A Pragmatic Approach to the Testing of Excel Spreadsheets

Many GxP critical spreadsheets need to undergo validation and testing to ensure that the data they generate is accurate and secure. This paper describes a pragmatic approach to the testing of Excel spreadsheets using a novel procedure of 'live' testing. Although demonstrated on Excel spreadsheets, this testing approach provides a flexible solution for testing many data intensive systems, and can be successfully adapted to databases, LIMS and ERP systems.

By David Harrison & David A Howard

Key Words: Validation, Compliance, Spreadsheets, MS Excel, 21 CFR Part 11, Pharmaceutical, GAMP, GxP, GLP, GMP, GCP, End User Computing, Testing, Qualification

Introduction

This is the final article in the short series of papers describing a generic process for validating Excel Spreadsheets. This article describes the practices and procedures that can be used to test Excel spreadsheets and to ensure they are maintained in a compliant state. Previous articles in this series have given an overview[1] of the validation process and a description of the specification phase[2].

Terminology

In traditional computer systems validation the terms Functional Testing (FT), Installation Qualification (IQ), Operational Qualification (OQ) and Performance Qualification (PQ) are used as an approximation to the traditional software development lifecycle (SDLC) model used for computer systems validation.

In this approach we actually advocate a **single document** that qualifies a spreadsheet for use, and are cautious of the fact that the terminology IQ, OQ, PQ tends to imply separate unique deliverables whereas in fact they should merely signify logical stages in the qualification process. We are not aware of any regulatory requirement dictating the use of terminology IQ, OQ, PQ, or of the requirement for separating these activities. To reinforce this thinking it is our understanding that this terminology will be removed from the forthcoming GAMP[3] guidelines (version 5).

If internal company policy dictates the necessity for separate qualification deliverables then the generic process is sufficiently flexible to allow for separate deliverables. We see no benefit of having separate deliverables, it will have no impact on the quality of the final spreadsheet and the additional documentation adds to the cost of the project and makes documentation traceability more complex.

Spreadsheet Qualification

This Spreadsheet Validation process uses an amended V model, Figure 1, as highlighted in the overview article[1].

The streamlined qualification process is designed to confirm that the spreadsheet calculates accurate results, and is installed and operates as defined in the specification document[2].

