Requirements Completeness Excel Macro Tutorial

For Version 2.0, Release 20090504

Scope

This document describes the Requirements Completeness macro for Excel.

The requirements completeness macro takes a set of functional requirements and determines if all of the system outputs can be derived from the system inputs through an unbroken chain of requirements. The requirements start out as English language statements. The user generates a summary of the data used in each requirement. The macro checks the summary for completeness.

Tutorial

Columns

As shown in the Requirements_Completeness_Example_1.xls spreadsheet the Requirements Completeness spreadsheet has seven labeled columns,

- Reference Version
- ID
- Description
- Update Status
- Type
- Defined Name
- Arguments

Reference Version Column

When making edits to an existing requirement, it is sometimes helpful to have the original wording available as a reference. The Reference Version column is intended to hold the original requirement. It is ignored by the Requirements Completeness macro.

ID Column

The ID column can be used for some kind of requirement number or identifier. It is ignored by the Requirements Completeness macro when checking completeness. It is used as part of the requirement label in the Requirements Analysis diagram, and it is used to create Requirements Analysis diagrams that only contain a subset of the requirements.

Description Column

The Description column can be used for the text of the requirement. It is ignored by the Requirements Completeness macro when checking completeness. It is used as the label text for the requirement in the Requirements Analysis diagram.

Update Status Column

The Update Status column is used to record the status of each requirement. The column header contains a button. When pressed, the Requirements Completeness macro runs and updates the entries in the rest of the column. The entry will be empty with a green background if the requirement does not violate any of the completeness rules. If the requirement does violate any of the completeness rules, the entry contains one or more error messages with a red background.

Type Column

The Type column contains one of fourteen labels to indicate the type of the requirement.

- Input
- Output
- Process
- Debug
- Const
- AKA
- Struct
- _ (underscore)
- Comment
- Feature
- Define
- Group
- FirstRow
- LastRow

The *Input* label indicates that the name in the Defined Name column is an input data item to the system.

The *Output* label indicates that the name in the Defined Name column is an output data item from the system.

The *Process* label indicates that the name in the Defined Name column is an intermediate variable produced as the result of the calculations specified by a functional requirement. The intermediate variable will be used as input data to other process requirements which will in turn produce further results and ultimately an output from the system.

The *Debug* label indicates that the name in the Defined Name column is an intermediate variable produced as the result of the calculations specified by a functional requirement. The intermediate variable will not be used to produce any further results.

The Const label indicates that the name in the Defined Name column is a constant. Something that is hard coded or hard wired within the system.

The AKA label indicates that the name in the Defined Name column has one or more synonyms. The synonyms are listed in the Arguments columns. Without a tool to enforce consistency, requirements will have multiple words or phrases for the same data item. The AKA type ties all the different names together so that the completeness test does not fail simply because the names don't match.

The Struct label indicates that the name in the Defined Name column is a record, or data structure. The fields or elements are listed in the arguments column. Requirements can refer to any of the elements by using one of the argument names, or to the structure as a whole by using the defined name.

The _ (underscore), Comment and Feature labels in the Defined Name column cause the completeness macro to ignore that row completely. These rows can be used to insert labels or commentary into the spreadsheet.

The Define label in the Defined Name column is used to attach an attribute to a defined name. Currently the only available attribute is Global. This is used to reduce the number of crossing lines in the Requirments Analysis diagram.

The Group label in the Defined Name column is used to define a subset of requirements that will appear by themselves in a separate Analysis Diagram.

The FirstRow and LastRow labels control the rows that are processed by the Requirements Completeness macro. Without these labels, the macro starts at row 2 and processes rows until the first row with an empty type column is encountered. The macro will also stop on a row with LastRow in the type column.

Defined Name Column

The Defined Name column contains the name for a data item. The data item can be an input to the system, an intermediate value produced within the system, or an output from the system. If the type column contains the Group label, the Define Name column contains the name for the group. The group name is also used as part of the file name for the Requirements Analysis diagram that is produced for that group.

