
UTILITY CALCULATION SOFTWARE MANAGEMENT	Manual Document Page Issue Date	TFC-ENG-DESIGN-C-32, REV H 1 of 32 July 21, 2015
--	--	---

TABLE OF CONTENTS

1.0	PURPOSE AND SCOPE	2
2.0	IMPLEMENTATION	2
3.0	RESPONSIBILITIES.....	3
3.1	Manager	3
3.2	Software Owner (Owner).....	3
3.3	Project Lead	3
3.4	Developer.....	3
3.5	Independent Technical Reviewer.....	3
3.6	Software Technical Support Analyst (STSA).....	4
3.7	Tester	4
3.8	Quality Assurance.....	4
3.9	User.....	4
4.0	PROCEDURE.....	4
4.1	Software Life Cycle for New Utility Calculation Software Registrations.....	5
4.2	Software Development Cycle	5
4.2.1	Planning Phase.....	6
4.2.2	Requirements Phase.....	6
4.2.2.1	Software Requirements Verification.....	7
4.2.3	Design and Implementation Phase.....	7
4.2.3.1	Software Design Verification	8
4.2.4	Test Phase	9
4.2.4.1	Stage 1 Review	9
4.2.5	Acceptance Testing.....	10
4.2.5.1	Stage 2 Review and Approval for Use.....	11
4.2.6	Installation and Checkout Phase	12
4.3	Maintenance and Operations.....	13
4.3.1	Problem Reporting and Corrective Actions.....	14
4.3.2	Software Change Request (SCR).....	14
4.4	Revisions to Existing Utility Calculation Software Registrations	15
4.5	Retirement.....	16
5.0	DEFINITIONS.....	16
6.0	RECORDS	17
7.0	SOURCES.....	17
7.1	Requirements	17
7.2	References.....	17

TABLE OF ATTACHMENTS

ATTACHMENT A – SOFTWARE QUALITY ASSURANCE TABLES	19
ATTACHMENT B – UTILITY CALCULATION SOFTWARE STANDARDS AND PRACTICES.....	23

1.0 PURPOSE AND SCOPE

(7.1.2)

This procedure establishes the software engineering method for managing the software life cycle (SLC) activities in TFC-PLN-02 and TFC-BSM-IRM-STD-01 for utility calculation software (UCS), namely, multiple-use spreadsheets (see Section 5.0 Definitions) initiated and registered in accordance with TFC-BSM-IRM_HS-C-01. This procedure applies to Washington River Protection Solutions, LLC (WRPS) employees, suppliers, and subcontractors managing and performing the SLC software work activities (SWA) related to multiple-use spreadsheets that perform calculations or data manipulation in support of technical products (e.g., documents and reports) and activities.

The following spreadsheet products are not subject to the requirements of this procedure:

- Grade level E or N/A UCS controlled in accordance with TFC-BSM-IRM_HS-C-01.
- Single-use spreadsheets (see Section 5.0 Definitions) controlled and independently checked/verified prior to use in accordance with TFC-ENG-DESIGN-C-10.
- Business, financial, and administrative activity-type (e.g., table generation) spreadsheets.
- Computation spreadsheets (see Section 5.0 Definitions) controlled in accordance with TFC-ENG-DESIGN-C-10.
- Spreadsheets generated as output (e.g., query results, reports, etc.) from other software applications approved for use in accordance with TFC-BSM-IRM_HS-C-01.
- Spreadsheets that are used to check/verify the results of another software application or calculation, provided that the spreadsheet is independently developed using an alternate method (i.e., different structure/formula).
- Add-ins for use with safety grade UCS that satisfy the definition for Otherwise Acquired Software are dedicated/controlled in accordance with TFC-ENG-DESIGN-C-65.
- Acquisition of subcontracted design and analysis software services is covered under TFC-BSM-IRM_HS-C-03.

The terms “software,” “spreadsheet,” “utility calculation software (or UCS),” and “multiple-use spreadsheet” are used synonymously throughout this procedure, unless otherwise stated (e.g., single-use spreadsheet).

2.0 IMPLEMENTATION

Multiple-use and single-use spreadsheets in the final review and approval process pending release prior to or on the issue date of this procedure revision may be processed in accordance with the prior revision of this procedure. Single-use spreadsheets released prior to the issue date of this procedure are not required to meet the documentation and verification requirements of this revision. Revisions to an issued single-use spreadsheet shall be processed in accordance with TFC-ENG-DESIGN-C-10. Multiple-use spreadsheets approved for use under previous revisions of this procedure can remain in use under their existing documentation and shall be brought into compliance with this procedure at the time of their next revision.

Revision H of this procedure introduces electronic review and approval of the Software Management Plan for Utility Calculation Software (SMP) via SmartPlant^{®1} Foundation (SPF) workflow. Individuals assigned roles within Section 3.0 should complete SPF Part 1 and Part 2 training and be well versed in the use of TFC-ENG-DESIGN-C-25 prior to using this procedure.

3.0 RESPONSIBILITIES

Responsibilities specific to this procedure are specified in this section; additional responsibilities reside in Section 4.0. Personnel can fulfill multiple role assignments unless otherwise noted. Outside subcontractor personnel tasked to develop software under this procedure assume all of the responsibilities in this section.