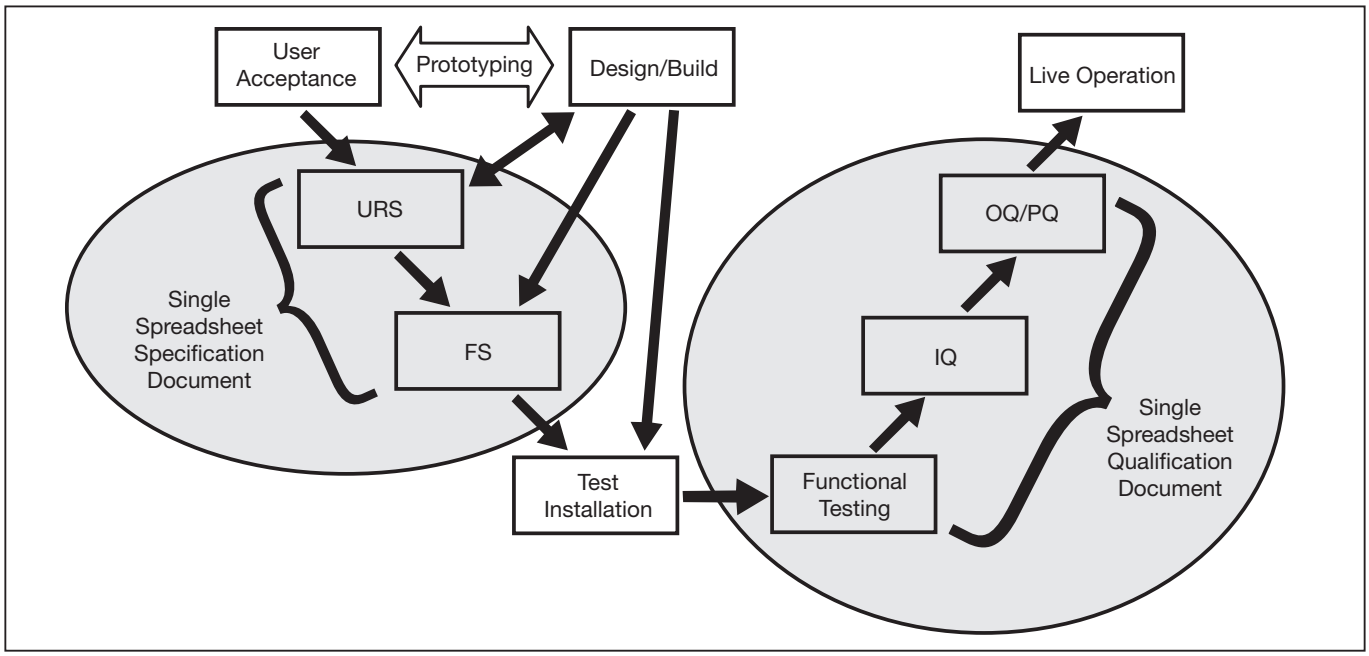

Figure 1: Amended V model for Excel Spreadsheet Validation

The validation process places significant focus upon documents that are generic in nature, allowing rapid deployment and minimising document preparation time. This is vital when applied to a large portfolio of spreadsheets.

Spreadsheet Qualification Protocol

The qualification document is designed to:

- be generic where possible to simplify generation
- be versatile and flexible both in format and scope for spreadsheets of various complexity (simple through to complex macros)
- correlate and document all 21 CFR Part 11[8] testing to remove the need for a separate document
- be adaptable to meet local/company requirements or other regulatory needs

The core document consists of the following sections:

- Main Body
- Appendix A – Spreadsheet Functional Testing
- Appendix B – Spreadsheet Installation Qualification
- Appendix C – Operational/Performance Qualification
- Appendices D & E– Additional Data Sheets (blank forms for completion)
- Appendix F – Traceability Matrix
- Appendix G – Qualification Report

The design uses appendices for the 'traditional' deliverables, and allows the flexibility to add or remove sections if required. Further details on the document sections are provided below.

Main Body

The core document text provides an overview of the spreadsheet being validated and provides information about the philosophies for testing, documentation and change control. It will also contain a glossary and references to other documentation relating to the validation.

In practice it has been found that the majority of this section will be identical for all spreadsheets within a company/department.

Appendix A – Spreadsheet Functional Testing

"Functional" testing of the spreadsheets is the key task for successful validation using this particular approach. This appendix is designed to demonstrate the correct functionality of the spreadsheet as highlighted below.

- **Testing the structure of the spreadsheet**. This ensures that the correct data is being used in the spreadsheet calculations and formulae, i.e. when you are performing a mean of 10 values, that the correct ten values are referenced in the calculation formula. This is a common error caused by drag and drop errors when generating spreadsheets.
- **Testing the calculations in the spreadsheet**. This ensures that the actual calculation formula is correct in the spreadsheet, i.e. when you have a calculation to multiply by a fixed value of 365, that it is in fact a multiplication and that the value is 365. These are common errors caused by typographical errors when generating spreadsheets.
- **Testing any macros in the spreadsheet**. This ensures that any processes controlled by macros operate as specified. Macros can be complex with multiple paths through them and errors are common in the logic used.

Usually the items above are inherently interlinked, and they can not be tested separately. However it is worth making these distinctions as it helps in understanding how to test spreadsheets, and also where to focus your effort when looking for potential errors.

For the vast majority of spreadsheets the functional testing can be executed outside of the final operating environment. Only when macros are used, or links to other spreadsheets or systems are involved do you need to worry about testing in the final environment.

