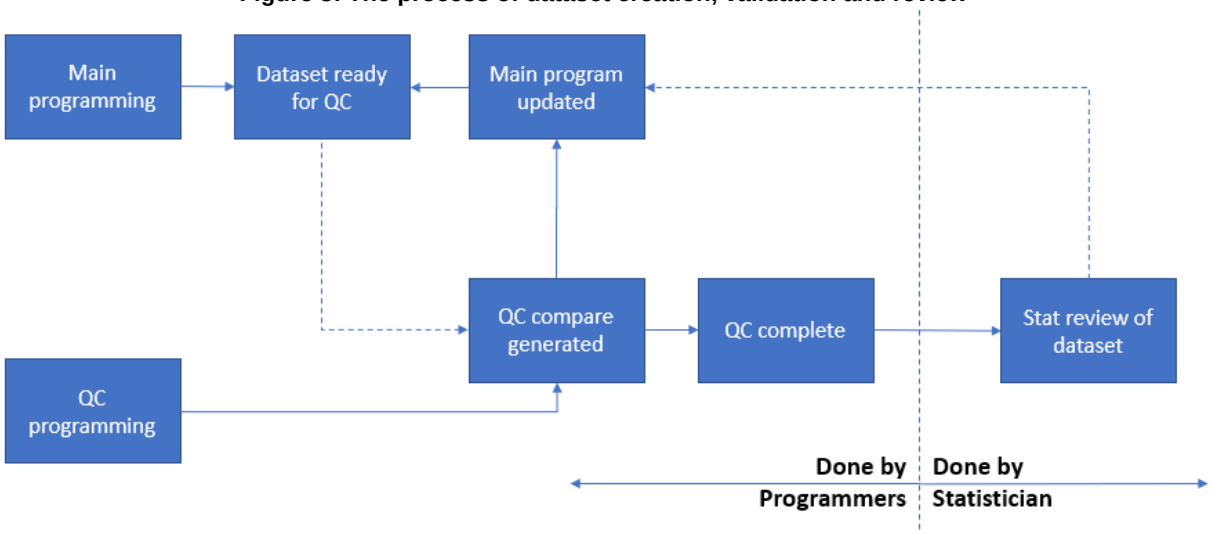
Figure 2: Components of a workflow



THE DATASET PROGRAMMING WORKFLOW TEMPLATE

Let us consider the typical scenario of a workflow to create, validate and review a dataset that we see in our teams on a day-to-day basis. **Figure 3** illustrates this process. Additional details such as code review, log checks, batch submit checks and version control check-ins have been left out of this process for the sake of simplicity.

Figure 3: The process of dataset creation, validation and review



What would be the best way to convert this process into a workflow that will allow work items to be created for each dataset that is programmed for every deliverable? To simplify the process, here are a few premises:

- The dataset specification is ready for programming. It is apparent that if dataset specification writing has to be part of this workflow, then this has to be made into another step that comes before dataset programming
- Minor processes such as code review and log checks albeit important are not going to be included in the workflow for now.
- There are only 2 types of steps in the SCE:
    - a simple sequential step that can be assigned to a single user
    - a parallel step that can be assigned to 2 users.

  A sequential step cannot be assigned to 2 users at the same time because if a step needs 2 users to complete it, it clearly indicates that there is a sub-process involved that the 2 accountable users need

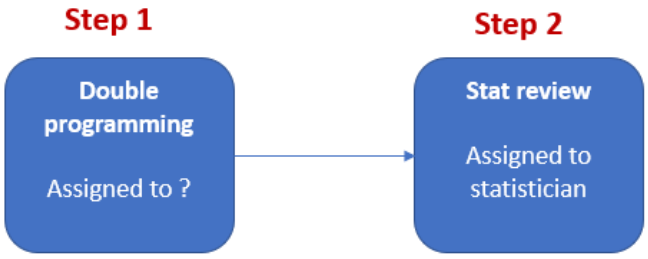
to complete one after the other or at the same time. Conversely, a parallel step cannot be assigned to a single user since 2 parallel activities assigned to the same user can be clubbed in a single step.