Arguments Column

Process, Debug, Struct, Define and Group requirements have one or more data items as arguments. These are listed starting in the Arguments column.

Putting Functional Requirements Into Spreadsheet Form

The requirements in the spreadsheet can appear in any order. However, it can be useful to list the *Output* requirements first followed by the *Input* requirements and then the *Process* and *Debug* requirements. Putting outputs at the top makes it easy to get an overview of what the system does. Having the Input and Output requirements together at the top of the spreadsheet provides a summary of the system, and can be easily checked against any interface definition documents.

Rather than reading through the requirements to extract the system inputs and outputs, the requirements can be put into an empty spreadsheet and the Requirements Completeness macro can be used to help identify the system inputs and outputs. The system inputs and outputs will then be added at the top of the spreadsheet.

To create a Requirements Completeness spreadsheet from a set of requirements:

Adding the Initial Requirements to the Spreadsheet

- Make a copy of one of the example spreadsheets.
- Delete all of the rows except the first.
- If the requirements have some kind of an identifier, copy and paste the identifier into column B (ID), one row per requirement.
- Copy and paste the requirement text into column C (Description), one row per requirement.
- Leave column D (Update Status) blank. The macro will fill it in.
- Put "Process" (without the quotes) in column E (Type)
- For each requirement
    - Identify all of the data items (nouns and noun phrases)
    - Select the data item that is produced by the requirement and put that name of that in column F (Defined Name).
        - If the requirement does not produce a named data item, revise the requirement (at least the version in the spreadsheet) so that it does. Note: "TBD" (without the quotes) is a predefined name that can be used until a better name is found. It will result in a warning on a yellow background.
        - If the requirement produces more than one data item, duplicate the requirement for each data item produced.
    - List the remaining data items, one per column, starting in column G (Arguments). These data items may be
        - Part of a condition that must be true for the requirement to be applicable
        - Part of an event that triggers the action defined by the requirement
        - The data items that are used to produce the result of the requirement

Adding Input and Output Requirements to the Spreadsheet

- Click on the button at the top of column D (Update Status) to run the Requirements Completeness macro
- Scan down column D (Update Status) looking for error messages that indicate that an argument is undefined. If the undefined data item is an input to the system:
    - Insert a new row at the top of the spreadsheet
    - Put "Input" (without the quotes) in column E (Type).
    - Put the data item name in column F (Defined Name).
- Scan down column D (Update Status) looking for error messages that indicate that a process name is not used. If the unused name is an output of the system:
    - Insert a new row at the top of the spreadsheet.

    - Put "Output" (without the quotes) in column E (Type).
    - Put the process name in column F (Defined Name).
- Click on the button at the top of column D (Update Status) again. At this point any remaining errors represent places where the chain of requirements is broken, and the requirements need to be revised.

Checkbook Balancing System Example

To make this more concrete, consider a "system" that balances a checkbook from a monthly statement. The Checkbook Balancing System (CBS) might have the following functional requirements (with the noun phrases highlighted):

REQ1 If there are any **service charges** or other **deductions** that are on the **monthly statement** but not recorded in the **checkbook**, the CBS shall subtract them from the **checkbook account balance**.

REQ2 If there are any **credits** that are on the **monthly statement** but not recorded in the **checkbook**, the CBS shall add them to the **checkbook account balance** to create the **new checkbook account balance**.

REQ3 If there are any **deposits** listed in the **checkbook** that are not on the **monthly statement**, the CBS shall add them to the **account balance** shown on the **monthly statement**.

REQ4 If there are any **withdrawals** listed in the **checkbook** that are not on the **monthly statement**, the CBS shall subtract them from the **account balance** shown on the **monthly statement** to create the **new statement balance**.

Adding the Initial Requirements to the Spreadsheet

These requirements were converted into the Requirments_Completness_Example_1.xls spreadsheet.