3.1 Manager

- Assigns personnel to perform SWAs.
- Ensures personnel qualifications are adequate to perform assigned SWAs.
- Approves SLC deliverables and UCS for acceptability and use.
- Approves Software Change Request (SCR).
- Responsible for the accuracy and completeness of SWA deliverables.

3.2 Software Owner (Owner)

- Ensures SWAs are completed and deliverables are compliant with this procedure.
- Identifies desired calculations/functionality to be contained in the UCS.
- Ensures software problems/errors requiring a Problem Evaluation Request (PER) are initiated and resolved in accordance with TFC-ESHQ-Q_C-C-01.

3.3 Project Lead

- Manages the performance and documentation of SWAs to the graded approach.
- Maintains configuration management and control of SWA deliverables and Configuration Items (CIs) in SPF.

3.4 Developer

NOTE: The Developer cannot be the ITR or the Tester.

- Designs UCS to perform calculations/functionality to approved software requirements.
- Develops UCS using a site-approved (i.e., “Operational” in HISI) commercial-off-the-shelf spreadsheet application platform (e.g., Microsoft Excel[®], Mathcad, Igor, etc.).

3.5 Independent Technical Reviewer

NOTE 1: In order to avoid “self-review” verification by the Developer, the Independent Technical Reviewer (ITR) cannot be the Developer for the specific software revision; this does not prevent someone who developed the software from verifying a subsequent revision of that software developed by someone else.

¹ SmartPlant is a registered trademark of Intergraph Corporation, Huntsville, Alabama.

NOTE 2: The Developer's supervisor only if the supervisor: (a) did not specify a singular design approach or rule out certain considerations and did not establish the design inputs used in the design, or (b) is the only individual in the organization competent to perform the verification.

- Reviews software requirements, evaluating the adequacy of the translation of functional requirements into software requirements.
- Provides assurance the software design approach is technically adequate and ensures internal completeness, consistency, clarity, and correctness of the software design.

3.6 Software Technical Support Analyst (STSA)

- Coaches owners in completing both SWAs and deliverables, when needed.
- Performs Stage 1 and 2 SMP reviews using the checklists in Table A-2 and Table A-3.
- Approves completed SMP and UCS for adequacy of documentation.

3.7 Tester

NOTE: The Tester cannot be the Developer.

- User that is trained on the UCS functions.
- Tests software in accordance with software-specific Test Plans and Test Cases.

3.8 Quality Assurance

NOTE: The individual assigned to the following Quality Assurance (QA) responsibilities cannot fulfill roles that perform and/or document SWAs; however, they can be a User.

- Approves completed SWA deliverables and software file for use.
- Verifies satisfactory completion of the software development cycle.

3.9 User

- Uses UCS in accordance with approved user training, instructions, and/or procedures.
- Initiates PERs for software problems/errors in accordance with TFC-ESHQ-Q_C-C-01.

4.0 PROCEDURE

(7.1.1, 7.1.2, 7.1.3)

To continue the SLC for UCS newly registered and graded in the Hanford Information System Inventory (HISI) in accordance with TFC-BSM-IRM_HS-C-01, proceed to Section 4.1.

To operate and maintain UCS (includes using UCS, problem reporting and corrective action, and SCRs) approved for use, proceed to Section 4.3.

To initiate a revision to an existing UCS approved for use, proceed to Section 4.4.

To retire UCS that is no longer needed, proceed to Section 4.5.

ENGINEERING	Document	TFC-ENG-DESIGN-C-32, REV H
	Page	5 of 32
UTILITY CALCULATION	Issue Date	July 21, 2015
SOFTWARE MANAGEMENT		

4.1 Software Life Cycle for New Utility Calculation Software Registrations

- | | |
|-----------------------|--|
| Owner | 1. Ensure TFC-BSM-IRM_HS-C-01 has been invoked and completed, including HISI registration and assignment of a software grade level. |
| Owner or Project Lead | 2. Create a new Software Management Plan (SMP) document in SPF. <ol style="list-style-type: none"> a. Complete the Document Release and Change Form (DRCF) entries; instructions are available on the SPF Forms & Instructions webpage, located at http://toc.wrps.rl.gov/rapidweb/SMART/index.cfm?pageNum=13. b. Attach a copy of the SMP for Utility Calculation Software template file obtained from the Information Resource Management (IRM) webpage. c. Assign, at a minimum, the following reviewers/approvers: <ul style="list-style-type: none"> • ITR (Software Requirements Verification) • STSA (Stage 1) • ITR (Software Design Verification) • STSA (Stage 2) • Project Lead • Software Owner • Quality Assurance Manager • Laboratory Configuration Control Board (for software supporting the 222-S Laboratory). d. Attach the Software Management workflow to the SMP and proceed to Section 4.2. |

4.2 Software Development Cycle

The software development cycle is implemented via SPF workflow attached to the SMP document. The SLC phases are performed within the limits of the workflow.

