PharmaSUG China 2023 - Paper CC-106

Software Development Life Cycle in Developing Statistical Programming Tools

Yan Qiao, BeiGene, Beijing, China

ABSTRACT

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use. SDLC can be applied in developing statistical programming tools.

INTRODUCTION

Many pharmaceutical companies have a dedicated team (infrastructure teams or standards and tools team) focusing on defining the standard process for programming work and developing related tools (SAS macros, R functions, VBA tools, Python tools, etc.) for the process from rawdata to SDTM, from SDTM to ADaM, from ADaM to TFL, and submission package preparation. Development of statistical programming tools should follow Software Development Life Cycle (SDLC). The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. In detail, the SDLC methodology focuses on the following phases of software development: user requirement analysis, design stage, development stage, testing and validating, deployment.

This article will explain how SDLC works to develop statistical programming tools (mainly SAS macros), dive deeper in each of the phases, and provide you with examples to get a better understanding of each phase.

USER REQUIREMENTS ANALYSIS

Before developing/updating a statistical programming tool, the developer needs to collect user requirements from whoever has requested it. The requester can be from inside statistical programming team or from external parties, such as statisticians' team, medical coding team, etc. The developer must collaborate closely with the users and communicate possible coding or technical challenges. If it is a new tool, the developer needs to know the main functionalities of the tool and what flexibilities the user wants to have while using this tool; if it is an update to existing tools, we need to know whether it is adding new functionality, modifying existing functionality, bug fix or other requirements.

Each version of the tool should have a well-documented user requirement and the user requirement should be reviewed and approved by the requesters. It is extremely important to gather the user requirements accurately and completely, and document them clearly and concisely. Documenting clear requirements will reduce the chance of ambiguity and misunderstanding in the development stage of the program and will reduce rework.

DESIGN STAGE

After we have got the user requirements and before we start to write any code, it is very important to plan and design the tool. Members of the infrastructure team will sit together to have a roundtable discussion: is this a relatively simple tool or complex tool? For simple tools: what is the best name for this tool? what are the input and output? what are the parameters/arguments? which parameters/arguments are required, and which parameters/arguments are optional? is there default value for the parameters/arguments? is there any constraint with this tool? For complex tools that need to split it into different modules, what is the name and functionality of each module? how the modules depend on each other? what are the coding conventions among the modules? How to assign the work among team members? Design stage is the most important part of SDLC since it guides the following work and make sure the work is done in a consistent and planned way.

During the design stage, the developers should also initiate the user manual for this tool. Many companies have their own template for user manual, usually including the following sections: description,

operation environment, parameters, constraints, examples and usage notes. Most sections (except for the examples) of the user manual should be decided in the design state. The initial version of the user manual should be reviewed by the requesters and all developers.

DEVELOPMENT STAGE

Once the design phase is over, the next phase is coding. In this phase, developers start build the tool by writing code using the chosen programming language (SAS, R, Python) and follow the decisions made in the design stage. For complex tool, tasks are divided into units or modules and assigned to the various developers. In this phase, tool developers should always keep GGP (good programming practice) and defensive programming in mind. Good Programming Practice includes standard header, proper comments, standard naming conventions, standard coding conventions and avoid of hard coding. Defensive programming is important in tools development since the tools are intended to use on all kinds of data by all users in the organization. A robust tool should be able to anticipate all possibilities in the future and handle the extreme cases elegantly. The author has written a paper on how to develop robust SAS macros and presented at China PharmaSUG 2019. The following topics are discussed in this paper: defining parameters; parameter validation; checking and reacting to outcomes of global statements, data steps, procedure steps and macros calls; returning status to the caller; restoring the environment.

TESTING AND VALIDATION STAGE

Only formerly validated tools can be used in both production and QC sides. Many companies have their own workflow to validate statistical programming tools. To validate a statistical tool, the following aspects should be covered:

1. review of source code and user manual
2. test case on error messages and return code

This is to test the tool can produce the expected error message and return code if the tool is wrongly used: such as required parameter is not specified, parameter is specified but value is invalid, input dataset/file does not exist, input dataset exists but required variable does not exist in it, output location does not exist, etc.. If the tool is wrongly used, it should provide enough information in the log to instruct the user on how to use it correctly.

3. test case on functionality

This is to test that when the tool is correctly used, output from the tool should be correct and accurate. It is better to use real phase 3 study data to test the functionality and usually we use double programming to validate the outputs.

4. test case on extreme values

This is to test that the tool can function as expected on extreme values. Dummy data with extreme values can be used in this test case.

All the test cases and testing programs should be prepared by the developer and documented in a Validation Protocol with detailed instruction on how to execute the test cases and the “Expected Outcome”. The validation protocol should be reviewed and approved by the requester.

After the validation protocol is approved, a Tester should execute the validation protocol following the instructions and record the results in the “Actual Outcome” and take screenshots as proof. If the actual outcome matches the expected outcome, the test case is marked as “Passed”; if the actual outcome does not match the expected outcome, record the deviations in the Test Deviation Report to document root cause, impact, and corrective actions necessary to resolve the deviation. The corrective actions can be updating user requirements, configurations, codes, execution instructions, etc.. Once testing is completed, tester forwards the executed validation protocol to a QA person. QA person reviews and approves the executed validation protocol. If retesting id needed, the tester should rerun the validation protocol after all the corrective actions are taken until all the test cases have status as “Passed”.

After the executed validation protocol is approved and all test cases have status as “Passed”, tester authors the Validation Report. The validation report records the approval date for all validation documents, including user manual, user requirements, validation protocol, execution of validation protocol. It also records all the deviations. The validation report should be reviewed and approved by the QA person.

The validation protocol, executed validation protocol together with screenshot proofs, validation report should be documented in QA system as proof that the tool is officially validated.

DEPLOYMENT STAGE

After the tool is officially validated, it should be deployed to be used by users. Different programming language has different deployment method. For SAS macros, it is a good practice to use macro catelog. Macro catelog provides the option of hiding the source code or not. You can secure your source code by hiding them. For R functions, you can create your own R package consisting of a group of related R functions.

Also developers need to finalize the user manual by refining it and adding examples in it. User manual should be released together with the tool. Users hate to use a tool without a detailed user manual, especially when the tool is extremely complicated. A good user manual means half of a successful tool.

VERSION CONTROL

Even though version control is not a part of software development life cycle, I still want to discuss it in this paper since it is an inevitable part in developing statistical tools. We always need to update our tools due to different reasons and we want to keep the updates in track. Each SAS macro has a version number, such as %m_xxx_v1. For minor updates that is back compatible and will not affect user's calling program (such as bug fix, macro metadata update), we usually don't change the version number. For major updates that is not back compatible, and user needs to update their calling program (such as adding new parameter, adding new functionality), it is better to upgrade the version number, such as %m_xxx_v2. All the updates should be clearly documented in the macro header and user manual, so that users will know the difference among different versions and pick the correct version to use.

CONCLUSION

This paper describes the software development life cycle in developing statistical programming tools (SAS macros, R functions). The following phases of software development are discussed in detail: user requirement analysis, design stage, development stage, testing and validating stage, deployment stage. By applying SDLC, tool developers can develop efficient, accurate, robust and user friendly tools and thus improve the whole team's working efficiency.

REFERENCES

Yan, Q. 2019 "Writing robust SAS macros" PharmaSUG China 2019 Conference

Available at https://www.lexjansen.com/pharmasug-cn/2019/CC/Pharmasug-China-2019-CC50.pdf

CONTACT INFORMATION

Yan Qiao

Associate Director of Statistical Programming, BeiGene

yan.qiao@beigene.com