- The SCE we are using allows workflows to be customized in every way possible

WORKFLOW VERSION 1

Let us consider the most simplified workflow derived from the above process. In this simplified workflow which we will refer to as Version 1, the first step is double programming and the second step is dataset review as shown in **Figure 4**. Now the question arises as to who should be assigned the first step – the main programmer or the QC programmer? And as per the premise mentioned above, one step cannot be assigned to 2 users. Hence this version of the workflow does not align itself to the actual process that is shown in **Figure 3** and does not capture many of the states of the typical QC cycle.

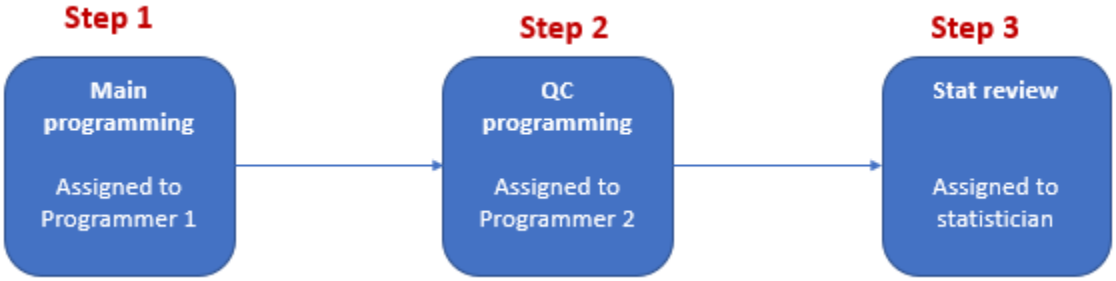
Figure 4: Version 1 – an overly simplified workflow



WORKFLOW VERSION 2

The next logical step would be to split the first activity of double programming into 2 steps so that it can be assigned to 2 separate users. This version is shown in **Figure 5**. This would mean that main programming is started first and when that is complete, the QC programmer gets a step notification to start programming. But this presents a problem, since very often, QC programming can be done in parallel with main programming.

Figure 5: Version 2 – More steps to the workflow



WORKFLOW VERSION 3

It is easy to see that if we quickly change the first 2 steps of version 2 in **Figure 5** to a parallel step as shown in **Figure 6**, main programming and QC programming can be assigned to different users and more importantly, both these activities can be started independent of the other. The challenge however is about how the QC programmer is intimated about the main dataset being ready for QC? There is a step notification when main programming is completed, but the notification goes to the statistician for initiating review. When does the QC programmer know when to start QC?

WORKFLOW VERSION 4

Version 3 of the workflow shown in **Figure 6** does not address the issue of the QC programmer getting the necessary trigger to start the QC process, hence we can try adding another step to the workflow between the parallel step and the review step. This results in the workflow shown in **Figure 7**. When the main programmer completes the dataset programming, the step notification goes to the QC programmer (programmer 2) assigned to the step named *QC process* (Step 2). The QC programmer can then complete the QC process and once the dataset is QC passed, then the step notification is sent to the statistician to begin review the dataset. Looks like we are set. But what happens if the QC programmer needs to communicate any comments to the main programmer and the main programmer needs to update the main dataset? This must be done offline, away from the SCE, since the QC interactions cannot be captured in this workflow. But can we add more steps and capture this process?

