

# Welcome to Dullsville!! The System Testing of SAS Macros

James McGiffen, GlaxoSmithKline, Greenford, UK

## ABSTRACT:

GSK has recently undertaken a project in which there was a requirement to test a new suite of SAS macros that created analysis ready datasets. This type of work presents both technical and managerial issues to not only our company but the entire pharmaceutical industry.

This paper presents and discusses the approach that GSK has taken to solve this challenge so that we can produce high quality, reusable testing in an environment which is technically challenging and motivating for the programmers involved. It also stresses the need to develop a high quality, rapidly repeatable series of tests so that the quality of the development does not suffer.

## INTRODUCTION:

Working in a regulated industry requires us to document the testing of any code used to produce clinical reports.

This provides the pharmaceutical industry with a problem. Programmers are often asked to do this work but programmers generally do not like to test! GSK faced this problem recently when it undertook the testing of a new batch of SAS macros. This paper discusses the approach that was taken in both technical and process terms.

## COMMON ISSUES WITH TESTING:

The general feeling of many programmers is that testing is dull, tedious and laborious. There is the danger that when programmers are testing they spend time writing or completing documentation, a role which suits few of them. This can lead to motivational issues within a programming team and can lead to incomplete testing.

The testing of code is time consuming and often this effort is underestimated in project plans where timelines and resources are focused on the development of the tools.

## ISSUES OF TESTING WITHIN THE PHARMACEUTICAL INDUSTRY:

There are specific issues within the pharmaceutical industry which means that a typical software development lifecycle is not suitable.

This pharmaceutical industry is a rapidly changing industry with new sources and types of data and study designs, therefore any system that handles this data needs to be changed and therefore easily, rapidly and accurately retested.

There is also the requirement for documented proof of testing for whenever an auditor knocks at the door. This can increase the need for paperwork which, as discussed above, may not be the programmer's strength.

Accuracy is also of paramount importance in this industry as the results of incorrect code can be fatal.

## THE SPECIFIC PROBLEM WITHIN GSK:

To understand our approach you have to be aware of the challenges that the project team faced. The initial testing requirement was to validate code which transformed data from our operational databases to a CDISC compliant standard. We therefore had to resolve the following issues:

1. The macros produced analysis ready datasets and we had to ensure that the output was correct. These macros had to work on all phases of studies and in all therapeutic areas, therefore the data could

originate from a Phase 1 clinical pharmacology study with a complex dosing regime, to a long term oncology study lasting several years.

2. The staff assigned to undertake the testing was a team of SAS study programmers and macro developers and whose skills were more suited to programming than generating and completing documentation.
3. Due to the modular design of our system the macros being developed were being called by others and incorporated into an existing set of tools. Therefore we had the additional complication of ensuring that we understood, documented and tested the interaction between the individual macros.
4. Our new tools had to work on all types of study designs that are used within GSK; the amount of testing required for one macro would be multiplied by the number of study designs we wanted the tools to be available for.
5. We had over 50 individual units to test. The number of tests multiplied by the interactivity and different study issue meant that we had the prospect of running thousands of tests.

Therefore at the start of the testing phase within the project we were faced with the prospect of many hours of long, tedious and repetitive testing.

Due to the volume and the deadlines of the project, the temptation was to just immediately start writing scripts and laboriously working our way through them.

### **APPROACH USED WITHIN GSK:**

The approach used within GSK did not use any complex tools or software development packages. We simply broke the problem down and use SAS to solve the issues:

1. We created two macros to capture the metadata of the input and output datasets. These were then used along with Proc Compare to determine what changes were made to the datasets.
2. Our team was separated into two, the developers who developed the tools and the developers who tested the tools. Our testers used the Unit specifications of the macros under test in order to duplicate the output (our QC dataset) from the macros. This was done by either by replicating code or producing known output. The testing macros then automatically compared the input, output and QC dataset.
3. We isolated each macro to ensure that we tested only the interactions that occurred. Our testing tools checked for any changes to Global Macro variables and the contents of the work directory so that we could confidently rule out any interaction.
4. We created an additional program to automatically run all of our tests. This program created directory structures, copies the test code, run's the suite of tests, captures error messages and produces pass/fail reports. All the repeatable testing processes were handled by this program.
5. As the tests were run automatically and we could add new test studies by simply increasing the number of iterations that the code went through.
6. The same testing macros were used in every test.

