PhUSE 2017

Paper AD04

Statistical End-User Tool Development

Mickael Borne, 4Clinics, Paris, France

ABSTRACT

To automate the creation of a complete statistical report for routine experiments, one often needs to develop a computerised system. If the need is a simple standardised statistical report from standardised data, the computerised system can also be simple. The Statistical End-User Tool, which can be considered halfway between software and a scientific calculator, has a user interface (UI) to allow users to enter data and request analyses, a statistical core to perform the desired analyses, and a component to manage the publishing of the report. Here, I describe the deliverables and steps needed to develop a Statistical End-User Tool and present an example I have developed for creating statistical reports based on the use of Bland Altman analysis in bioanalytical method comparison studies. This example tool employs Excel® as the UI for entering data, SAS® as the statistical core, and Word® for publishing the report.

INTRODUCTION

The theme of the 2017 PhUSE conference is “Digital Innovation in Healthcare”. For me, as a statistician, one way to benefit from digital innovation is to promote the development of Statistical End-User Tools. This concept represents a simplified computerised system that allows a researcher, toxicologist, pharmacologist, or anyone who has standardised data to request a statistical report at the click of a button.

Before going further, a few words about the author: I am a statistician and not an application development specialist, so some words and phrases used in this paper to describe development could appear ‘exotic’. For the same reason, this article will focus on the statistician/developer point of view for the development. Moreover, I have not invented the concept of the Statistical End-User Tool; I discovered it in the preclinical biostatistics department of a large pharmaceutical company where I was a contractor. I worked as a statistician/developer for over 9 years with many brilliant colleagues, and have since worked for 4 years as a statistician at 4Clinics, France where last year I started to offer our clients a software development service. Our first in-house developed tool was successfully delivered this year.

First, I will present the concept, highlighting some of the challenges faced by Statistical End-User Tool development teams. Then, to better explain what Statistical End-User Tools, I have created an example that you are free to discover in the Appendix to this article. In a clinical laboratory, the need to assess the agreement between two quantitative methods of measurement is a common task. The aim of the example tool is to provide a statistical report based on the use of Bland Altman analysis in method comparison studies. The main components of a Statistical End-User Tool will then be detailed, first in general, then for the example.

A user interface (UI) is a point of interaction allowing data transfer between the user and the computer system. In the example, Excel will be used as the UI, and the advantages and disadvantages of this choice will be discussed. For the statistical core of the tool, I always choose SAS, in which the batch mode function is particularly useful. In a computerised system, the user is not allowed to perform manual post-processing, so the generated statistical report must be well designed and complete. Although SAS has many in-built functions to generate reports, I do not recommend using its report publishing functions in computerised system development. In the example I provide, Word is used to manage the publishing of the statistical report, and I discuss the mailing functionalities of Word.

The last part of this article will discuss some specificities linked to software development, such as audit trail management and the versioning of the programs. All the deliverables of a Statistical End-User Tool will be detailed, including the documents and components of a release, as will operations management, system maintenance, periodic review, and change controls.

STATISTICAL END-USER TOOL DEVELOPMENT

A Statistical End-User Tool is a software custom designed to suit the business process. Like all software, it has a life-cycle beginning with a kick-off meeting (KOM), a definition phase, a development phase and a validation phase. Beyond approval of the Validation Report and product deployment, the life-cycle continues with release and maintenance up to a possible retirement.

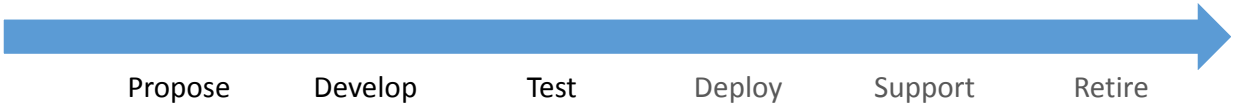


Figure 1. Software development life-cycle

In this article, I focus on the development and testing of the Statistical End-User Tool life-cycle, up to deployment. Moreover I focus on the supplier perspective (i.e. where supplier refers to software developer).
The method used for the development of this type of tool is freely inspired by the Agile method (see below). Furthermore, because this tool will be used in a regulated context where good manufacturing, laboratory or clinical practices (GxP) must be followed, validation will follow the Good Automated Manufacturing Practice (GAMP) guidelines, specifically the GAMP 5 guideline for GxP Computerized Systems.
We named this computerised system a Statistical End-User Tool because this software allows anyone, even those without training in statistics, to produce full statistical reports.

AGILE-LIKE APPROACH TO PROJECT MANAGEMENT FOR STATISTICAL END-USER TOOL DEVELOPMENT

Agile software development methods are based on a manifesto published in 2001, available at http://agilemanifesto.org/. The manifesto is composed of twelve principles, the first of these principles (and that with the highest priority) being “to satisfy the customer through early and continuous delivery of valuable software”. [1]
In brief, Agile’s recommendations are:

- Iterative development and short term goal: one of the key points of Agile development is to deliver regular working software versions to the client
- Adaptability: specifically to “welcome changing requirements, even late in development”
- Self-steering teams, to ensure the best architectures and designs with a clear understanding of the user requirements,
- Engaging stakeholders, and maintaining trust and motivation to ensure success

However, the pharmaceutical industry is a regulated sector that requires a highly documented approach. Hajou, Batenburg and Jansen, from Utrecht University, have developed a method they named ‘æ’ based on Agile’s principles but tailored for the pharmaceutical industry. [2]
Statistical End-User Tool development projects are always small in scale. They are not generally classified as software, instead being custom designed programs to automate creation of a standard statistical report, and usually take around 8 months (not full time for the supplier).
Consequently, a cutting of the project cannot be found in sprints or sections of coding (as can be the case in a ‘classical’ Agile project) but there will be iterative release of the tool.