Figure 6: Version 3 – Introducing a parallel step

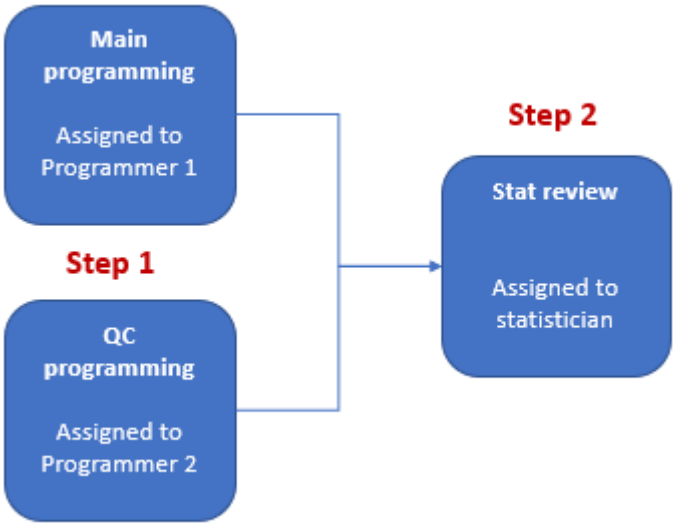
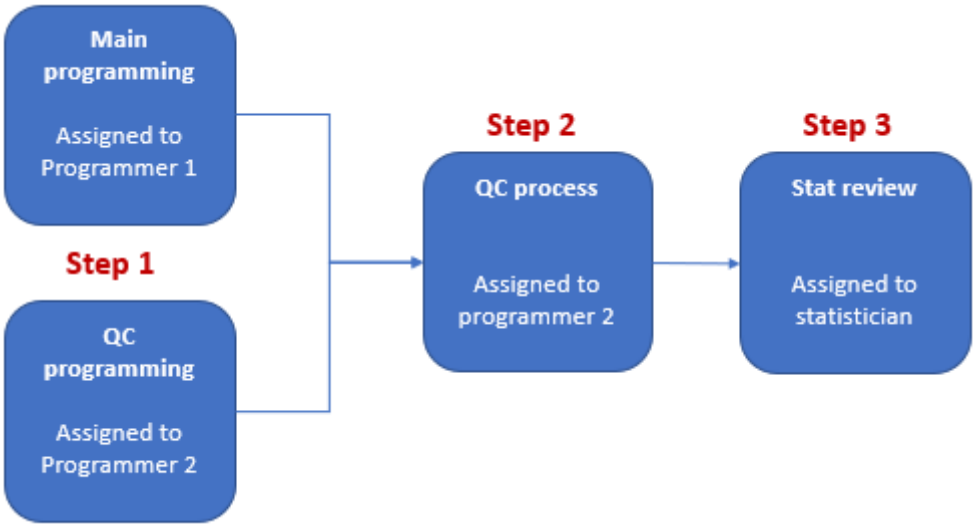


Figure 7: Version 4 – Adding an additional QC step



WORKFLOW VERSION 5

Version 4 of the workflow shown in **Figure 7** appears to capture the dataset programming workflow to reasonable accuracy, enough for most audit trail purposes. To make things complete and make project management easier, however, the workflow can be upgraded further by adding a feedback loop from the QC process to the first step. Any feedback from the QC process can be used to reset the state of the first step to *Programming in progress* so that the programming updates made during the QC process can be captured in the workflow and will be visible to project managers tracking the project status. This feedback loop is illustrated in **Figure 8**.

For most part, version 5 shown in **Figure 8** has the potential to capture the important aspects of the programming process while also allowing for programmers to mark work items complete as and when they complete programming activities. **Figures 9** through **12** illustrate a set of step state diagrams that the Version 5 workflow would go through for a single dataset. The state of each step in each state diagram is shown in an oval within the step. Most SCEs provide a comprehensive and possibly visual view of the status of all work items in a project for the team lead or project manager to get a better view of the state of the deliverable.

Additional detail could be added to Version 5 to make it more comprehensive. If log checks and version control check-in of a program is to be recorded, then that would be another step. If code review is needed, another step and another loop is needed and so on. There is a however, a limit to the usefulness of adding too many finer details in a workflow since it also complicates a reasonably simple process and adds the overhead of too many step notifications and actions to perform for programmers who would rather spend a better chunk of their time writing code.