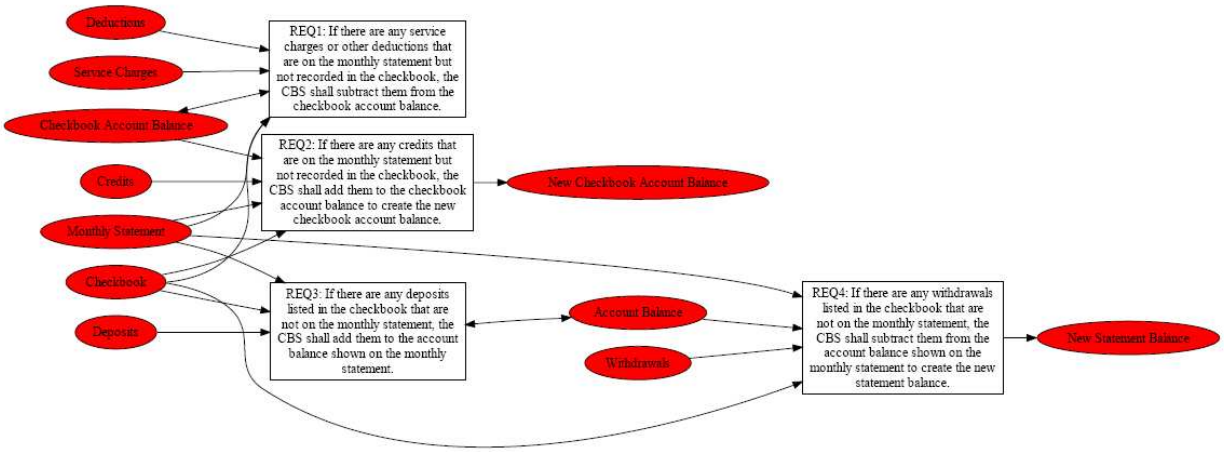
This was done by taking a copy of an earlier spreadsheet and deleting all of the rows except row 1. Then each of the requirements was copied into column C (Description) and "Process" was entered into column E (Type).

The output for each requirement was identified. For REQ2 and REQ4 the output was indicated by the verb phrase "to create". So "New Checkbook Account Balance" became the Defined Name for REQ2 and "New Balance Statement" became the Defined Name for REQ4. Once the Defined Name was selected, the remaining nouns in the requirement were listed as Arguments.

REQ1 and REQ3 present a problem. In both cases the output of the requirement is not named. This is because one of the inputs is being updated. For these requirements, at this point, the updated input was used as the Defined Name. So the Defined Name for REQ1 became "Checkbook Account Balance" and the Defined Name for REQ3 became "Account Balance". Then all of the nouns in the requirement were listed as Arguments, including the noun that became the Defined Name. (i.e. the Defined Name was repeated as an Argument.)

Once all the requirements had been added to the spreadsheet, the Update Status button was pressed and all the requirements turned red, as expected.

Requirements Analysis Diagram for Example 1



Requirements are shown on the diagram as boxes which contain the ID and Description. Internal data items (i.e. data other than system inputs and system outputs) are shown as ovals. The arrows follow the processing flow in the Description.

At this point the system has no inputs or outputs. Consequently, all of the requirements are incomplete. This is shown on the Requirements Analysis diagram by the red shading on all of the data items.