Roles and responsibilities from the supplier perspective

The ***project coordinator*** is typically a statistician, whose role is to manage the project and act as the principal developer. The project coordinator also acts as the point of contact between supplier and client, leads meetings with the client and/or the internal team, assists the client in the writing of the user requirement document, and writes the technical specifications.
The second role is the ***tester***, the person who validates the development. This person validates each release of the tool, writes the forms for statistical test calculations and the development qualification, performs the tests, and writes the development qualification report.
Finally, the development team may involve **experts** specialised in the project’s statistical domains or development languages.

Roles and responsibilities of the client (e.g. pharmaceutical industry)

The major role from the client perspective is the ***project leader*****,** usually a technical expert, who is the main point of contact with the statistician from the supplier.

Success of the project is dependent of the relationship between the project coordinator (supplier) and the project leader (client). The client's project leader usually has a technical background, and will be in charge of the administration of the tool after the final delivery. This person has a lot of documents to write and meetings to attend, as well as software tests to manage or perform. Moreover, this person is the interface between the supplier, the users, IT, and other parties.

One of the future users of the tool must be identified as the ***user leader*** to centralise requests and assist the client project leader and the supplier project coordinator in the user requirements specifications (URS)/functional specifications (FS) writing. Often this user will also be involved in the operational qualification.

An **IT representative** must also be identified to oversee, for example, any necessary server configuration, and account creation and management.

The ***validation leader*** (who assists the project leader for the qualification of the tool) is required to ensure the tool's quality (e.g., whether it adheres to GAMP 5 regulations).

A ***QA representative*** is also indispensable in the client's internal validation process. Finally, the ***sponsor*** from the client acts as the system owner and is responsible for the final software approval.

Main steps of the project

Below are described the main points of the approach, once again from the supplier perspective.

Offer: From the supplier perspective, the very first step of the project is the offer. The client provides a description of the need, usually a routine experiment that requires a statistical report to be created but the necessary statistical resources are lacking. Hence, they usually need a computerised system that allows a non-statistician user to generate the statistical report. Based on the information received, the supplier must understand the need, design a solution, and define the cost by estimating the amount of Full Time Equivalent (FTE) needed for the project. This can be one of the more difficult stages of the project. To give an idea, projects I work on usually require up to 150 FTE and have a global cost up to 100k € and a period of approximately 8 months, but of course this is very project-dependent and only covers the supplier side of the project.

User Requirements Specifications (URS) / Functional Specifications (FS): These requirements define the intended use of the system. The URS is a critical document that defines the software (e.g. what the system must do and must not do), non-software (e.g. SOPs), and hardware requirements. The FS provide a written definition of what the system does and what functions are provided. [3]

As Statistical End-User Tools are small systems, the URS and FS can be included in the same document. [3] The supplier writes this document in close collaboration with the client's project leader and users.

The first (pilot) release: A pilot version of the Statistical End-User Tool is produced and delivered to the client as soon as possible, usually before the end of the URS/FS writing and usually before all the functionalities are programmed. The principle is that the earlier the client sees the tool working, the faster the URS/FS will be completed. From my experience, the writing of the URS/FS is one of the most time-consuming steps in the development because it is a complex task and involves a larger number of people than other development stages. Using the pilot or observing how it is used helps the developer to understand the client's needs.

A demonstration should be sufficient in cases where the pilot version cannot be installed by the client.

Development: Tool development begins before the end of the URS/FS writing, and finishes at the approval of the development qualification report. Similar to Agile project development sprints, development is iterative, so there are several releases between the pilot version of the tool (version 0.1) and version 1.0 delivery. The tool is split into three main units. The UI groups display screens to allow users to enter data, define the analysis parameters, and get the statistical report. The statistical core groups all of the statistical analysis programs and prepares the data for reporting. The publishing component creates the statistical report files.

Validation: The main objective of the validation is to demonstrate that the system is compliant with necessary guidelines and fit for its intended use. Software validation is a requirement of the Quality System regulation published in the Federal Register on October 7, 1996 (which took effect on June 1, 1997). [4] The tool often has to be used in line with strict regulations (e.g. GxP guidelines if used by a pharmaceutical company), so the validation process is based on the GAMP 5 guidelines edited by the International Society for Pharmaceutical Engineering (ISPE). The GAMP is a technical subcommittee of the ISPE, and produces the guidelines for manufacturers and users of automated systems in the pharmaceutical industry. The purpose of the guidelines is to provide a cost-effective framework of good practice to ensure that computerised systems are fit for use and compliant with regulation. [5]

Several software categories are defined in the GAMP5 guidelines. Because a Statistical End-User Tool is always a custom solution developed on request, it falls into the highest category (category number 5), requiring the most complete and complex validation process.

One of the first steps in the validation process is to edit a risk assessment document (for this step the supplier is not concerned). The supplier then has to be audited to verify their ability to produce a compliant solution. Before the official process of validation can begin, there are some additional pre-validation steps:

- Self-validation: the developer always tests the codes he produces
- Release validation: The tester has to validate each release of the tool before delivery to the client

The activities, procedures and responsibilities for establishing the adequacy of the Statistical End-User Tool are defined in the Validation Plan. This document must be approved prior to commencing formal validation activities, and serves to guide all subsequent validation activities.

The tool has to be tested in an appropriate test environment. To define the calculation qualification tests and unit tests, the supplier writes a development qualification document. Each statistical calculation is verified independently from the tool by another statistician, using the same or different statistical software. Each component of the tool is tested: there are specific tests for the graphical user interface, the statistical core of the application (different from the calculation qualification), and the statistical report generator. All tests are linked to the URS/FS, and some are listed in a Traceability Matrix (described later). Moreover, all the source code has to be reviewed by the tester or an expert (different from the statistician/developer). In my opinion, it is a good idea to perform multiple dry runs of these test forms. Users write some tests, execute them on the last release version, and the results are discussed. Potential anomalies are backlogged and repaired where necessary, with the objective that the version 1.0 release passes the validation process.