Figure 8: Version 5 – Introducing the feedback

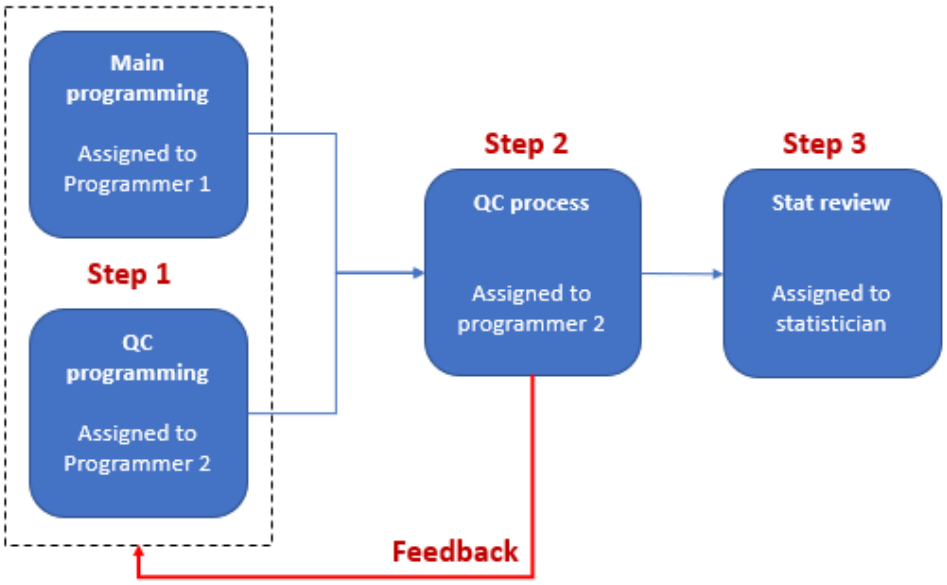


Figure 9: An early state diagram for the Version 5 workflow

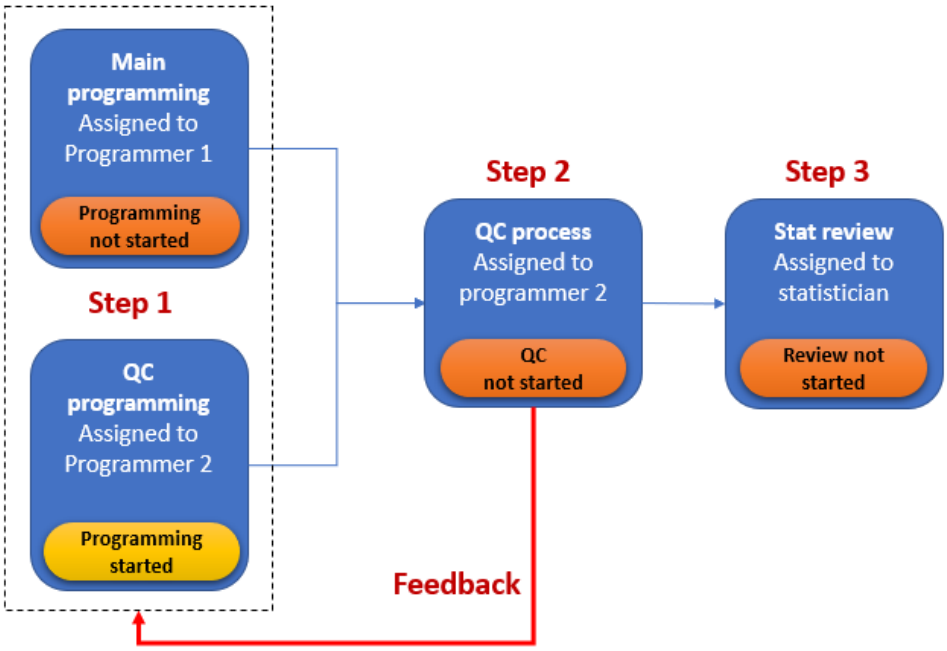


Figure 10: A mid-process state diagram for the Version 5 workflow

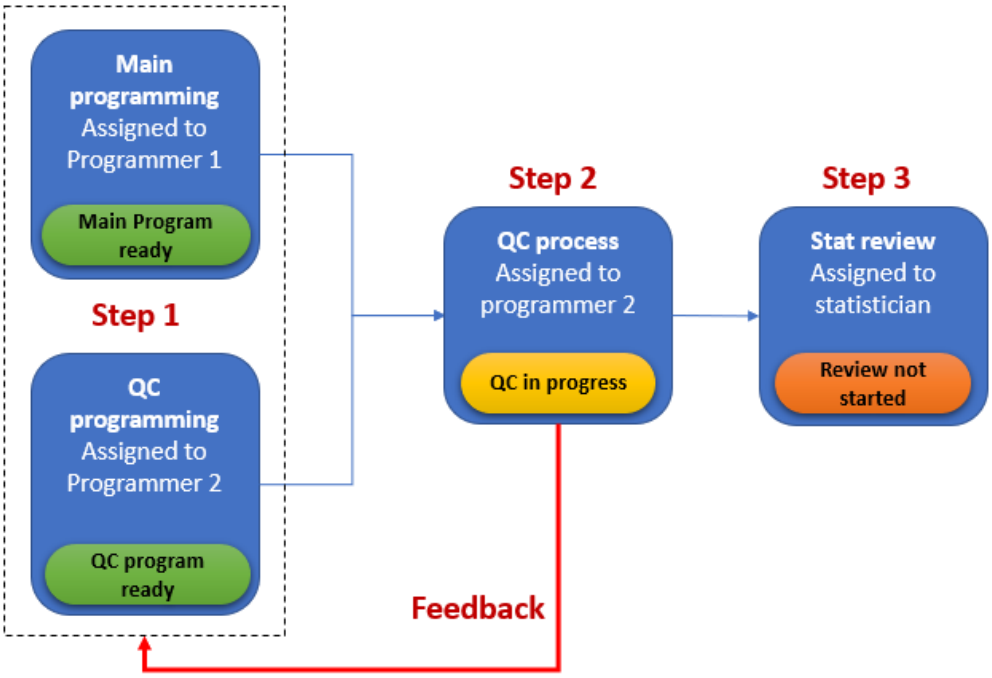


Figure 11: A state diagram for the Version 5 workflow showing the QC feedback

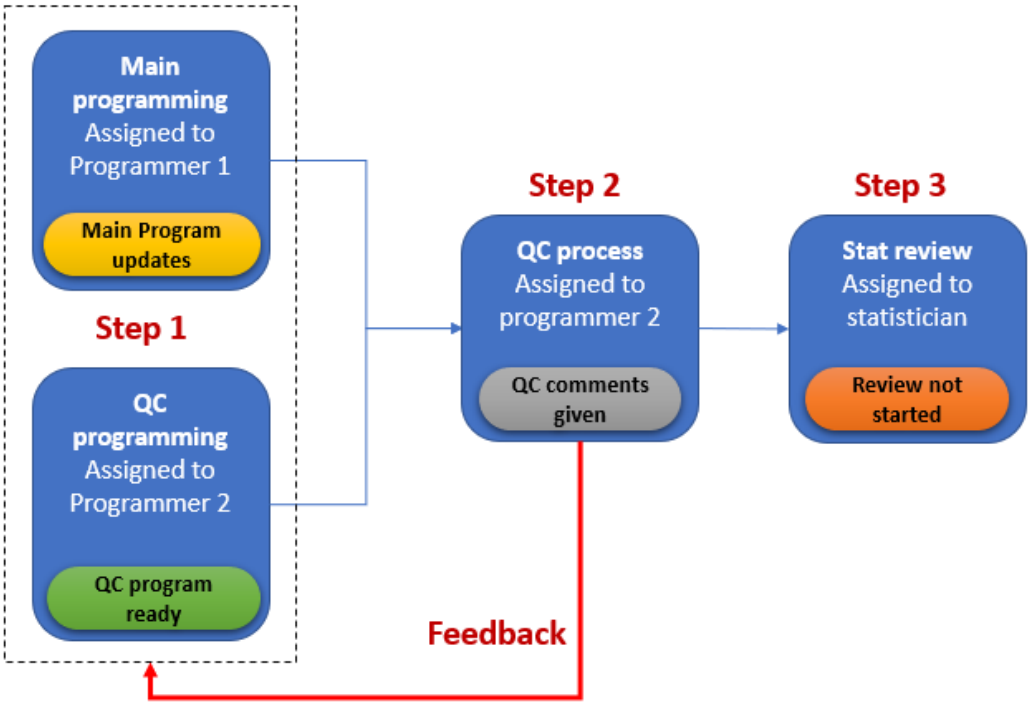
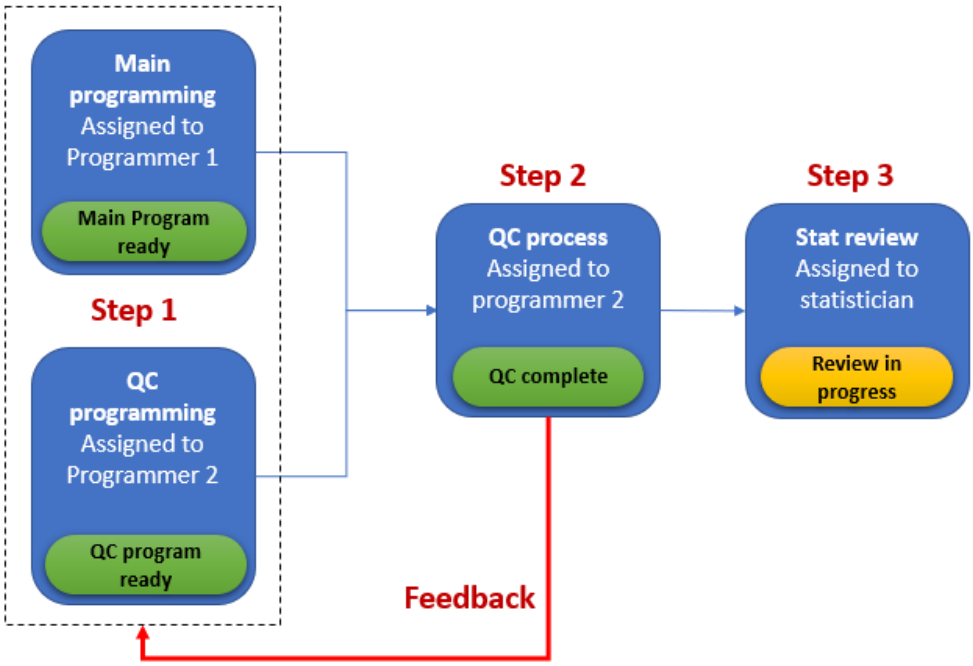


Figure 12: A state diagram for the Version 5 workflow showing the completion of programming and start of review



ADDITIONAL WORKFLOW TEMPLATES

Like the workflow presented above for dataset programming, SCEs need additional workflows for other items that are part of the deliverable such as TLFs, define.xml etc. depending on the scope of the study and the deliverable. These workflows could be similar to the dataset programming workflow discussed above or very much different depending on each organization's SOPs. What is more important here is how these workflows are inter-connected to one another. When the SDTM workflows are completed, ADaM workflows should be triggered and similarly, when ADaM workflows are completed, TLF workflows need to be triggered. This will always ensure traceability within the system. Configuring such top level workflows consisting of multiple smaller workflows is no simple task considering the amount of foresight and analysis needed to configure something that can be frozen only after a certain amount of trial and error and real-time project usage.