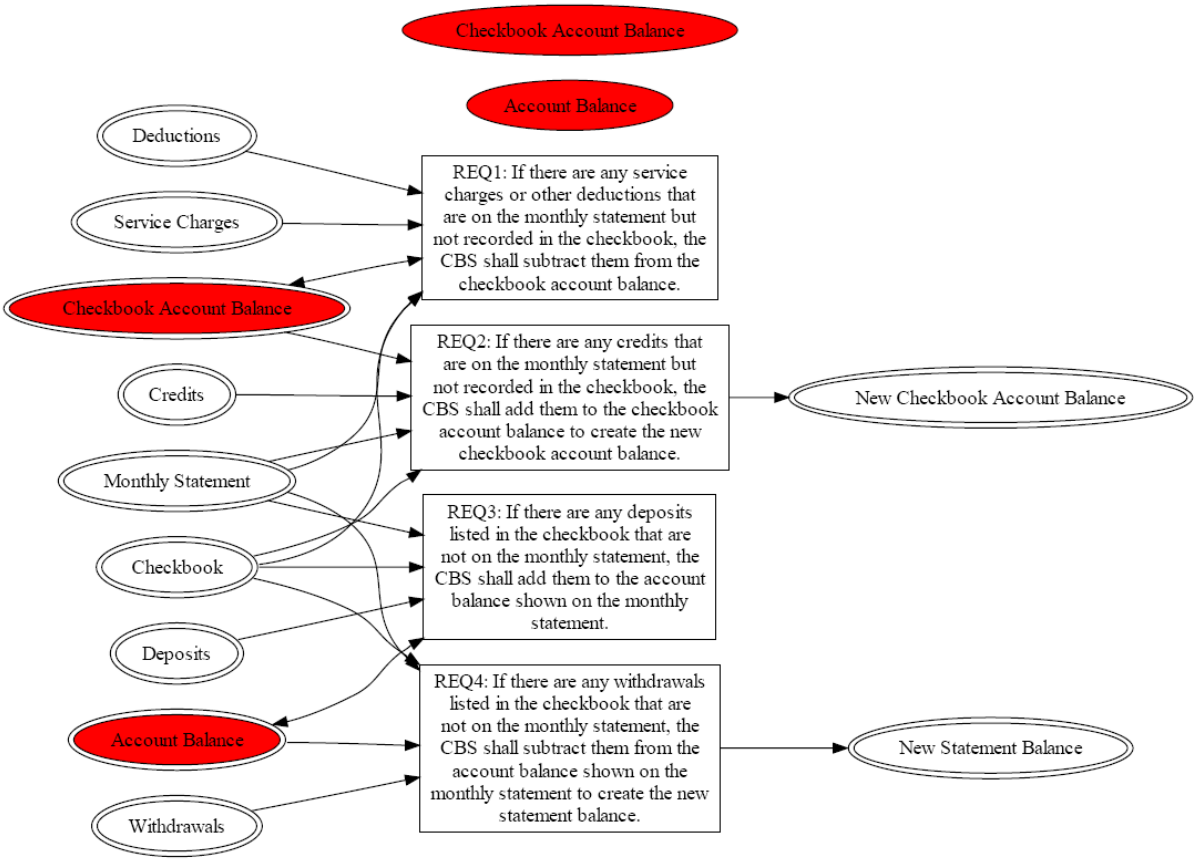
Adding Input and Output Requirements to the Spreadsheet

Scanning through the error messages in Requirements_Completeness_Example_1.xls looking for unused names showed that “New Checkbook Account Balance” and “New Statement Balance” were neither used as arguments to some other Process requirement nor used as the Defined Name of an Output requirement. This is expected because these are both outputs of the system. Consequently both of these were added to the top of the spreadsheet as Output requirements. See the Requirments_Completeness_Example_2.xls spreadsheet.

Scanning through the error messages again looking for undefined arguments showed that there were 7 arguments that never appeared as a Defined Name, and 2 arguments that appeared as Defined Names, but only in requirements where they were also an argument. All 9 of these data items were inputs to the system and so were added to the spreadsheet as Input requirements. Note that the Output requirements are listed first, followed by the Input requirements and then the Process requirements. This is the recommended order.

Once the Input and Output requirements had been added to the spreadsheet, the Update Status button was pressed and all but four of the requirements turned green as shown in the Requriements_Completeness_Example_2.xls spreadsheet.

Requirements Analysis Diagram for Example 2



The Inputs and Outputs are shown on the diagram as ovals with a double border. Inputs are at the extreme left of the diagram, and Outputs are at the extreme right.

Adding New Names for Intermediate Values

Half of the remaining issues are there because REQ1 is updating the Checkbook Account Balance. This makes Checkbook Account Balance both a system input and an intermediate value created within the system. Since inputs cannot be changed, this is an error. The fix is to introduce a new name for the intermediate value and rewrite the requirement (at least in the spreadsheet).

The other half of the remaining issues are there because REQ3 is updating the Account Balance. The solution is the same as for REQ1.

