MS Excel Spreadsheet Validation

Overview

This article will present a simple method for validating MS Excel spreadsheets for GxP use. The goal of our validation strategy is to improve testing for individual spreadsheets in less time. This is accomplished by creating a common methodology for gathering and documenting spreadsheet requirements, then demonstrating that the spreadsheet meets those defined requirements. Once the methodology is established, it can rapidly be applied to many spreadsheets.

Scope

This article covers common examples of spreadsheet validation activities. Some special or complicated examples are addressed, but it should be understood that special cases not explicitly discussed can be added as needed to fit your existing validation requirements. A good example is any spreadsheet that uses or is used with custom automation. We have validated spreadsheets that contained hundreds of formulas and custom macros, but did not require any user intervention at all because they were created, populated with data and saved entirely by external code. In this case, the focus of the validation was on validating formulas and macros, but security testing was limited to proving that users could not intercept or interrupt the operation at any time.

Assumptions

We do not make assumptions about your existing SDLC (Software Development Life Cycle); all material presented here can be adapted to meet your existing standards and practices.

We do assume the spreadsheet requiring validation has already been created, as this is our most common scenario.

Approach

The basics of our validation approach are applicable to almost all spreadsheets:

- Define basic documentation practices and methodology in a single document, the Spreadsheet Validation Master Plan. This document can be referred to for all individual spreadsheet validation projects, and does not need to be reviewed or approved for each validation effort.
- For each workbook, define the requirements for each worksheet or chart in the workbook, then verify that these requirements have been properly implemented.
- Emphasis is on defining and testing formulas, and also on the security for each sheet by limiting the parts of each sheet that the users are allowed to edit.

Methodology

To keep things simple, we define how we enter information into the requirements and design specification documents, then describe how we go about testing the spreadsheet in the test protocols. Input cells may be defined as one single cell or as a range of cells.

Requirements Specification

This document includes all the requirements that your spreadsheet must accomplish. This document should be kept simple and relatively non-technical so that anyone who reads the document will understand the requirements.

Start with a common validation template:

- Identify and list all worksheets and charts in the workbook.
- Describe any enforced workflows.
- Document how the spreadsheet meets the requirements of 21 CFR 11, especially audit trails and limited system access.

For each worksheet, define:

- Which users may enter data into the sheet.
- Which cells accept data and the format of the entered data.
- Which formulas or calculations exist on the sheet.
- Which cells represent the output or final calculation.

For each chart, define:

- Which data sets are used to create the chart.
- Critical properties including the title, axis labels and units.

Design Specification

After defining requirements, create the design specification. The design specification describes how requirements outlined in the requirement specification have been implemented. This document should include enough information so that a developer can create the entire software project based on the information contained within this document and from reading the requirements specification.

Separate each worksheet into four sections: Inputs, Processing, Outputs and Security.

Inputs

Document cells where users enter or update data. In an automated system, you can also define the source, the input data or instructions. If any validation rules are used to enforce proper data entry, these should be documented here as well.

Processing

Document formulas that are used on the worksheet. Any custom macros or VBA code are also documented here.

The majority of errors that we find when validating spreadsheets are in the formulas. The best technique to identify errors is to define each formula in the design specification using the actual names of the variables represented as input cells.

Ofni Systems

For example, it may be easy to document that the range of cells F10:F20 contains the formula =(A10*C5)/(D5*B10). However, it is difficult to verify if this formula is correct. Instead, write the formula out like this:

Cell C5:	Volume (V)
Cell D5:	Ideal Gas Constant (R)
Cells A10:A20	Pressure (P)
Cells B10:B20	Temperature (T)
Cells F10:F20	Final Result (moles of gas, n)
F10=(A10*C5)/(D5*B10)	n = (PV)/(RT)

Writing out formulas in this manner is the easiest and most effective way to detect errors in formulas.

In addition, macros and VBA code used in the spreadsheet must be documented. Copy the code into the design specification and annotate as needed to describe the purpose of each function.

Note: If you have good coding standards and user proper headers and comments in your code, this step may be already be done for you.

Outputs

Outputs usually fall into three main categories:

- The cell or range of cells that contain the final result of all previous calculations
- Charts – many times these are printed and saved with external reports
- Data that is copied into a final result sheet or exported to a separate file or database.

Security

This section can be a short statement, e.g. “All non-input cells must be locked to prevent changes.” You can also include additional security settings if you are using custom code or a third party add-on to implement multiple levels of security to control who can edit certain cells, run a macro or function, etc.

Test Protocols

The Test Protocol demonstrates that spreadsheet requirements were properly implemented according to the design specifications and function as expected.

Installation Qualification (IQ) testing is usually limited to verifying and documenting the file location and the version of the spreadsheet , MS Excel and any add-ons (third party apps, etc.).

Operational Qualification (OQ) testing verifies requirements, primarily formulas, macros, and testing the security of each sheet to verify that all non-input cells are locked to prevent changes.

Performance Qualification (PQ) testing verifies functionality under live conditions. Excluding certain situations (for example, where the spreadsheet is used to identify an unknown sample), the PQ can be omitted if allowed and justified in the Spreadsheet Validation Master Plan.

The IQ, OQ and PQ may be combined as needed.

To start generating test cases, break testing down into Inputs, Processing, and Outputs. Security testing is easier to test in separate test cases.

Input Testing

Examples of the types of test cases to write for input testing:

- What data can be entered into each input cell?
- Are validation rules being enforced?

Process & Output Testing

These are usually easier to combine when writing test cases. FOCUS ON TESTING THE FORMULAS! There are several methods for verifying and testing formulas.

- Visual inspection of each formula or range of formulas
- Verification of each numerical calculation using a calculator.
- Testing for consistent formulas in a certain range.
- Put the same value in all input cells to verify that all formulas report the same output value.
- Alter an input cell to verify that the calculated value and the final result changes as expected.
- For critical applications, continue altering each subsequent input cell to verify that all formulas report the same value. For less critical sheets, you may try testing the first, last and a random third input cell in the middle to spot-test formulas.
- Continue testing until all input cells have had data entered or changed, and the results verified. The key point is to look for formulas that are incorrect or charts that are not using the correct or complete set of data, or are pointing to the wrong columns of data.
- Test Macros by entering a range of data and comparing the results with a hand calculator. Visual inspection is often adequate to verify that the function performed as expected.
- Charts can be tested by a combination of visual inspection and verification of the properties, including the dataset used as the basis for the charts.

Security Testing

The type and amount of security testing needed spreadsheets is largely based on how security has been implemented. At a bare minimum, you should test that users are limited to entering data into the defined input cells only, and that users do not have the ability to alter any other part of the spreadsheet. Failure to do this compromises the integrity of the validation effort and the resulting data or information generated by the workbook.

Conclusion

The methodology presented here will result in a User/Functional Requirements Specification, a Design Specification, and a Test Protocol ready for approval and execution. Deviations identified during testing are resolved according to existing validation practices, and a summary report which shows that all the activities specified in the Validation Master Plan or SOP can be generated.

About the Author

Tyson Mew is President of Ofni Systems, a regulatory compliance consulting, software and validation firm for FDA-regulated companies. Ofni Systems is the creator of ExcelSafe, which provides MS Excel spreadsheets with all technological tools for compliance with 21 CFR 11, including audit trails, electronic signatures, passwords and user-level security. Ofni Systems is also the creator of the FastVal Validation Document Generator, which automates the generation and execution of validation documents, including for spreadsheet validation. For more information, visit www.OfniSystems.com.

Ofni Systems