CONCLUSION

SCEs are the next big wave all set to upgrade decade-old processes in statistical programming to improve productivity, subsequently leading to lower time-to-market for drugs. While many other features of new-age SCEs are close on the heels of what our conventional SCEs already offer, one of the features that stand out is that of workflows within SCEs. Workflows have the potential to vastly improve traceability and provide detailed audit trails while also making project management for study deliverables a seamless activity that is integral to the SCE.

SCEs with customizable workflows are preferred over SCEs that come with hard-wired workflows for many reasons not limited to - better fit for differing organizational SOPs, allowing for easier upgrades as processes evolve and, allowing for tweaks to suit slightly differing processes across teams within the same organization. In this paper, a hypothetical SCE with customizable workflows was considered for configuring a dataset programming workflow to be used to create various datasets needed for a deliverable. It was demonstrated how a seemingly simple dataset programming workflow can be configured in many ways using customizable workflows. Using a simple state diagram, a not-so-complex version of the workflow was shown to suit the needs. There is no perfect way to configure such workflows and often, what works for one organization or team may not work for another. Setting up of workflows takes time and an in-depth analysis of the underlying processes being mapped. Customizing and setup of workflows can be a long-drawn process with a lot of lessons learnt along the way.

It is very easy to see that programming workflows can vastly improve traceability, enhance audit trails in the system and significantly reduce the scope for human error in statistical programming deliverables. The subsequent advantage for project managers is fewer spreadsheets and implicit project tracking which is a welcome relief. But it is important to keep in mind the overhead needed for setting up programming workflows in SCEs for large organizations since project tracking becomes more than just starting with an SOP and an empty spreadsheet template. Too many steps in a workflow can impede productivity and stifle natural process flow whereas too few steps can lead to poor process capture resulting in a very diluted audit trail. It is also important to keep in mind how

programming workflows can affect the way programmers function and the kind of resistance programming teams may have to work with elaborate workflow setups.

There is also no one-size-fits-all solution for setting up a catalog of workflows for capturing the end to end processes in statistical programming starting with the raw data until the define.xml. There are plenty of ways to accomplish the same result and anything works if end to end traceability is possible and project tracking is reasonably uncomplicated.

In summary, modern SCEs are paving the way forward to improve traceability and audit keeping and programming workflows are the need of the hour for improved project tracking in these newer systems. For multiple reasons, customizable workflows are better than hard-wired ones. Customizing workflows for programming processes is very much possible but entails significant planning and plenty of trial-and-error and fine tuning before a suitable solution can be set in place for regular usage by programming teams. It is important to stop at the right level of detail in programming workflows since most of the time, a simple solution is quick and better than a complex and tedious one.

REFERENCES

[1] Alan Hopkins et. al., Statistical Computing Environments and the Practice of Statistics in the Biopharmaceutical Industry, Drug Information Journal, Vol. 44, 2010.
[2] State of the SCE, A White Paper by the PhUSE CSS Emerging Trends and Technologies Working Group Statistical Computing Environments (SCE) Project, April 2016
[3] Peng Yang, et. al., Using Workflows and Metadata Information to Standardize Business Processes in Pharmaceutical Programming, PharmaSUG 2013, Chicago, USA.
[4] Tyagrajan Swaminathan, et. al., Regulatory Compliance in the Digital Age – A cloud-based statistical computing environment (SCE) to the rescue, PharmaSUG 2018, Seattle, USA.
[5] Yeqian Gu, How Process Flow Standardized our Process, PharmaSUG 2017, China.

CONTACT INFORMATION

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

Sridhar Vijendra
Ephicacy Lifescience Analytics
Bangalore, India
sridhar.vijendra@ephicacy.in

Tyagrajan Swaminathan
Ephicacy Lifescience Analytics
Bangalore, India
tyagrajan.swaminathan@ephicacy.in