Using the same name for both an input to and the output of a requirement is a common error, the failure to give the output value a unique name. If the requirements are revised to include unique output names they might look like:

REQ1 If there are any service charges or other deductions that are on the monthly statement but not recorded in the checkbook, the CBS shall subtract them from the checkbook account balance **to create the trial checkbook account balance**.

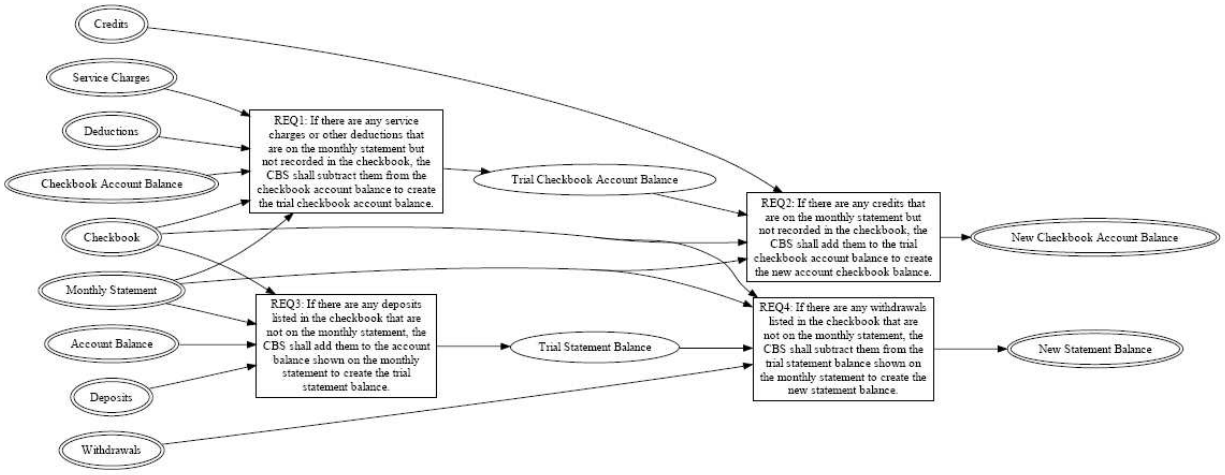
REQ2 If there are any credits that are on the monthly statement but not recorded in the checkbook, the CBS shall add them to the **trial checkbook account balance** to create the new checkbook balance.

REQ3 If there are any deposits listed in the checkbook that are not on the monthly statement, the CBS shall add them to the account balance shown on the monthly statement **to create the trial statement balance**.

REQ4 If there are any withdrawals listed in the checkbook that are not on the monthly statement, the CBS shall subtract them from the **trial statement balance** to create the new statement balance.

The result is shown in the Requirements_Completeness_Example_3.xls spreadsheet. The status column is all green showing that this is a complete set of requirements in the sense that all of the outputs are derived from the inputs through an unbroken chain of requirements.