The following subsections cover the functional testing and are divided into separate test scripts.

Environment Recording

As functional testing may be performed outside of the final environment the details of the test environment and spreadsheet file being tested are recorded.

We believe it is usually preferable to validate a spreadsheet template (.xlt) file rather than a workbook (.xls) file. Templates can be validated, secured and used to create multiple 'validated' spreadsheets.

It is also essential to have reliable file naming, storage, and change control procedures, particularly when testing is taking place outside of the final environment. We recommend the use of a secure file repository or write-once optical storage medium to minimise the risk of file changes between functional testing and installation.

Optional sections also record any additional controls (e.g. 3rd party add-ins) where appropriate.

Manual Testing of Spreadsheet Calculations and Functionality

The traditional 'wordy' step-by-step approach to protocol design is impractical and unnecessary when applied to the majority of spreadsheets. The step-by-step approach is **not** ideal because:

- Writing the protocol steps usually takes much longer than actually executing them and therefore you spend little time testing and a lot of time writing protocols
- Suitable test data for spreadsheets can be difficult to fully define in advance. For example:
  - envisage the situation where a calculation is to mean three values A = AVERAGE(X:Y:Z)
  - The test data that you have been presented with (perhaps from a live test run) has X=25, Y= 25, and Z=25.
  - Your resulting calculation gives you an answer of A=25.
  - Although this result is correct, it leaves some doubts as to whether the calculation was correctly structured. Perhaps it was A = AVERAGE(X:Y) or A = AVERAGE(Y:Z), this can easily happen with drag and drop errors.
  - It would make more sense to use differing data values such as X=20, Y=23, and Z=29. A correct result from this data set would be a lot less like to occur by error.

  Spreadsheets are full of situations like this, and normally a single set of test data will never fully check the structure and calculation adequately. The answer is **not** necessarily to use multiple predefined data sets however, as the complexity of spotting these situations in advance is immensely time-consuming.

Because of the concerns above, spreadsheet testing is best performed using flexible data entries to allow for differing data inputs and the ability to stress test the spreadsheet. Line by line descriptions of this flexibility is very difficult to write, it normally relies on the testers initiative and curiosity.

An Example of Manual Testing

A simple example spreadsheet is depicted in Figure 2 showing the calculation of averages for multiple rows.

	A	B	C	D	E	F
1		Reading 1	Reading 2	Reading 3	Reading 4	**Average**
2	Set 1					
3	Set 2					
4	Set 3					
5	Set 4					
6	Set 5					
7	Set 6					
8				**Overall Average:**		

Figure 2: Example Spreadsheet.

The Spreadsheet Specification described previously[2] would have specified all the calculations in the User Requirements Table appendix. In this example the specification would define two calculations as summarised below in Figure 3:

Calc. Ref.	Cell Name Description	No. Cells	Comments
A.1.1	Average	6	Provides an average of up to 4 readings for each data set
A.1.2	Overall Average	1	Provide an overall average of all **Average** values

Figure 3: Example User Requirements Table.

The 'Manual Testing' process allows the tester the flexibility to perform appropriate tests with a set of representative test data. This data is used, but can be deviated from if the tester sees situations that don't give complete confidence in the generated results.

In this example the test data is shown in Figure 4.

	Reading 1	Reading 2	Reading 3	Reading 4
Set 1	24.2	26.3	25.1	24.6

Figure 4: Example Test Data.

The goal of 'manual testing' is to **verify that the spreadsheet gives the same result as if performed by a manual calculation**. i.e. that the spreadsheet gives the same result as you get if you performed the calculation on paper or by calculator.

The tester would input the test data into the spreadsheet to confirm the calculation in cell F2. It would be apparent to the tester that the calculation in cell F2 is repeated down to F7, and therefore to shorten the process, the input can be repeated for all the identical Average cells (F2:F7) by a simple copy/paste of the same data values as shown in Figure 5. A screen shot of the results would be taken.