The tool must also be tested in the production environment. This phase of testing is performed by the client and involves: (i) an Installation Qualification to verify that physical hardware and software components are installed and configured correctly; (ii) an Operational Qualification to verify that the system's user interface, functionalities, limits and error handling, data backup, and security system operate in an expected manner; and (iii) a Performance Qualification involving User Acceptance Tests to verify that the required business functions are operating in a manner suited to real-world circumstances and usage. [3]

CHOOSING A CONTRACT RESEARCH ORGANISATION (CRO) AS THE SUPPLIER OVER A GENERAL SOFTWARE EDITOR

Pros and Cons of choosing a CRO for such a project

CROs do not usually offer software engineering services and therefore it may be difficult to find a CRO willing to develop a Statistical End-User Tool.

A statistical programmer or a statistician who is insufficiently trained in application development could result in software that is poorly documented, hard to maintain, and that does not meet the needs of its intended users. [6]

Statisticians working within CROs are likely to have more experience and understanding of a client's needs because they regularly generate the kind of statistical reports used by clients. Project statisticians in a CRO are also more likely to be accustomed to the frequent short-term deliverables required in Agile projects, and to working with client satisfaction as a priority. Furthermore, as CRO statisticians work for distinct clients, they have to follow distinct SOPs, and are therefore adaptable. CROs are therefore best-placed for providing Statistical End-User Tool development.

Pros and Cons for the CRO

By accepting to develop a Statistical End-User Tool, the CRO will engage in a long-term relationship with the client and be responsible for any repairs or further validation required.

Underestimating the amount of work for computerised system development is a risk to the CRO, especially with the Agile's approach where the client is encouraged to add or change some functions after the beginning of the development. Moreover, proposing this kind of service may make it necessary for some of the statisticians involved to be trained in software development. Nonetheless, CROs regularly use computerised systems (for example SAS®) and their statisticians must be trained in and follow the GAMP 5 principles for computerised system validation.

One challenge for the CRO is to keep statisticians motivated, which can be achieved through proposing career path evolution incentives. Moreover, based on CDISC standards and the increasing importance of metadata, the projects statisticians have to manage become more and more technical. This means that investment in training for statisticians motivated by development may help them to manage advanced projects with technical dimensions, such as Integrated Summaries of Safety/Efficacy (ISS/ISE) and submission packages.

From Statistician/Statistical programmer to Developer role

The transition from a statistician/statistical programmer to a software developer is not always a conscious act. A statistician may begin to automate some processes by programming some SAS® macros or utility programs shared with colleagues. After that the statistician may propose a small tool for corporate-wide use, perhaps not realizing they are now equipped with basic software developing skills. However, there is still a lot for the novice developer to learn at this stage, and ignorance or imperfect knowledge may result in an application lacking in quality and insufficient for use in the pharmaceutical industry. [6]

Indeed, this evolution is not without any risk but, in my opinion, the development of a Statistical End-User Tool is exciting and fulfilling because the developer must get into the mind-set of the end-user to fully understand how they

work and what they need from the tool. Moreover, I find it rewarding to participate in the creation of a computerised system as it adds an artistic dimension to our day-to-day work.
To succeed in developing a Statistical End-User Tool, the developer has to be at the interface between the users, IT, and the biostatistics domain, while knowing the roles, functions and constraints of each aspect.

To summarise, a statistical end-user tool is a computerised system allowing a non-statistician to edit a statistical report. This kind of software is composed of a UI, a statistical core, and a publishing unit. The method of development we use for such a project is freely inspired by the Agile manifesto and validation follows the GAMP 5 principles. Now we know the basics of a Statistical End-User Tool, let us investigate an example to help understand their capabilities and illustrate how to manage their development.

PRESENTATION OF THE TOOL DEVELOPED FOR THE PHUSE PAPER

I developed the following example of a Statistical End-User Tool for the PhUSE conference and writing of this article. All the source code is provided in the appendix of this article, and each step of the project will be detailed.

Medical laboratories often need to assess the agreement between two measurement methods, for example, if they need to change from one method to another, evaluate a new or alternative method, or quite simply if they have an alignment problem between two instruments. Thus, a tool is needed to measure and appraise the differences as well as the causes of these differences. For this purpose, I recommend developing a custom Statistical End-User Tool.
For our example, let us imagine I receive the following request:

A new client asks us to develop a Statistical End-User Tool for method comparison. The tool has to run on the Windows® operating system, the expected UI must be MS Excel, and all statistical analyses have to be done using SAS. The intended users have no knowledge of statistical programming.

For most applications I have been involved in, the development was within a client/server application using Java™ screens as the UI, but for the purposes of this article it was easier to automate a tool that uses Excel as the UI.
In doing so, to install the pilot provided in appendix, you just need a computer with the Windows operating system with SAS (9.3 or later), MS Word, and MS Excel software installed.

THE OFFER TIME

As the supplier, we have to provide the client with an estimation of the global cost of the project. To do this, we have to imagine a solution, anticipate any potential technical problem, identify all the deliverables, and then estimate the corresponding FTE.
In our example, the demand is clear. The client is expecting a computerised system running under Windows, using Excel as the UI, and SAS for the statistical calculation.
For the offer, we must also clearly understand what type of data will be analysed and the statistical methodology required.

When we have the contract, our first activities will be to schedule project meetings and commence the URS/FS writing. From my experience, clients appreciate it when we return to them quickly with a first draft of the URS/FS. Although each project is unique, the structure of this document remains the same. The URS/FS writing is very time-consuming in a Statistical End-User Tool development project. The statistician/developer has a key role in this task, ensuring that the supplier and client are aligned and agreed on the presented solution.

For the development we will have to design the Excel file and code Visual Basic for Applications (VBA) procedures and/or functions. The statistical methodology must be clearly detailed and documented, and several SAS programs will be required to import the data for analysis, perform checks on the data, analyse the data, and prepare the report. We will also have to help the client design the report, create its template, and automate the insertion of observed results into the template. For the development purpose we will have also to create some script in order to connect the distinct units (i.e. UI, Statistical Core, and Report Publishing), handle exceptions, and manage the processing log and the audit trail. We will also need to backlog all decisions made during the entire development process and write the (often substantial) technical specifications.