Requirements Analysis Diagram for Example 3

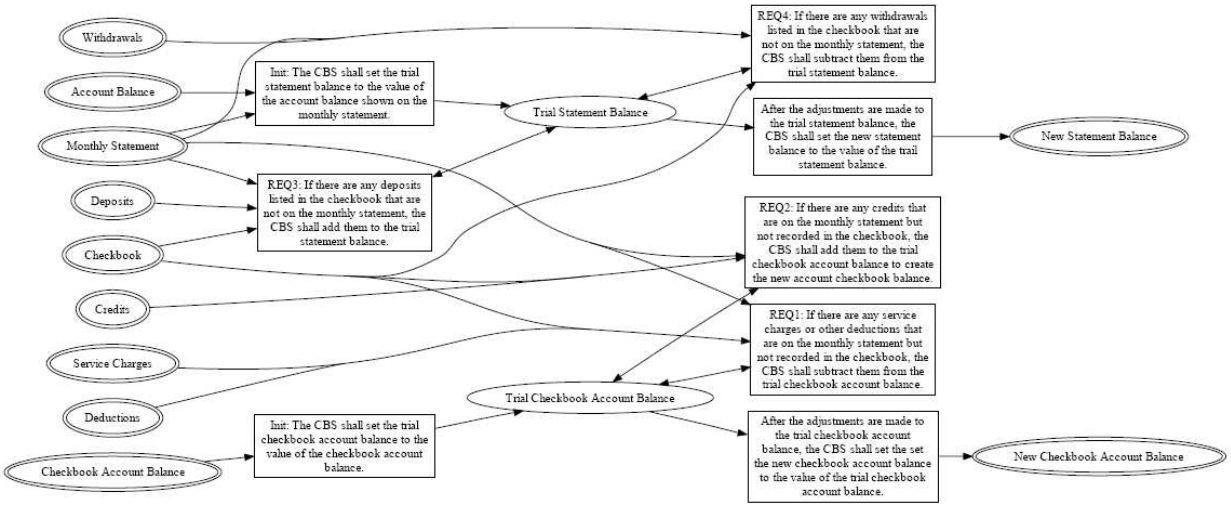


However, there are two problems in the third example that are not detected by this analysis; an ordering dependency and an initialization error. A dependency has been introduced between REQ1 and REQ2. Any Service Charges and Deductions must be subtracted before any Credits are added. The order of operations has become a requirement of the system. Worse, if there are no Service Charges or Deductions, the Checkbook Account Balance is uninitialized. The same issues occur in REQ3 and REQ4.

One way to deal with this is to have an additional requirement that simply creates the Trial Checkbook Account Balance from the Checkbook Account Balance. Then both REQ1 and REQ2 could refer to the Trial Checkbook Account Balance without introducing any dependency between them. After being updated, the Trial Checkbook Account Balance would be transferred to the New Checkbook Account

Balance. The Trial Statement Balance would be handled similarly for REQ3 and REQ4. The results of these changes are shown in the Requirments_Completeness_Example_4.xls spreadsheet.

Requirements Analysis Diagram for Example 4

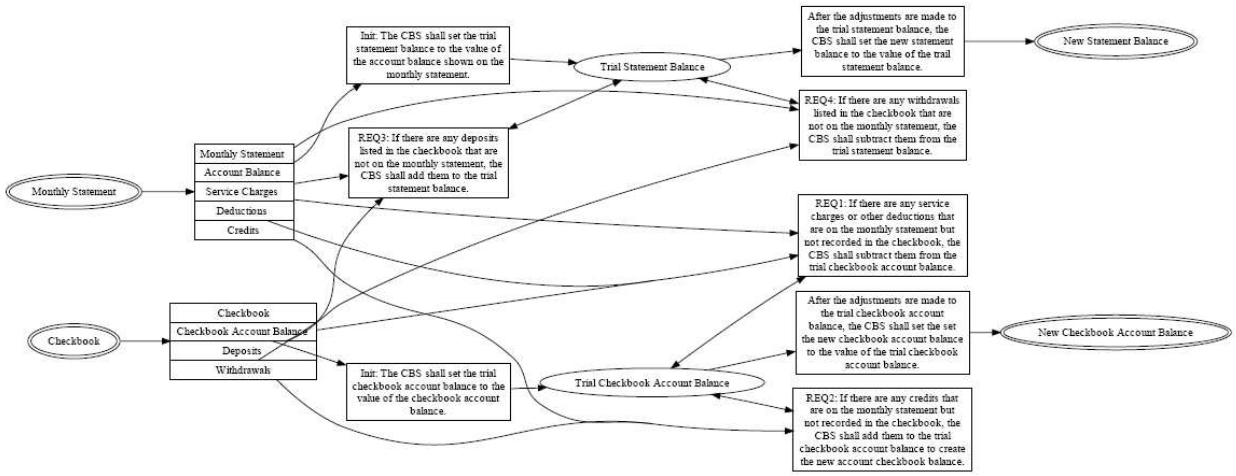


Advanced Techniques

Data Structures

The Monthly Statement is made up of an Account Balance, Service Charges, Deductions and Credits. Similarly, the Checkbook is made up of a Checkbook Account Balance, Deposits and Withdrawls. The Requirements_Completeness_Example_5.xls spreadsheet shows how to use structures to handle data like Monthly Statement and Checkbook. There is very little benefit from using structures if they only appear in the spreadsheet. The real benefit shows up when they are viewed on a Requirements Analysis diagram.