	A	B	C	D	E	F
1		Reading 1	Reading 2	Reading 3	Reading 4	**Average**
2	Set 1	24.2	26.3	25.1	24.6	25.05
3	Set 2	24.2	26.3	25.1	24.6	25.05
4	Set 3	24.2	26.3	25.1	24.6	25.05
5	Set 4	24.2	26.3	25.1	24.6	25.05
6	Set 5	24.2	26.3	25.1	24.6	25.05
7	Set 6	24.2	26.3	25.1	24.6	25.05
8				**Overall Average:**		25.05

Figure 5: Spreadsheet Calculation Confirmation of Cells F2:F7

To confirm the spreadsheet is giving the correct result the tester would then manually calculate an average of 24.2, 26.3, 25.1 and 24.6 (using a calculator) **and show the working and result** onto a manual calculation form (a paper form in the protocol) as shown in Figure 6 (Test No.1). This shows the tester what the correct result should be and confirms that the spreadsheet result matches a manual calculation. The tester has documented evidence of both processes/results.

Test No: 1	Calc. Ref: A.1.1	Worksheet: Calculation Sheet			Test Type
Cell(s): F2:F7	(24.2 + 26.3 + 25.1 + 24.6)/4 = 25.05				Calculation √
					Logical Test
				Printout page No(s):	3
Calculation Method:	Calculator √	Excel ☐	Manual ☐	RESULT:	PASS / FAIL

Figure 6: Manual Calculation Confirmation of Cells F2:F7

The Overall Average calculation in cell F8 is a different calculation, and the use of the test data in Figure 5 would be inappropriate. Therefore, the tester would use their initiative and verify this calculation using different data such as that shown in Figure 7. This data ensures the calculation is correctly challenged.

	A	B	C	D	E	F
1		Reading 1	Reading 2	Reading 3	Reading 4	**Average**
2	Set 1	24.2	24.2	24.2	24.2	24.20
3	Set 2	25.2	25.2	25.2	25.2	25.20
4	Set 3	26.2	26.2	26.2	26.2	26.20
5	Set 4	27.0	27.0	27.0	27.0	27.00
6	Set 5	28.0	28.0	28.0	28.0	28.00
7	Set 6	23.6	23.6	23.6	23.6	23.60
8				**Overall Average:**		25.70

Figure 7: Spreadsheet Calculation Confirmation of Cells F8

The subsequent manual calculation is show in Figure 8 (Test No.2).

Test No: 2	Calc. Ref: A.1.2	Worksheet: Calculation Sheet			Test Type
Cell(s): F8					Calculation √
(24.2 + 25.2 + 26.2 + 27.0 + 28.0 + 23.6)/6 = 25.70					Logical Test
				Printout page No(s):	4
Calculation Method:	Calculator √	Excel ☐	Manual ☐	RESULT:	PASS / FAIL

Figure 8: Manual Calculation Confirmation of Cell F8

Testing of the above example can be performed with 2 screenshots, and 2 sets of entries on the generic form. If the same example was defined by a step-by-step protocol approach, it would probably require dozens of steps, and multiple sets of test data to give the same conclusion. It is vital to note that this is a very simple example, and the more complicated the spreadsheet becomes, the more difficult it is to write step-by-step protocols. The approach described above will work flexibly for any sized spreadsheet and any complexity. It is easy to adapt the use of the manual testing approach to deal with logical statements, drop down selections, conditional formatting and data validation.

The philosophy here is to spend your time testing rather than spending your time writing test scripts.

The approach is extremely adaptable to a wide range of scenarios and is heavily dependent upon the competence and experience of the tester, therefore controls need to be in place to minimise the risks introduced by such flexibility. The following points are crucial to ensure the accuracy and integrity of your generated results and conclusions.