Another key point for the development offer estimation is whether external expertise is required. If the supplier does not have all the necessary internal resources in place, external expertise can be very expensive.

Validation is another substantial part of the project. The time needed for self-validation by the statistician/developer must be taken into account, as should the tester's work (including development test and calculation test design), the writing and execution of the main release and intermediate releases, and the independent code review tasks.

For the present example, if data to be analysed are very standardised, I would estimate a total of 100 FTE for a global coast of 70K €. If we find this time has been overestimated, the client is only charged for the amount of work performed. However, if the time required is underestimated, obtaining an amendment can be complicated, which is why the offer estimation must be carefully calculated.

THE KICK-OFF MEETING (KOM)

The KOM allows the supplier to show the client their ability to take charge of their part of the project, it is also the time to lay the foundations of communication between the statistician/developer on the supplier side and the technical expert on the client side. The relationship between these two people will be the cement that will make software a solid and effective tool. During the meeting, the solution planned at the time of the offer is briefly presented, as well as a draft timeline for the project and a reminder of the deliverables.

After the KOM, the project is launched, and the statistician/developer must now begin the exciting task of developing the pilot version and writing the URS/FS with the client.

IMPLEMENTING THE SOLUTION

At the time of the offer, the solution to the client's request must be planned in as much detail as possible. In our example, the proposed solution will work using the process illustrated in Figure 2.

Figure 2. An Example Statistical End-User Tool

The user will enter data into an Excel sheet, define the analysis parameters, and launch the process at the click of a button. A VB script will initialise the SAS session and SAS will perform the statistical calculation. After the check of the SAS return code by the VB script, Word is launched to prepare the statistical report, and then the process returns to Excel and alerts the user that the report is created.

In the next stage, we have to develop the pilot version of the tool from our planned solution, to help users in the URS/FS writing. Let us explore the technical challenges we have to overcome to succeed in developing this tool.

THE USER INTERFACE

The UI of a computerised system groups together everything needed for the user to issue commands. For a Statistical End-User Tool, this often involves display screens and help messages.

We do not usually use Excel for the UI, but our clients are familiar with this program and it tends to reduce the amount of IT support needed, which usually means that the pilot version is produced faster. Another good reason is that this tool will exist only for purposes of demonstration for PhUSE, so it seemed important to minimise the prerequisites in case a reader of this article would like to install this pilot.

In our example, I will therefore create a XLSM file, transform one sheet within this file into a data entry form and manage the protection. Some VBA functions and procedures will also need to be developed to manage the functions of this unit and launch the process. In this way, the user will have access to a single sheet at the opening of the Excel file, which presents itself as an input form with a button linked to a VBA procedure. Clicking this button will launch the VB script.

The provided example

After installing the pilot, the XLSM file will be located at <MethCompRoot>\PhUSEMethComp\DataEntry\ with <MethCompRoot> equal to the path you choose during the installation (the default is C:\PhUSEMethComp\DataEntry). The structure of the workbook and the VBA codes are protected by a password, which is 'PhUSE4Ever'. In the VBA project of this workbook you will find the function RunVBSFile, which executes a VBS file from Excel using a WScript.Shell object.

THE STATISTICAL CORE OF THE TOOL

The statistical core of the tool groups all of the statistical analysis programs and prepares data for the reporting.
Every Statistical End-User Tool development project I have been involved in has used SAS for the statistical core. Even though regulatory authorities do not want to impose it, in my opinion, SAS is the gold standard for statistical software.
In our example, the client has asked us to use SAS. However, because users of the tool may have no knowledge of SAS programming, SAS must be used in batch mode. As you probably know, there are different ways to run a SAS session. The two best-known are the SAS windowing environment and the batch mode. The batch mode allows the SAS program to run without any window, and is useful for automating and scheduling recurring tasks on a server. Batch mode can also be used to automate processes that cross several different applications because batching allows streamlining, documenting, and reduces the likelihood of error.
For this example, I have used a VB function to launch SAS. In this way, the SAS session will be initialised by the VB script, which launches the main SAS program like an autoexec.

The provided example

After installation of the pilot, the SAS components (programs, macros, and dictionaries) will be located in the 'SAS' folder and VB scripts in the 'VBScripts' folder.
The 'VBScripts\RunSASBatchModeLib.vbs' file contains all the functions linked to the SAS session management (initialization and checks). Here, you can find how to query the Windows registry to identify the path to the SAS executable, how to initialise the SAS session in batch mode, how to retrieve the return code for the session, and how to parse the SAS log file.
The 'SAS\Program\MethCompMain.sas' file is the SAS program called at session initialization. This program can run in batch mode or in interactive windows mode. Notice how SAS gets the session parameters when it runs in batch mode, and how these differ from window mode.
The 'SAS\Macros\CheckThat.sas' file contains the CheckThat macro that shows how the checks on input data are externalised using the 'SAS\Dictionaries\SASChecksOnData.csv' dictionary.

THE PUBLISHING

The statistical report files could have also been created using SAS but, from my point of view, choosing a specialised software such as Word is a better solution. Certain objects, such as tables of contents or field codes, are easier to manage directly in Word, as well as the conversion of the report into different formats (e.g. .doc, .pdf). One constraint of this choice is that the report must be highly standardised so that the variable parts can be managed across fields. Nonetheless, several templates can be designed for the statistical report so users can choose the report style most appropriate for the data in question.
Another great benefit to using Word is its Mailings function.