Each SLC phase involves the completion and documentation of required SWAs prior to the software being used and/or relied on. The SLC is implemented in this procedure using a graded approach that will be reflected in the SMP. TFC-PLN-02 and TFC-BSM-IRM-STD-01 require all SWAs and their associated deliverables be completed; however, the implementation (i.e., level of rigor applied to the activity or the level of detail in the objective evidence) can vary with consideration given to the level of risk or consequence of failure of the software reflected in the Software Grade Level (A/B/C/D).

- “Full” implementation of a particular SWA requires that all essential deliverables are completed to the degree necessary to ensure and provide objective evidence that the work activity is performed in a traceable, planned, and orderly manner with thorough and detailed coverage of content. Recommended evidence at this level includes items such as diagrams, schemas, flowcharts, formalized and documented analysis techniques. This is

a high priority, detailed activity and commensurate with the risks associated with software failure.

- “Graded” implementation allows for less formality in the required SWA and associated deliverables. The same SWAs and deliverables are required for a “graded” implementation as for “full” implementation, but the objective evidence of compliance can include less detail, or mapping of equivalency to other documents with less structured format.
- “Optional” implementation refers to work activities that are not required but are feasible and may add value to the overall software project.
- “N/A” or “Not Applicable” means the SWA is not required and there is no associated deliverable.

Refer to Table A-1 for the implementation level (i.e., “Full,” “Graded,” “Optional,” or “N/A”) to be applied to the SWAs and associated deliverables for each SLC phase based on the Software Grade Level (A/B/C/D).

4.2.1 Planning Phase

NOTE: Typically, there is no procurement activity associated with UCS development unless it is developed by a separate third-party or an add-in is needed.

- | | |
|--------------------|---|
| Owner/Project Lead | <ol style="list-style-type: none">1. If procuring an add-in software to be used with the UCS, process the add-in in accordance with TFC-BSM-IRM_HS-C-01 as a new software project.2. Complete the Planning Phase SWAs deliverables to the Implementation Level specified in Table A-1 using the directions in the SMP template, then proceed to Section 4.2.2. |
|--------------------|---|

4.2.2 Requirements Phase

This section addresses the preparation, review, and approval of the software requirements. Attachment B, Section 1.1, provides a standard for developing good requirements statements.

NOTE: Traceability to the design and test cases is documented on the Requirements Traceability Matrix (RTM), which occurs in subsequent SLC phases.

- | | |
|--------------|--|
| Project Lead | <ol style="list-style-type: none">1. Complete the Requirements Phase SWA deliverables to the Implementation Level specified in Table A-1 using the directions in the SMP template and the standard in Attachment B, Section 1.1.2. Ensure the RTM reflects all requirements and software functions to be traceable and testable.3. Place the SMP with completed Requirements Phase deliverables under configuration management in SPF. |
|--------------|--|

4. Complete the SPF workflow step to submit the SMP with completed Requirements Phase SWA deliverables for verification, and proceed to Section 4.2.2.1.

4.2.2.1 Software Requirements Verification

This section establishes the process for performing and documenting an independent technical review of software requirements to evaluate the adequacy of translation of the functional requirements into software requirements. Software requirements verification ensures acceptance criteria are established and the requirements are verifiable. For any revisions to requirements, re-verification shall be performed to ensure changes to requirements do not cause unintended defects in the requirements.

- | | |
|--------------|--|
| ITR | <ol style="list-style-type: none"> 1. Perform an independent technical review of the Requirements Phase deliverables, evaluating: the requirements for technical adequacy and completeness; the translation of functional requirements into software requirements; and, the adherence to Attachment B, Section 1.1. <ol style="list-style-type: none"> a. If no comments are warranted or previous comments are resolved, record Signature in SPF as the ITR (Software Requirements Verification) and complete the workflow step to approve the requirements baseline; otherwise, b. Document comments in SPF via the electronic comments feature, attaching an electronic redlined file to the comment if one was produced and complete the workflow step in SPF. |
| Project Lead | <ol style="list-style-type: none"> 2. If there are review comments to resolve, incorporate comment resolutions into the SMP and reject the workflow step to return the SMP to the ITR in Step 1 to obtain agreement on the resolutions. 3. If there are no review comments to resolve and the ITR (Software Requirements Verification) signature is recorded in SPF, complete the workflow step and proceed to Section 4.2.3. |

4.2.3 Design and Implementation Phase

This section addresses the design and implementation SWAs performed to the approved requirements baseline from the Requirements Phase. This section addresses developing the spreadsheet file to the planned design.

NOTE: Unit testing (as defined in Section 5.0) is performed on an ad hoc basis by the Developer and requires no documentation.

- | | |
|-----------|---|
| Manager | <ol style="list-style-type: none"> 1. Ensure personnel assigned to design and develop the software (i.e., the Developer) complete training course 356123, WRPS Software Quality Assurance. |
| Developer | <ol style="list-style-type: none"> 2. Ensure the application is developed using a site-approved (i.e., “Operational” in HISI) commercial-off-the-shelf spreadsheet application platform (e.g., Microsoft Excel®, Mathcad, Igor, etc.). |

3. Design the software considering the practices described in Attachments B, Section 2.0.
4. Include a “Documentation” section or worksheet inside the application file that includes at a minimum, the Owner Name, SMP document number and revision number, and HISI number.
5. Describe the planned software design and design approach in the Software Design Description (SDD) section of the SMP per direction in the template.