- Testers must be trained and familiar with both Excel, validation requirements, and the manual calculation methods.
- An SOP on the testing process is recommended which includes examples recommended to check different spreadsheet scenarios.
- Pre-specified data should be used where possible when performing calculations, we recommend attaching representative test data to the test script to allow the tester to select typical data.
- The tester should fully understand the spreadsheet and have the initiative to challenge and stress test it. This is something that normally does not occur when testers are instructed to follow a line-by-line test procedure. In the traditional approach testers follow the prewritten steps and see what the protocol tells them to see. We feel that the approach defined here frees up a lot of additional testing time, and that the tester can invest this time in actually challenging the system.
- Be aware of the multiple possibilities to further streamline the generation of results such as using a single screenshot to test multiple calculations, or to copy and paste data from one sheet to another.
- There are many situations where clever thinking or experience is required to deal with an individual spreadsheet or calculations. All situations can be dealt with as shown by the examples below:
  - Data rounding issues may arise depending on the test data used, be prepared to alter the number formats to verify calculations
  - To 'manually' verify complex calculations use a separate instance of Excel or a different statistical package; the formulae should not be copied, but redeveloped from first principals
  - Logical checks, conditional formatting, data validation etc. can often be verified with a string of screenshots which get annotated with their purpose once printed
- Throughout the testing the following important rules should be followed:
  - Justify and document any assumptions made during the testing; use risk based judgements whenever possible to make it easy to explain your logic to an auditor
  - Be flexible, yet consistent throughout the testing

Undertaking the 'manual testing' will be the most time consuming testing activity and the most technically demanding. The generic nature of the approach however ensures minimal document preparation time and **focuses the majority of the qualification activity on the task of testing**.

Data Generation/Formatting

Once the detailed spreadsheet functionality has been verified, a check is performed of the fully populated spreadsheet. This checks data formats and general spreadsheet layout and additionally provides a 'before installation' snapshot. This snapshot is used later to verify that the spreadsheet has not been modified during its move into its final operating environment.

Macro testing

This section is only applicable if macros are present in the spreadsheet. Macro functionality is defined as GAMP category 5 (custom programming), and as such requires extensive verification. In this case, generic test scripts are not appropriate and a traditional 'step-by-step' approach is necessary for the macro components.

Macro testing is performed following a 'black box' approach, where inputs are checked against outputs. For this type of testing it is usually expected to perform testing of all possible routes through the macro, and although this may sound daunting, most spreadsheet macros are small and straightforward to test.

URS Reference	Description	IQ/OQ/PQ Ref (or other)	Comments
3.7.2 3.7.4	Logical access to cell contents and spreadsheet calculations	A1, C3	Verifies that data input cells are accessible, and calculation cells are locked when applicable.
3.7.3	Logical access to workbook files	A1, B1, B2, C3	Access protection applied in B1, tested in A1, B2, C3.
3.9.1	Presence of spreadsheet specification	C1	Documentation listed.
3.9.2	Presence of developer and user training records	C1	Developer Training records within the specification, user training records verified in C1
3.10.1-x	Electronic signatures – functionality	A2	Electronic signatures tested in A2
3.10.1-x	Handwritten signatures – location and format	C3	Handwritten signatures form and locations verified in printouts in C3.
3.11.1-3	Spreadsheet Calculations - -refers to URS Table (Appendix A in specification [1])	A2	Summary statement, not verifiable. See Calculation Verification traceability in table below for calculations checked in A2

Figure 9: Traceability Matrix - High Level Generic User Requirements.

Appendix B – Spreadsheet Installation Testing

This appendix typically consists of two, relatively short, generic test scripts. The first records and verifies the installation of the spreadsheet template into the final operating environment. The second verifies the security of the spreadsheet and the application of any necessary audit trails and date/time stamping functionality.