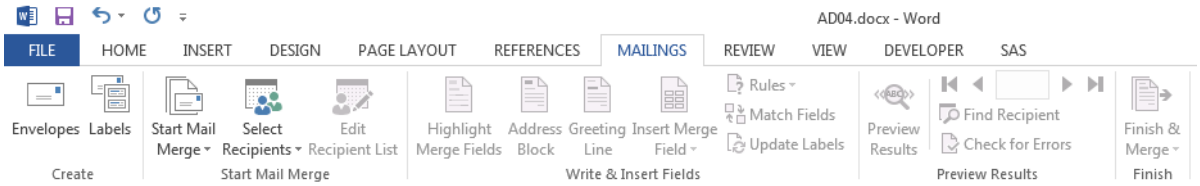


Figure 3. Mailing functionalities in the Word ribbon

Essentially, a Word document is created for use as a template of the statistical report (Main document) and during the tool process SAS will create a CSV file (Mailing list) with the results and the values of fields inserted in the template. By using VB functions, the template and CSV file can be merged, the fields updated, and a DOC and a PDF rendition of the report produced. Field values can be a number, a character string, or the path of an image file or an RTF file containing a table to include in the report.

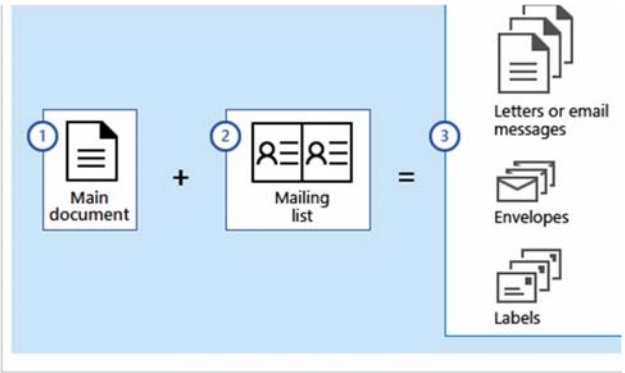


Figure 4. Word's mailing process

The provided example

After installing the pilot, the Word template file will be located in the 'Template' folder and the VB scripts in the 'VBScripts' folder.

In the 'Template\PhUSEMethCompTemplate.docx' file you will see how the results insertion is managed, how a RTF file produced by SAS is inserted to give a table, and how a SAS image file is inserted to give a figure.

The 'MsWordMailingLib.vbs' file contains the VB procedure I developed to manage the Word mailing functionalities.

INSTALLATION AND USE OF THE TOOL

The installer of the pilot version can be found in the appendix of this paper, along with instructions for the installation.

A shortcut to the XLSM data entry file will be generated on your computer.

The prerequisite are the following: Windows as Operating System with a working script host configuration, Office pack including Excel and Word (2007 or later), and SAS (9.3 or later) at the end of installation these prerequisites are tested.

SOFTWARE DEVELOPMENT SPECIFICITIES, DOCUMENTATION, INSTALLATION AND MAINTENANCE

This section of the article covers some of the software development specificities, a description of the deliverables for a Statistical End-User Tool, the different documents required, and the components within a version release.

I will then discuss the installation of the tool, the guide, the setup and the qualification process and an evocation of the operation management, the maintenance system, the review and change controls.

SOFTWARE DEVELOPMENT SPECIFICITIES

Below are detailed some of the software development specificities that are likely to be unfamiliar to a classical data scientist.

Versioning

Versioning is the creation and management of multiple releases of a product, all of which have the same general function but are improved, upgraded, or customised. For a Statistical End-User Tool we choose the standard GNU version numbering scheme consisting of two or three numbers separated by periods (e.g. 1.2.1). The first number, called the major number, is increased when there are significant improvements or changes in functionality. The second number, called the minor number, is increased when there are minor feature changes or significant fixes. The third number, if it exists, is called the revision number. It is increased when minor bugs are eliminated.

Following this format, the pilot, or first delivered release, would be version 0.1 of the tool.

Access control

Linked to the regulated context of their use, access to the Statistical End-User Tool must be secured and limited to authorised users. [4] It is recommended that each user of the computerised system has an individual account and individual password. The system should also be designed to record unauthorised access attempts.

Audit trail

The audit trail provides documented evidence of the sequence of activities performed by the Statistical End-User Tool, this being a requirements of the 21 CFR 11 regulations. [4] The automated audit trail typically operates in a privileged mode, in which it can access and supervise all actions from all users without a normal user having access to stop or change this. Similarly, a trail file or database table with a trail should not be accessible to normal users. The idea is that any time something significant happens you write some record indicating what happened and when it

happened. Although there are no rules for Statistical End-User Tools, most of the time we use a simple ASCII file. This computer-generated time-stamped audit trail is secure and non-perishable.
In most Statistical End-User Tools, the software log that records all user actions is distinct from the audit trail that records only actions involving statistical report creation. The software log is perishable, the audit trail is not.

Externalization and dictionaries
When working on a computerised system, it is important to keep its maintenance in mind. Software developers have to follow strict programming practices. One of the most important things to remember to ease future maintenance is the externalization. The best example here is the handling of exceptions (e.g. errors). Because a Statistical End-User Tool is always made using multiple programming languages, the use of a built-in solution to manage exceptions is not the best idea if you want them standardised whatever the unit of the tool. In this case, I would recommend externalizing the information in a dictionary. Such a dictionary could be an object, a DLL, a table of a data base, or (in my experience) a simple ASCII delimited file. By using externalization of a priori information, time can be saved for technical analysis of a problem detected by a user, for the implementation of the solution, and for the validation of the modification.

Choosing the right programming language
A Statistical End-User Tool always involves multiple programming languages. As a SAS programmer I began my career as a statistical tool developer by using only SAS features. I used SYSEXEC or X commands to manage external files and folders, SAS/AF with the SCL language for the screen building, and data _NULL_ steps to create an HTML Page. Moreover, I used many ODS functions and the 'DOCUMENT' procedure to publish the reports. However, experience has taught me that scripts (kornshell, JavaScript…) are a better solution than SAS for managing external files, folders, and permissions. The choice of the programming language is often driven by the developer's previous experience and available resources, but the developer must also consider which language is most efficient for the tool.