NOTE: Software design verification can begin upon completion of the software design in accordance with Section 4.2.3.1, but is formally documented and signed off by the ITR in parallel with the Stage 2 Review in Section 4.2.5.1.

6. Develop the spreadsheet file according to the design in the SMP.
 - a. Perform unit testing in parallel with development.
 - b. As needed, return to Step 3 when the planned design cannot be developed to satisfy the requirements to modify the design.
 - c. Attach the developed spreadsheet application file to the SMP document in SPF.
7. Complete the Design and Implementation phase SWA deliverables to the Implementation Level specified in Table A-1 per the directions in the SMP template.
8. Update the RTM with the design elements, tracing them to each applicable approved software requirement, then proceed to Section 4.2.4.

4.2.3.1 Software Design Verification

Software design verification is performed and documented to evaluate the technical adequacy of the design approach; ensure internal completeness, consistency, clarity, and correctness of the software design; and verify the software design is traceable to the software requirements. Software design verification includes, at a minimum, the following: considerations of the requirements related to the activities of preparing the software for acceptance testing, and review of test results.

NOTE 1: The SMP document specifies which software design verification method to use.

NOTE 2: Software design verification can begin upon completion of the software design in Section 4.2.3, but is formally documented and signed off by the ITR in parallel with the Stage 2 Review in Section 4.2.5.1.

ITR 1. Perform an independent technical review in accordance with method specified in the SMP, evaluating: the technical adequacy of the design approach and ensure internal completeness, consistency, clarity, and correctness of the software design and verify the software design is traceable to the software requirements.

Project Lead 2. Document the review comments and their disposition and retain until they are incorporated into the updated software. Comments not incorporated, and their disposition, shall be retained until the software is approved for use.

3. Ensure any requirements that are not met in the design activity are revised to reflect the final product using the same process as the original requirements.

4.2.4 Test Phase

Project Lead 1. Complete the Acceptance Test Plan deliverables to the Implementation Level specified in Table A-1, using the directions in the SMP template and Attachment B, Section 1.2.

2. Update the Requirements Traceability Matrix deliverable in the SMP with Test Case information, ensuring each requirement has at least one associated test case and each test case has at least one associated requirement.

3. Complete the In-Use Tests deliverable to the Implementation Level specified in Table A-1 using the directions in the SMP template.

4. Place the SMP with completed Acceptance Test Plan, Requirements Traceability matrix, and In-Use testing deliverables under configuration management in SPF, checking-in the SMP template to establish a baseline.

5. Complete the workflow step to submit the SMP baseline for Stage 1 Review and proceed to Section 4.2.4.1.

4.2.4.1 Stage 1 Review

STSA 1. Review the SMP Stage 1 SWA deliverables in accordance with the checklist in Table A-2.

a. If no comments are warranted or previous comments are resolved, record Signature in SPF as the STSA (Stage 1) and complete the workflow step to approve the baseline; otherwise,

b. Document comments in SPF via the electronic comments feature, attaching an electronic redlined file to the comment if one was produced and complete the workflow step.

ENGINEERING	Document	TFC-ENG-DESIGN-C-32, REV H
UTILITY CALCULATION	Page	10 of 32
SOFTWARE MANAGEMENT	Issue Date	July 21, 2015

- Project Lead
2. If there are review comments to resolve, incorporate comment resolutions into the SMP and reject the workflow step to return the SMP to the STSA in Step 1 to obtain agreement on the resolutions.
 3. If there are no review comments to resolve and the STSA signature is recorded in SPF, complete the workflow step and proceed to Section 4.2.5.

4.2.5 Acceptance Testing

Acceptance testing consists of testing the software to ensure all of the requirements have been met. Acceptance testing of a UCS satisfies the requirements for a functional configuration audit. Validation of UCS consists of confirming it is acceptable for use. The requirement for a physical configuration audit is satisfied in the review and approval of the completed SMP in SPF.

- Manager
1. Ensure personnel assigned to evaluate the software (i.e., the Tester, the ITR) complete training course 356123, WRPS Software Quality Assurance and:
 - a. Complete prescribed User Training within the SMP
 - b. Are not the individual(s) who designed and implemented the design for the software (i.e., not the Developer).
- Project Lead
2. Ensure testing is controlled, performed, and documented in accordance with the Acceptance Test Plan in the SMP.
- Tester
3. Test the software using the Acceptance Test Plan in the SMP.
 - a. Ensure the software requirements and acceptance criteria are met.
 - b. Document testing results (i.e., Pass/Fail, issues, and comments) on each Test Case.
- Project Lead
4. Append the completed Test Cases to Appendix B – Test Case Execution Record of the SMP and complete the section per direction in the template.
 5. Evaluate the test results for acceptability of the software to adequately and correctly perform all intended functions (i.e., specified software requirements), ensuring the software:
 - Does not perform adverse unintended functions.
 - Handles abnormal conditions, events, and credible failures.
 - Does not degrade the system either by itself, or in combination with other functions or CIs.