This approach gave us the following benefits:

1. We reduced manual error in checking data. SAS did it all for us.
2. Our programmers were programming, not writing and running test scripts therefore we were using their strengths rather than their weaknesses. The interest and challenge to the testers remained high.
3. We could confidently reduce the amount of testing required allowing for faster retest of and re-release of code.
4. We had consistency of testing across all our units thus could accurately follow our test plan because we were allowing SAS to do it for us.
5. We could have different test studies in our informal test development area and our formal test areas therefore, by default, test the tools on double the amount of studies. As the testers developed their tests they found issues with the macros which were resolved, then when we moved to formal testing a smaller

number of issues were found and the code was more likely to pass first time reducing the need for additional documentation.

6. Once the programmers gained experience of how to use the testing programs, they could concentrate on testing the macros knowing that the majority of the documentation is done for them.

This approach was also successfully implemented in another project that was developing a system to handle and report Pharmacokinetic data.

### **WHY IT'S IMPORTANT TO BE ABLE TO RAPIDLY RETEST:**

In addition to the issues mentioned above the quality of your software development is often compromised if your testing cycle takes too long. This is a very important issue that is often overlooked.

An example of this can be demonstrated in a development that has five very similar macros. The first four pass their testing, the fifth fails on an obscure issue which may affect the other four. As they have already passed your testing there may be little enthusiasm or time within your project timelines to update and retest, especially if this process takes too long.

The ability to make small incremental updates to systems is vital if it is to remain a useful tool within your organisation.

### **LEARNING GATHERED FROM THE APPROACH USED WITHIN GSK:**

This approach has been deemed successful within GSK. It has, as mentioned above, been used on two projects, one which is currently in development. We are also planning to migrate the testing from previously released reporting tools to use this methodology. We also found issues that we would not have encountered using a more manual process. We have learned that:

1. It does not completely negate the need for paper. There is the perception that more paper = more testing.
2. The tools need to be adaptive and flexible. Like the tools under test, the testing tools need to be able to be updated in order to accommodate other testing requirements. The tools, however, do need to be subject to formal change control procedures
3. The first creation of the test script may take longer than a more manual process but there are definite savings in the retesting of the macros.
4. We found that the main 'testing' happened when the test developer was writing the test programs. The finalised output was to signify the fact that testing was completed. All formal conversation happened via the Unit Specification document and therefore we can ensure that these documents were up to date.
5. We found it useful to use different data between the informal and formal testing areas. This, as already mentioned, allowed us to test the macros on more study data but also allowed us to increase the quality of the test code by challenging it with different data.
6. It is sometimes more intellectually challenging testing the code rather than writing it as you have to test every decision in the Unit Specification. This, along with the automation of the documentation, meant that motivation remained high for developers.
7. The creation of the test code took longer than the development of the macro. We began to realise that this made perfect sense as you have to test every possible path that the program may take
8. To fully test the macros requires time and resource, which has to be acknowledged and incorporated in any project plans.

### **FUTURE DEVELOPMENTS AND USES:**

After the initial hurdle of learning how these tools work, this approach has been successfully used within a subsequent project within GSK, where more novel uses of the tools have been subsequently developed

For future developments we are considering more automated reports and tests and this will be explored on an ongoing process.

**AUTHOR CAN BE CONTACTED:**

**James McGiffen**  
**Principal Analyst**  
**GlaxoSmithKline**

Phone (+44) 020 8966 2146  
Email [James.x.Mcgiffen@gsk.com](mailto:James.x.Mcgiffen@gsk.com)