ACCOMPANYING DOCUMENTATION

The following documents are always required for a Statistical End-User Tool. Additional supplemental documents can also be required, depending on the project.

User Requirement Specifications and Functional Specifications
The URS essentially describe what the end-user needs the computerised system to do, whereas the FS describe the functions of the computerised system. The requirements included should follow the "SMART" guidelines, that is they should be Specific, Measurable, Attainable, Relevant, and Testable. [7]

Validation Plan (VP)
The Validation Plan has to be approved before the beginning of validation activities. It defines the activities that are required, the responsibilities, the way to perform the validation, all the deliverables, and the documents linked to the validation process.

Risk Assessment (RA)
The GAMP 5 guidelines used for the validation process involve a risk-based method. Importantly, only the client should participate in the writing of the RA document, not the supplier. The RA identifies risks associated with the implementation of the tool in its ultimate real-world usage.

Technical Specification (including Design Specification)
This is the biggest document the supplier has to write. It contains the system requirements (hardware and software specifications), a presentation of the computerised system, a detailed statistical methodology, and descriptions of the main processes (audit trail, generation of report identifier, error handling), the most complex algorithms, and each component, dictionary, and input/output files. The document also contains a guide for installation and uninstallation. This document is written for users to troubleshoot problems and maintain the tool, and this must be remembered during its writing.

Development Qualification Report (including Calculation qualification and Unit Test)
This document compiles all the test forms completed by the tester for the main releases of the tool (i.e. the version 1.0, and any subsequent version after a Change Control).

Code Review
A code review is required in the GAMP 5 guidelines. All the programs have to be clearly annotated by the author. The code review must be planned as early as possible so that errors are detected early before they can cause problems. A checklist is used to carry out the code review, which is performed by an independent developer.

Installation/Operational/Performance Qualification
The Installation Qualification verifies that the hardware and software components have been correctly installed and configured. The supplier usually provides the client with an automated utility tool to perform this validation step, with

the resulting completed test forms added to the Installation Qualification Report and main user Qualification Document as appropriate.

The Operational Qualification verifies that the system operates as expected. This includes testing of the UI, input and output (statistical report, log, audit trail,…) documents, the functions defined in the URS, the limits of the system, and the exception handling. The Operational Qualification also verifies the backup functionalities, recovery, security, and access control.

For the Performance Qualification, User Acceptance Tests must demonstrate that the Statistical End-User Tool functions are operating in a manner suited to real-world circumstances and usage.

Traceability Matrix (TM)

The TM links the URS, FS with the corresponding test protocols, and is used to outline project requirements and ensure they are met. TMs usually take the form of a table that tracks the requirements and specifications for testing as well as the development of testing documents. The TM should be verified at each stage of development to ensure that all system requirements have been adequately tested.

Supplier Assessment

The Supplier Assessment verifies that the Quality Management System of the supplier is appropriate for the tool development, and that the supplier is capable of providing the expected documentation and quality services. This document is written by the client (i.e., the regulated user) and is between a simple questionnaire and an audit.

Validation Report (VR)

The writing and approval of the VR corresponds to the transfer from the project stage to the operational stage. The VR should document the completion status of validation work as defined in the Validation Plan, any failure or unplanned change from the VP, the completion of training programs, and that the system has been reviewed and assessed as fit for its intended purpose. [3]

Other documents

Linked to the use and maintenance of the tool, the client has to write SOPs covering all aspects of the tool's operation. For example, the client may consider writing an SOP describing the installation of the tool, an SOP detailing how the tool will be maintained, and an SOP for how a potential change control will be managed. A further SOP should describe how future users will be trained. [8]

FOCUS ON THE INSTALLATION OF THE TOOL

Installation guide/procedure

The supplier has to write a guide for the installation of the software, which forms part of the technical specification. The client may have to write a SOP based on this guide.

Installation setup

An installer or setup executable file needs to be created by the supplier to run the installation. This action is required for each new version release. Internet resources can be used to help the developer create the installer/setup file.

Installation Qualification

The Installation Qualification for our example Statistical End-User Tool is performed at two levels: at an automated level by a small utility tool, and at a manual level by the user as defined in the Installation/Operational/Performance Qualification definitions document. The last step of the installation guide requests the user to execute the utility tool that performs some automated checks, verification of the component extractions, and other prerequisites such as the existence of a registry key. This utility tool produces a ASCII or PDF log file, which is kept on the data repository of the tool and can be saved in other places depending on the client procedure. The tool ensures that the necessary system requirements are met, all the expected components have been copied, and that the expected data repository is created.

When the client subsequently executes the test forms for acceptance, there is at least one test form that validates installation and checks everything that cannot be automated.

Uninstallation

Uninstallation usually involves more than just erasing the program folder because code is generally copied/generated in multiple locations. For example, registry files and other system code may need to be modified or deleted for a complete uninstallation.

Software deployment

Software deployment involves all of the activities that make the computerised system available for use, including the installation. Software deployment is done by the technical expert (who is usually also the project leader) with help from the supplier if needed.

CONCLUSION

A Statistical End-User Tool is a computerised system that allows a non-statistician to create a full statistical report without assistance from a biostatistician. This kind of software is particularly suited to non-clinical studies performed routinely by a laboratory.

Development of this kind of software must comply with all applicable statutory and regulatory requirements, in particular the 21 CFR (specifically part 11) demanded by the FDA.

The method used for Statistical End-User Tool development is driven by the Agile manifesto principles, which prioritise client satisfaction, advise an iterative approach, and welcome changes to the plan at any time. For this reason, development and validation begin just after the kick-off meeting without waiting for user requirement approval. Early delivery of a pilot version (version 0.1) to the client is a key objective in the development cycle, as doing so significantly facilitates the later development stages.