- a. Proceed to Step 6 to address any issues and comments in the test results, otherwise
- b. Proceed to Step 7.
6. Resolve issues and Tester comments.
 - a. Incorporate comment resolutions into the software and SMP with the Developer.
 - b. Return to Step 3 to re-test the software and obtain agreement on comment resolutions.
7. Complete the Acceptance Test Report deliverable to the Implementation Level specified in Table A-1 using the directions in the SMP template.
8. Place the SMP with all life cycle phase SWA deliverables completed under configuration management in SPF, checking-in the SMP template to establish a baseline.

4.2.5.1 Stage 2 Review and Approval for Use

Software design verification includes, at a minimum, the following: considerations of the requirements related to the activities of preparing the software for acceptance testing, and review of test results.

NOTE 1: The SMP document specifies which software design verification method to use.

NOTE 2: Steps 1 through 3 occur in parallel in the SPF workflow.

ITR

1. Perform an independent technical review, evaluating: the technical adequacy of the design approach and ensure internal completeness, consistency, clarity, and correctness of the software design and verify the software design is traceable to the software requirements.
 - a. If no comments are warranted or previous comments are resolved, Record Signature in SPF as the ITR (Software Design Verification) and complete the workflow step to approve the requirements baseline; otherwise,
 - b. Document comments in SPF via the electronic comments feature, attaching an electronic redlined file to the comment if one was produced and complete the workflow step in SPF.

STSA

2. Review the SMP Stage 2 SWA deliverables in accordance with the checklist in Table A-3.
 - a. If no comments are warranted or previous comments are resolved, record Signature in SPF as the STSA (Stage 2) and complete the workflow step to approve the baseline; otherwise,

- | | | |
|--------------------------|----|---|
| | b. | Document comments in SPF via the electronic comments feature, attaching an electronic redlined file to the comment if one was produced and complete the workflow step. |
| Owner/QA/Other Approvers | 3. | Review the completed SMP document and spreadsheet file for completeness and readiness for use. |
| | a. | If no comments are warranted or previous comments are resolved, record Signature in SPF and complete the workflow step to approve the baseline; otherwise, |
| | b. | Document comments in SPF via the electronic comments feature, attaching an electronic redlined file to the comment if one was produced and complete the workflow step. |
| Project Lead | 4. | If there are review comments to resolve, incorporate comment resolutions into the SMP and reject the workflow step to return the SMP for review to Steps 1 through 3 to obtain agreement on the resolutions. |
| | 5. | If there are no review comments to resolve and the ITR (Software Design Verification), STSA (Stage 2), Software Owner, and Quality Assurance signatures are recorded in SPF, complete the workflow step. |
| Manager | 6. | Review the completed SMP document and verified spreadsheet file for completeness and readiness for use. |
| | a. | If no comments are warranted or previous comments are resolved, Record Signature in SPF as the Manager and complete the workflow step to approve the UCS for use; otherwise, |
| | b. | Document comments in SPF via the electronic comments feature, attaching an electronic redlined file to the comment if one was produced and reject the workflow step to return the SMP to the Owner in Step 4. |
| Project Lead | 7. | Submit the SMP document in SPF to the Document Service Center to be released, then proceed to Section 4.2.6. |

4.2.6 Installation and Checkout Phase

This section of the procedure establishes the process for installing the UCS into a production environment (e.g., a share drive governed by access control permissions). Once installed, in-use tests are performed to confirm acceptable performance of the UCS within the production environment.

SPF is the repository for UCS approved for use. The “master” application file is attached to the SMP document in SPF, where configuration control over the file is maintained (i.e., the master spreadsheet file cannot be modified independent of the approved SMP document baseline).

- Owner/Project Lead
1. Prior to the initial release of a UCS, configure and control access to the production environment per direction in the Installation Plan section of the SMP.
 2. Install (i.e., Save) the current revision of the spreadsheet file obtained from SPF to the production environment.
 3. Perform and document in-use testing for the installation of the software as prescribed in the In-Use Testing section of the SMP.
 - a. If testing does not pass, proceed to Section 4.3.1; otherwise,
 - b. Proceed to Step 4.
 4. Change the Status field in the Core Information tab of the HISI entry to “Operational” and click on the “Update Core Information” button.
 5. Notify Users when the UCS is available for use, and proceed to Section 4.3.

4.3 Maintenance and Operations

This section applies to UCS that is approved for use, installed in an operating environment, and has an “Operational” status in HISI. This section of the procedure establishes the process for controlling use of the software, training personnel on use of the software, and use of the software by Users.

- Owner
1. Control the use of the software in accordance with the SMP, including:
 - Application documentation
 - Access control specifications
 - In-Use Tests
 - Configuration Change Control (see Section 4.3.2)
 - Problem Reporting and Corrective Action (see Section 4.3.1)
 - If the UCS is no longer needed, proceed to Section 4.4.
 2. Ensure personnel assigned to use (User) the software meet the User Qualifications, complete User Training, and follow the User Documents as defined in the SMP.