Requirements Analysis Diagram for Example 5



Requirements Analysis Diagrams

Examine the .pdf files associated with examples 4 and 5. By grouping the individual data items into structures, the diagram is simplified, and as a consequence, the system is easier to understand.

Whenever the Completeness Macro is run, it not only updates the Status column of the spreadsheet, it also creates a text file which contains the ASCII version of the Requirements Analysis diagram. Running the doit.bat file will use the Graphviz application to produce a .pdf file with the graphical form of the Requirements Analysis diagram.

Graphviz is an open source graph (network) visualization project from AT&T Research. It can be obtained from http://www.graphviz.org/

Graphviz also comes as part of Doxygen, which is used to document C++ code.

Once Graphiviz is installed, the doit.bat file shows how to use it to create the Requirements Analysis diagrams.

Reference

Overview

The requirement types are “Input”, “Process”, “Debug” and “Output” (without the quotes).

The macro does string comparisons, so spelling and capitalization matter.

Input requirements are treated as system inputs. They are as they are, and cannot be changed. They can be used as an input argument in any other requirement.

Process requirements take arguments and produce an intermediate data item with the Defined Name. The Defined Name of a Process requirement can be used as either an argument to other Process requirements or as the Defined Name of an Output requirement, but not both.

Debug requirements take arguments and produce a data item with the Defined Name. However, the Defined Name of a Debug requirement cannot be used as an argument.

Output requirements are treated as system outputs. They must have the same Defined Name as one of the Process requirements. The Defined Name of an Output requirement cannot be used as an argument.

The spreadsheet is processed starting at row 2 and continuing until the entry in column 4 (Type) is blank.

Input Requirements

- The defined name cannot be redefined as any other requirement type.
- The defined name cannot be multiply defined as an input type.
- The defined name does not have to be used as an input argument, but it will have a green status with an Info: message if it is not.
- An Input requirement does not have input arguments.

Process Requirements

- The defined name cannot be redefined as an Input or Debug type.
- The defined name can be redefined as an Output requirement.
- The defined name can be multiply defined as a Process type.
- The defined name must be
    - Either used as an input argument to a Process or Debug requirement.
    - Or redefine an Output requirement and not be used as an input argument
- The defined name can be "TBD" (without the quotes), and will result in a warning on a yellow background.
- A Process requirement must have one or more input arguments.
- Each of the input arguments must be the defined name of an Input, or Process requirement.

Debug Requirements

- The defined name cannot be redefined as any other requirement type.
- The defined name can be multiply defined as a Debug type.
- The defined name cannot be used as an input argument.
- The defined name can be "TBD" (without the quotes), and will result in a warning on a yellow background.
- A Debug requirement must have one or more input arguments.
- Each of the input arguments must be the defined name of an Input or Process requirement.

Output Requirements

- The defined name cannot be redefined as an Input or Debug type.
- The defined name must also be the defined name of a Process Type.
- The defined name can be multiply defined as an Output type.
- The defined name cannot be used as an input argument.
- An Output requirement does not have input arguments.

Unit Test

The Requirements_Completeness_Unit_Test.xls spreadsheet shows an example of all the uses and misuses of the Requirements Completeness macro. The first column is used to name the test case. The second column is a short description of the test, with a background color that is supposed to match the background color of the Status column. Mismatched colors indicate a bug somewhere. The Type, Defined Name and Arguments columns are used in the normal way. The naming convention for the defined names is to use an abbreviation of the test case name and a digit.

Revision History

12/30/2008	1st Edition
12/31/2008	Updates for readability. Added page numbers.
01/23/2009	Revised for clarity. Expanded tutorial. Added a fourth example spreadsheet.
05/25/2009	Revised for Version 2.0, Release 20090504