Appendix C – Operational/Performance Testing

Completion of the qualification of the spreadsheet is achieved in this appendix which performs the following:

- Confirmation that critical documentation is present, this includes SOP's, Specifications, User Training records and Backup/Restore procedures.
- A 'double check' confirmation that the spreadsheet operates correctly in the final environment and that nothing has affected or altered since it was thoroughly tested in the functional testing stage.
  - The spreadsheet is not tested in great detail, but it is fully populated with data and compared back to the functional testing stage to confirm that it gives identical outputs to those generated in Appendix A.
- If macros are present additional tests similar to those performed in Appendix A would be executed to confirm that the macros have not been affected by the spreadsheets installation into the final operating environment.

Appendix D / E – Additional Data Sheets

These two Appendices are single pages that can be photocopied and appended to test scripts.

- Appendix D is a blank 'Additional Observation Sheet' that may be used for any pertinent information.
- Appendix E is a blank 'Manual Calculation Sheet" for use within Appendix A.

Appendix F – Traceability Matrix

In this streamlined approach the Requirements Traceability Matrix (RTM) is also integrated into the protocol for completion once all testing is complete. The RTM is divided into two sections, the first deals with tracing back to the high level User Requirements such as testing of security and approval methods. An example extract is shown in Figure 9. This part of the RTM benefits significantly from the use of consistent, generic specification and qualification documents. In practice we find that there is rarely any need to update this section of the RTM.

The second section of the RTM is finalised manually after completion of the Manual Testing. It traces back the spreadsheets calculations to the location and pages where they were tested. Using the very simple example shown earlier, this section of the RTM would be completed as shown in Figure 10.

Appendix G – Qualification Report

The final Appendix is a short Qualification Report, avoiding the need for a separate summary/final report and ensuring that all documentation is kept in one place. This further streamlines the process by avoiding the need for a separate set of approval signatures.

Performance and Compliance Monitoring

We advocate the use of regular performance and compliance monitoring reviews to ensure that the spreadsheets remain in a compliant state. Most regulated companies perform on-going 'reviews' of computer systems at one or two year intervals, these are a regulatory necessity and typically assess areas such as:

- Change control records
- System error records
- User access lists/training records
- Unauthorised access attempts
- User comments and enhancement requests

Figure 10: Traceability Matrix- Manual Calculations

Calculation Verification.
This section cross-references the specific spreadsheet calculations detailed in Appendix A of the Spreadsheet Specification. The IQ/OQ/PQ reference must be completed by the tester/reviewer to cross reference the test number within Appendix A (of this document) that verified the calculation (or function).

URS (App A) Reference	Cell Name / Description	IQ/OQ/PQ Reference (Manual Test No.)	URS (App A) Reference	Cell Name / Description	IQ/OQ/PQ Reference (Manual Test No.)
A.1.1	Average	1			
A.1.2	Overall Average	2			

Signature	A.N Other	Date	1/1/01	Comments	N/A A.N Other 1/1/01

Maintaining compliance of spreadsheets is however difficult due to Excels flexibility and the ability for users to modify content. If spreadsheets are regularly changing then the monitoring process becomes arduous and the risks to operational use are significantly increased.

The majority of spreadsheets we have implemented have built in protection and audit-trails[4] and this makes it easier to focus on the 'high-risk' areas within spreadsheets such as data/formula overwriting and modification of protection. The difficulty however is scheduling in these reviews, and although the data is available for review, it requires regular manual/procedural activities to audit and identify areas of concern. An alternative approach is to implement systems[5] that automate the reporting of the audited events (such as generating system alerts when passwords are removed).

In all cases such software additions greatly facilitate performance and compliance monitoring of spreadsheets and help to increase confidence in the use of spreadsheet in regulated industries.

Risk Based Validation

The current FDA thinking emphasises the value and benefit of risk based validation and we are often asked if we perform increased validation and testing on the more critical spreadsheets. In most cases we do not unless a spreadsheet is particularly complex. Undertaking a risk assessment (and report) is usually more time consuming than the additional effort of actually undertaking a full validation of the spreadsheet. In our testing process we check all calculations, and we find that because we don't write step-by-step test scripts we have the time to test every calculation thoroughly. Therefore every (GxP critical) spreadsheet gets a complete validation with detailed review of all calculations, and although this sounds daunting it is rare for the testing to take more than a few hours, and even very complex spreadsheets take less than 2 days.