NOTE: The results from each use of UCS may be documented and released in a variety of document types listed in TFC-ENG-DESIGN-C-25. The typical document types used include but are not limited to: Engineering Calculations, Waste Compatibility Assessments, Model Results Documentation, Other Software Documents, General Documents or Reports.

- User
3. Use the software in accordance with the approved User Documents, User Training, and specified governing procedures, referring to Section 4.3.1 if a software problem is encountered.

ENGINEERING	Document	TFC-ENG-DESIGN-C-32, REV H
	Page	14 of 32
UTILITY CALCULATION	Issue Date	July 21, 2015
SOFTWARE MANAGEMENT		

4. Input data manually entered into a UCS must be checked in accordance with TFC-ENG-DESIGN-C-10 or the governing procedure for the type of document being used to document the UCS results as part of the checking, review, and release process for the document.

4.3.1 Problem Reporting and Corrective Actions

This section applies when a User encounters a problem or error (as defined in Section 5.0) while using “Operational” software. TFC-ESHQ-Q_C-C-01 describes the process used by the Tank Operations Contractor (TOC) for the timely identification, evaluation, and correction of issues adverse to the environment, safety, health, and quality. This section provides additional steps for identifying, evaluating, and correcting problems or errors related to software.

- | | |
|------------|--|
| User | 1. Communicate software problem to the Owner. |
| User/Owner | 2. For grade A, B, C, or D software, if an error exists that adversely impacts the product output, initiate a PER in accordance with TFC-ESHQ-Q_C-C-01, and proceed to step 3; otherwise, proceed to Step 6. |
| Owner | 3. Notify the assigned STSA. |

NOTE: It is recommended the Owner review HISI to identify and notify other Hanford/DOE-complex users of the software potentially affected by the error.

4. Notify Users of impacts to the use of the software and upon issuing a revision to the software.
5. If the usability of the software is compromised in that the software can no longer perform its intended function(s), change the Status in HISI Core Information to “On Standby.”
6. If a Software Change is necessary, proceed to Section 4.3.2, otherwise, resolve with User and return to Operations and Maintenance.

4.3.2 Software Change Request (SCR)

This section describes the process for documenting and approving proposed software changes prior to incorporating the changes into the software baseline. Changes include items such as:

- | | |
|-----------------------|--|
| User/Owner/Manager | 1. Identify the need to revise the UCS and disseminate to the Owner. |
| Owner or Project Lead | 2. Document the proposed change in an IRM Software Change Request (site form A-6006-767) per the form instructions. |
| | 3. Evaluate requested change to the software to determine whether to approve the requested change. <ol style="list-style-type: none"> a. If an existing UCS is to be revised to create a new UCS for a different purpose, exit this procedure and go to TFC-BSM-IRM_HS-C-01 to initiate the new software project. |

- b. Otherwise, document the evaluation and the approval or rejection on the SCR.
4. Process the SCR for release in accordance with TFC-ENG-DESIGN-C-25, assigning both the Owner and Manager as approvers on the SPF document.
5. Proceed to Section 4.4 to initiate the software development cycle.

4.4 Revisions to Existing Utility Calculation Software Registrations

NOTE: Proposed changes (e.g. bug fix, enhancements, change to existing functionality, etc.) to the software are documented and approved via the SCR as described in Section 4.3.2 prior to initiating a revision to the software in this section

- | | |
|-----------------------|---|
| Owner | <ol style="list-style-type: none"> 1. Ensure an SCR has been approved and issued in accordance with Section 4.3.2. 2. If the proposed change potentially impacts the currently assigned software grade level, ensure TFC-BSM-IRM_HS-C-01 is invoked to obtain an appropriate software grade level. |
| Owner or Project Lead | <ol style="list-style-type: none"> 3. Initiate a revision to the SMP document in SPF. <ol style="list-style-type: none"> a. Complete the DRCF entries; instructions are available on the SPF Forms & Instructions webpage. b. Ensure the most current revision of the SMP template file is in-use or is obtained from the IRM webpage. c. Assign, at a minimum, the following reviewers/approvers: <ul style="list-style-type: none"> • ITR (Software Requirements Verification) • STSA (Stage 1) • ITR (Software Design Verification) • STSA (Stage 2) • Project Lead • Software Owner • Quality Assurance • Manager • Laboratory Configuration Control Board (for software supporting the 222-S Laboratory) d. Attach the Software Management Workflow to the SMP. 4. If Planning Phase deliverables listed in Table A-1 need to be revisited, proceed to Section 4.2, otherwise proceed to Section 4.2.2. |

4.5 Retirement

Owner

1. When the UCS is no longer needed, retire software in accordance with the Software Retirement section in TFC-BSM-IRM_HS-C-01.

5.0 DEFINITIONS

Terms common to the WRPS software procedures are defined in TFC-BSM-IRM_HS-C-01 and TFC-BSM-IRM-STD-01. Terms unique to this procedure are retained within this procedure.

Baseline. A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for use and further development, and that can be changed only by using an approved change control process.