Tool validation begins at the very first stage of development, and continues through each step of the software lifecycle. The statistician/developer has to review and test their own code ahead of validation by nominating an independent tester. The tester then verifies the development and the statistical analysis results to ensure they are exact and repeatable. Finally, the users have to qualify the installation, operation, and performance. These validation processes follow the GAMP 5 principles.

A Statistical End-User Tool is usually composed of:

- a UI unit that groups screens to allow users to enter data, define analysis parameters, and obtain a statistical report.
- a statistical core unit that performs the statistical analyses and prepares the data for the report
- a publishing unit which creates the statistical report and records all the potential renditions of the report into different formats (e.g. .doc, .pdf, etc.)
- script files to link the distinct units of the tool

In this article, I have proposed an example of a Statistical End-User Tool which allows the automated creation of a report for a bioanalytical method comparison using the Bland-Altman graphical method.

The example Statistical End-User Tool I designed links familiar software, which the client is likely to already have access to. Nonetheless, Statistical End-User Tools could just as easily be developed to use other software programs if this is desired by the client, for example R or WinNonLin for the Statistical Core in place of SAS. The links between the software units in the example are performed using VB scripts, in keeping with VBA codes that Excel and Word are designed to use. These could however be replaced with JavaScript if desired.

Regular communication between all parties is key to the success of any project. Each party should also be prepared to undertake some risk; for the CRO, changes to the development plan could risk exceeding the predicted FTE; for the client, the choice of CRO is crucial, as a tool might be produced without proper validation if the statisticians designing it are insufficiently trained in software development.

Computerised systems are increasingly used in clinical trials, encouraged by advancements in the standardization of data (CDISC). Moreover, in line with the GAMP 5 principles, CROs increasingly have to implement SOPs for qualification of their own software packages. Perhaps, therefore, CROs are in a better position to offer Statistical End-User Tool development than ever before.

Feel free to install the tool provided in the appendix to investigate how it works and to understand some of the functions described – for example how to launch SAS from Excel, how to externalise error handling using a dictionary, and how to externalise publishing using the mailing functions of Word.

REFERENCES

[1] Kent Beck, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler, Brian Marick (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010. Available at: http://agilemanifesto.org/

[2] A. Hajou, R. S. Batenburg, and S. Jansen, “Method æ; the agile software development method tailored for the pharmaceutical industry”, Utrecht University, the Netherlands. Available at: http://method.ae/

[3] “Guidance for Industry: Computerized System Validation version 1.0”, Drug Office – Department of Health. Available at: https://www.drugoffice.gov.hk/eps/do/en/doc/guidelines_forms/Guidance%20for%20industry_CSV_201312.pdf?v=0tgvviy66r

[4] “Code of Federal Regulations Title 21 Part 11 Electronic Records; Electronic Signatures”. Available at: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=11

[5] N. Vishal Gupta et al, “A Review on applications of GAMP -5 in Pharmaceutical Industries”, Int. J. Drug Dev. & Res., July-September 2013, 5 (3): 4-16. Available at: http://www.ijddr.in/drug-development/2a-review-on-applications-of-gamp-5-in-pharmaceutical-industries.pdf

[6] Jeffrey Abolafia, Chapel Hill, Carrie Dundas-Lucca, “The Rewarding (and Rocky) Road from SAS Applications Programmer to SAS Software Developer”, PhUSE 2015 paper PD11. Available at: http://www.lexjansen.com/phuse/2015/pd/PD11.pdf

[7] Paul Daniel, “Using the ISPE’s GAMP Methodology to Validate Environmental Monitoring System Software”. Available at: http://www.vaisala.com/Vaisala%20Documents/White%20Papers/CEN-LSC-G-Using-GAMP-Methods-to-Validate-CMS-Software-B211370EN-A.pdf

[8] “Guidance for Industry: Computerized Systems Used in Clinical Investigations”, FDA, 2007. Available at: https://www.fda.gov/ohrms/dockets/98fr/04d-0440-gdl0002.pdf

ACKNOWLEDGMENTS

I would like to thank Kurt Liittschwager and Jonathan Pitt for their work in English translation of the draft of this paper written in FrEnglish, and my supervisor, Mohamed Amakrane, for his confidence and his incentive to always go beyond my limits.

RECOMMENDED READING

Alistair Dootson, “Agile Project Management in a Regulated Environment”, PhUSE 2014 paper AD06. Available at: http://www.lexjansen.com/phuse/2014/ad/AD06.pdf

CONTACT INFORMATION

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

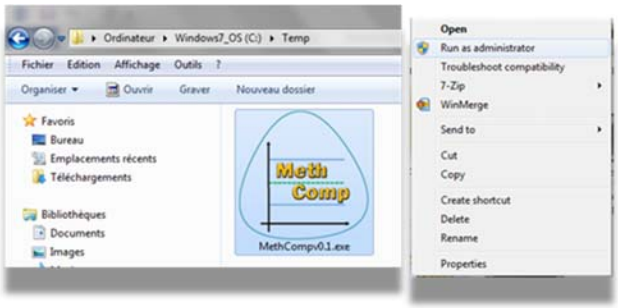
Mickael Borne
4Clinics
18-26 Rue Goubet
Paris / 75019
Work Phone: 01 84 19 35 59
Email: mborne@4clinics.com
Web: www.4clinics.com

APPENDIX I: PILOT OF THE TOOL DEVELOPED FOR PHUSE

I have to apologise that the pilot is not as successful as I would have liked due to lack of time. Nonetheless, once you have installed this sample statistical tool you will find most of the main features have been programmed.