For the future we do see benefits in the use of spreadsheet audits instead of repetitive calculation testing. Although we have not seen this approach used significantly, we have considered it and think that it is viable. We see the use of auditing in two separate areas:

- Spreadsheet accuracy auditing. Developer checks and peer reviews are commonly performed during spreadsheet development and specification, yet it is rarely formally documented. There are a number of tools available [6, 7] to aid the auditing process and to assist in the confirmation of a spreadsheet's accuracy. We believe there is scope for the increased use of such audits to replace some calculation testing. We do, however, believe that there is still a necessity to independently demonstrate that critical calculations or functionality are accurate.
- Spreadsheet performance auditing. The use of automated tools has been discussed in a previous section (Performance and Compliance Monitoring).

Conclusion

Testing of Excel spreadsheets can be time consuming, and it can be difficult to determine the depth at which to test. The use of traditional step-by-step test scripts makes the protocol writing stage arduous, which leads to a rigid and often unchallenging testing process. The approach recommended in this paper uses a generic 'manual testing' form which allows the tester to thoroughly check the spreadsheet and stress test with suitable data. The time saved in not writing lengthy test scripts is invested into undertaking a complete and very thorough qualification which ensures the spreadsheet is fit for purpose and will satisfy regulatory audit. This testing approach provides a flexible solution for testing data intensive systems, and can be successfully adapted to databases, LIMS and ERP systems.

Maintaining compliance is vitally important, and is an area that is currently under performed in the industry. Procedural processes combined with software tools are recommended to ensure that the validation status is maintained. ■

More information on spreadsheet validation can be found at http://www.spreadsheetvalidation.com

David Howard
BSc CChem MRSC.
Validation Consultant.
ABB Engineering Services
PO Box 99
Billingham, TS23 4YS
United Kingdom
+44 (0)1937 589813 (Office)
+44 (0)7740 051595 (Mobile)
david.howard@gb.abb.com
www.abb.com/lifesciences

Dave Howard is a Validation Consultant at ABB Engineering Services, specialising in End User Computing applications.

David Harrison BSc MBA
Principal Consultant.
ABB Engineering Services
PO Box 99
Billingham, TS23 4YS
United Kingdom
+44 (0)1207 544106 (Office)
+44 (0)7957 635046 (Mobile)
david.harrison@gb.abb.com
www.abb.com/lifesciences

Dave Harrison is a Principal Consultant at ABB Engineering Services where he is the Product Manager for spreadsheet validation solutions.

References:

[1] David A Howard & David Harrison, ABB Engineering Services "A Pragmatic Approach to the Validation of Excel Spreadsheets – Overview" Pharma IT Journal, Vol. 1 No. 2 April 2007.
[2] David Harrison & David A Howard, ABB Engineering Services "A Pragmatic Approach to the Specification of Excel Spreadsheets" Pharma IT Journal, Vol. 1 No. 3 July 2007.
[3] Good Automated Manufacturing Practice Guide, Version 4, ISPE, Tampa FL, 2001
[4] DaCS™, Data Compliance System, Compassoft Inc. http://www.spreadsheetvalidation.com/solutions/dacsproduct.htm
[5] Enterprise Spreadsheet Management Software, ClusterSeven Inc. http://www.clusterseven.com.
[6] Exchecker™, Compassoft Inc, http://www.compassoft.com/exchecker.htm.
[7] PUP, Power Utility Pack, J-Walk & Associates. http://j-walk.com/ss/pup
[8] FDA 21 CFR 211, Current Good Manufacturing Practice Regulations for Finished Pharmaceutical Products.