Computation spreadsheets. A simple spreadsheet containing only simple calculations to perform simple tasks without the use of macros, third-party add-ins, or links to other data sources (e.g., workbooks, databases, etc.); use is limited to preliminary or rough-order-of-magnitude work that does not support safety-significant equipment, safety basis or environmental compliance, technical documents, or data in other software. The status and extent of verification for computation spreadsheets shall be clearly communicated if presented to an external regulator.

Error. A condition deviating from an established baseline, including deviations from the current approved computer program and its baseline requirements.

In-Use Tests. Tests performed on a periodic basis or in response to a triggering event (e.g., upon installation into a production environment, operating system or Excel® version changes/patching) to preclude software errors, data errors, computer hardware failures, or instrument drift from affecting required performance by a software.

Macros. In Excel®, a macro is a Visual Basic® for Applications (VBA) module containing commands or functions, i.e., a series of actions used to automate a task or perform complex operations).

Multiple-use spreadsheet. Spreadsheet or spreadsheet template to be used multiple times or by multiple users for performing routine or standardized calculations or analyses that is independently pre-verified in accordance with this procedure and whose design inputs are independently verified in accordance with TFC-ENG-DESIGN-C-10 or other controlling procedure.

Problem. Any anomaly perceived to not be functioning properly or as expected.

Software Development Cycle. The activities that begin with the decision to develop a software product and end when the software is delivered. The software development cycle typically includes the following activities: software design requirements; software design; implementation; test; and sometimes, installation.

Software engineering. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Single-use spreadsheet. Spreadsheet to be used one time to perform a single set of calculations or analyses.

Unit Test. Tests performed on an ad hoc basis by the Developer in parallel to the Software Implementation SWA to check an individual element of the software (e.g., formulas, conditional formatting, custom-developed functions and subroutines [macro code], etc.) in isolation from the rest of the software.

Utility Calculation Software. Software that typically uses commercial-off-the-shelf spreadsheet applications as a foundation and user-developed algorithms or data structures to create simple software products.

6.0 RECORDS

The following records are generated during the performance of this procedure:

- Software Management Plan for Utility Calculation Software.
- Software Change Request (IRM Software Change Request [Site form A-6006-767]).

The record custodian identified in the Company Level Records Inventory and Disposition Schedule (RIDS) is responsible for record retention in accordance with TFC-BSM-IRM_DC-C-02.

7.0 SOURCES

7.1 Requirements

1. [TFC-BSM-IRM-STD-01](#), “Software Life Cycle Standard.”
2. TFC-PLN-02, “Quality Assurance Program Description.”
3. TFC-PLN-112, “Graded Approach to Quality.”

7.2 References

1. ATS-310 Section 8.14, “Laboratory Computer Software Management.”
2. IEEE Standard 830, “IEEE Recommended Practice for Software Requirements Specifications,” The Institute of Electrical and Electronics Engineers, New York, New York, 1993.
3. TFC-BSM-CP_CPR-C-05, “Procurement of Services.”
4. TFC-BSM-IRM_DC-C-02, “Records Management.”
5. TFC-BSM-IRM_HS-C-01, “Software Development, Implementation, and Management.”
6. TFC-BSM-IRM_HS-C-03, “Software Management.”
7. TFC-BSM-IRM_HS-C-09, “Software Administrative Installation.”

8. TFC-ENG-DESIGN-C-10, "Engineering Calculations."
9. TFC-ENG-DESIGN-C-25, "Technical Document Control."
10. TFC-ENG-DESIGN-C-65, "Commercial Grade Dedication of Software."
11. TFC-ESHQ-Q_C-C-01, "Problem Evaluation Request."

ATTACHMENT A – SOFTWARE QUALITY ASSURANCE TABLES

Table A-1. Graded Approach for Software Work Activities Matrix.

Software Life Cycle		Implementation Level (by Software Type and Software Grade Level)		Deliverables		
Phase	Software Work Activity (SWA)	Utility Calculation		Software Management Plan (SMP)		
		A/B/C	D	Section	Heading	
Planning	Software Project Management and Quality Planning	Graded	Graded	1.0	Introduction	
				1.1	Purpose	
				1.2	Scope	
				1.3	Assumptions and Constraints	
				1.4	Access Controls	
				1.5	Project Organization	
				1.7	Roles and Responsibilities	
				1.8	Software Tools	
				1.9	Applicable SQA Work Activities and Deliverables	
				1.10	Software Verification and Validation Plan	
		1.12	Records Management			
				Optional	1.6	Schedule and Budget Summary
	Full	Graded	2.10	Retirement Plan/Checklist		
	Graded	N/A	2.3	Alternatives Analysis		
	Training of Personnel in the Design, Development, Use, and Evaluation of Safety Software	Full	Optional	1.11	Training	
	Software Risk Management	Full	Optional	2.4	Risk Management	
	Software Configuration Management	Graded	Graded	2.6	Software Configuration Management Plan	
	Procurement and Supplier Management	Full	N/A	2.11	Acquisition	
Requirements	Requirements Identification and Management	Full	Graded	2.1	Functional Requirements Definition	
				2.7	Software Requirements Specification	
		2.2	Requirements Traceability Matrix			
	Software Safety	N/A	N/A	2.8	Software Safety Plan	
Design and Implementation	Software Design and Implementation	Graded	Graded	2.12	Software Design Description	
		Optional	Optional	2.13	Unit and System Testing	
	Software Verification and Validation	Graded	Optional	2.14	Technical and Peer Reviews	
	Software Project Management and Quality Planning	Graded	Optional	2.15	Software Installation Plan	
				2.5	Contingency Plan	
	Training of Personnel in the Design, Development, Use, and Evaluation of Safety Software		Full	Optional	2.16	User Qualification
					2.17	User Training
2.18					User Documents	
Test	Software Verification and Validation	Graded	Graded	2.9	Acceptance Test Plan	
				2.19	Acceptance Test Report	
				2.20	In-Use Tests	