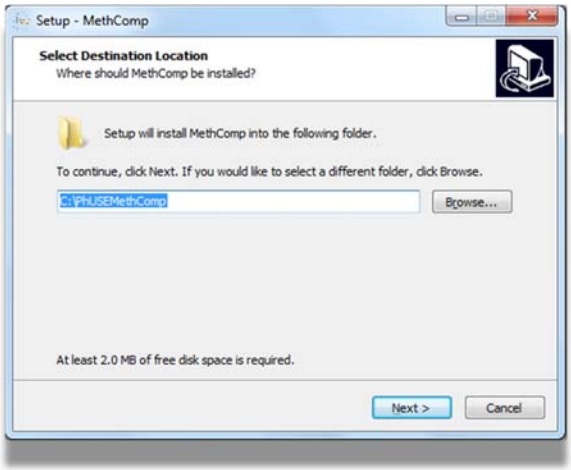
How to install the pilot of the PhUSE statistical end-user tool

1. Before you begin installing MethComp, ensure that you log on using an administrator account.
   Download the installer at the following address:
   https://drive.google.com/drive/folders/0B2sIht8FC_k5RHp5dE0yOXlSNzQ?usp=sharing
   In case of any trouble don't hesitate to email me.
   To start the setup of installation, browse to 'my computer/this PC' and right-click on the appropriate exe file, choose to Run as administrator.

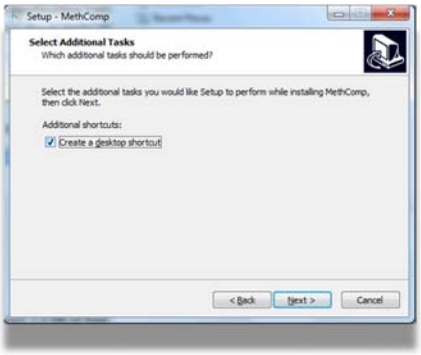


2. Select your preferred language for the setup of installation.

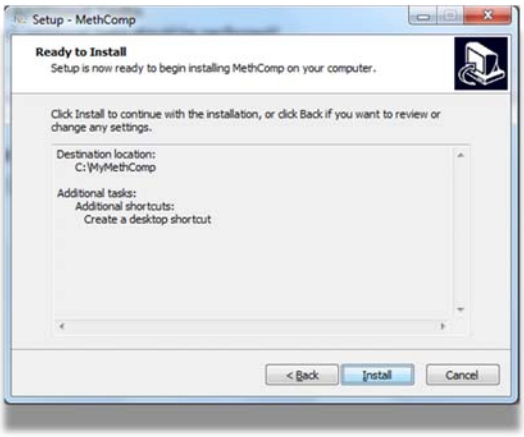
3. Select the destination location



3. Check the ‘Create a desktop shortcut’ box and click Next.

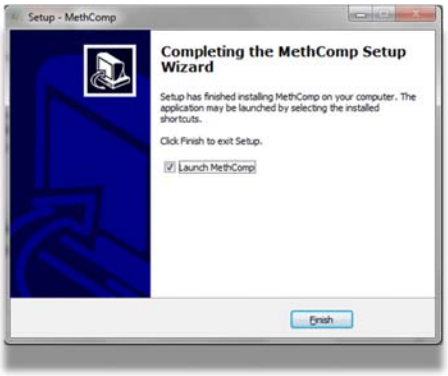


4. Click **Install** to start installation.

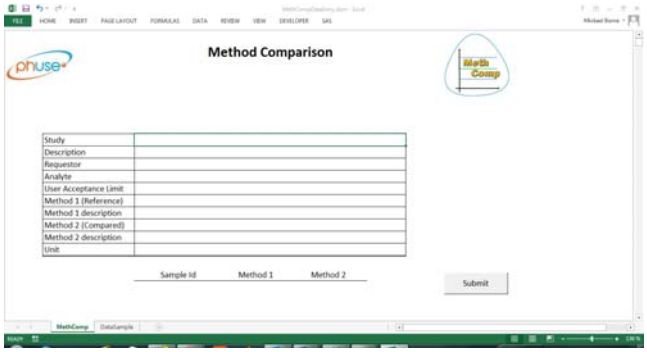


5. At the end of installation a qualification tool is launched to test that the prerequisites are correctly installed and configured (WSH, MethComp registry key creation; SAS version and the copy of all the components). The following pop-up box will appear to confirm successful installation.

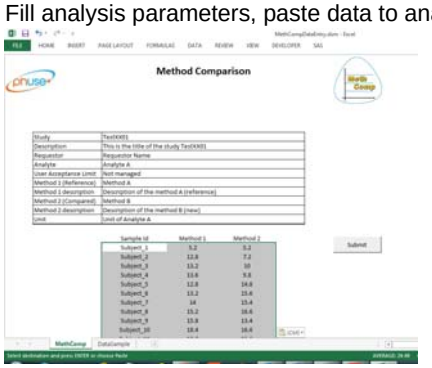
6. Finish installation, click **Finish**.



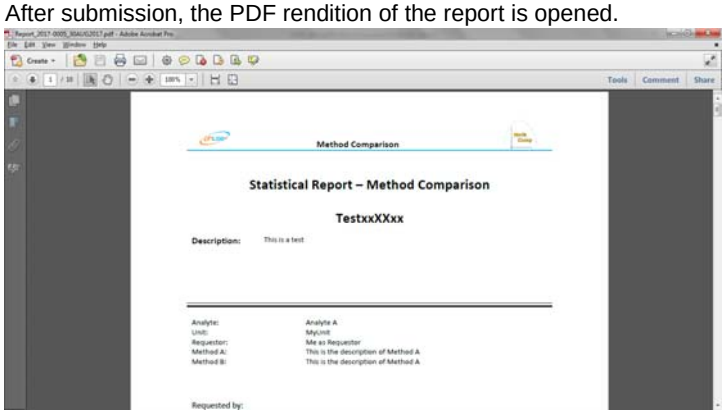
7. If the **Launch MethComp** box was checked in the previous screen, the DataEntry Excel file will open after installation.



8. Fill analysis parameters, paste data to analyze and click the **Submit** button



9. After submission, the PDF rendition of the report is opened.



10. To uninstall browse to 'my computer/this PC' and right-click on the appropriate exe file, choose to Run as administrator