ATTACHMENT A – SOFTWARE QUALITY ASSURANCE TABLES (cont.)

The following items in the Utility Calculation Software SQA Deliverables Review Checklists are *guidelines* that should be addressed for each deliverable.

NOTE: Reviewers documenting review comments in SPF should copy/paste the key point for consideration into the “Basis” field for the comment. A comment is not required for each checklist item if the key point to consider is satisfied in the SWA deliverable.

Table A-2. SMP SWA Deliverables Stage 1 Review Checklist.

Section	Heading	Key Points for Reviewer to Consider
1.0	Introduction	The software project to be managed is described including Introduction, Purpose, and Scope.
1.1	Purpose	
1.2	Scope	
1.3	Assumptions and Constraints	Any standards, conventions, work practices, metrics and Software Engineering Methods that will be used to perform work activities are identified.
1.4	Access Controls	Access control methods for protecting computer physical media from unauthorized access or inadvertent damage or degradation is identified.
1.5	Project Organization	The organization(s) responsible for performing the SQA work activities are identified and documented.
1.6	Schedule and Budget Summary	The resources necessary to perform the SQA work activities are identified and documented.
1.7	Roles and Responsibilities	
1.8	Software Tools	Software tools that will be used to support SQA work activities are described as well as their intended use, applicability, or circumstances under which it is to be used and limitations.
1.9	Applicable SQA Work Activities and Deliverables	The applicable SQA work activities and deliverables for the software project with minimum requirements are described.
1.10	Software Verification and Validation Plan	<ul style="list-style-type: none"> • (1.10.1) The required reviews for the deliverables identified in the SMP are clearly stated. • The software verification process is performed throughout the software life cycle and provides objective evidence whether the software and its associated life cycle activities and deliverables: <ul style="list-style-type: none"> a. Conform to requirements for all life cycle activities during each life cycle phase. b. Satisfy standards, practices, and conventions during life cycle phases. c. Successfully complete each life cycle phase and satisfy all criteria for initiating succeeding life cycle phases. d. Solve the right problem. e. Satisfy intended use and user needs.
1.12	Records Management	The section describes maintenance and storage of SQA deliverables
1.13	Definitions	Definitions for unique terms used in the SMP are documented.
2.3	Alternatives Analysis	Alternatives Analysis information is documented and complete.
1.11	Training	Any required training of the software project members that is necessary for them to perform SQA work activities are described.
2.4	Risk Management	Project risks are identified and documented in the “Risk Management” section and include: <ul style="list-style-type: none"> • Risk Identification • Risk Priorities • Risk Avoidance, Mitigation, or Transfer • Risk Management Policies and Procedures.

ATTACHMENT A – SOFTWARE QUALITY ASSURANCE TABLES (cont.)

Table A-2. SMP SQA Deliverables Stage 1 Review Checklist.

Section	Heading	Key Points for Reviewer to Consider
2.6	Software Configuration Management Plan	<ul style="list-style-type: none"> • Configuration activities are documented in the “Software Configuration Management Plan. • Each Configuration Item has a unique identification numbering scheme. • All CIs are identified including documentation deliverables. • (2.6.4) A plan is documented to control changes to CIs during the operations and maintenance phases. • (2.6.5) A plan is in place to ensure configuration status accounting activities to track the status of CIs throughout the SLC and during changes. • (2.6.6) Configuration audits and reviews are conducted at a determined frequency to verify that CIs include the following: • (2.6.8) If required, the Data Security Plan is described and documented. • (2.6.8) If required, the Data Security Plan addresses OOU, SUI, or PII information.
2.11	Acquisition	<ul style="list-style-type: none"> • Methods for error reporting and corrective action are described. • Media control methods to store, copy, and recover electronic files are identified.
2.1	Functional Requirements Definition	<ul style="list-style-type: none"> • The Functional Requirements Definition is complete. • A unique number is assigned for each functional requirement.
2.7	Software Requirements Specification	<ul style="list-style-type: none"> • The Software Requirements Specification defines specific requirements of the software. • A unique number is assigned for each specific requirement that corresponds to the functional requirements.
2.2	Requirements Traceability Matrix	The Requirements Traceability Matrix is complete and lists unique Functional Requirements, Specific Requirements, Design Elements, and Test Case IDs
2.12	Software Design Description	<ul style="list-style-type: none"> • The Software Design Description identifies the overall structure (control and dataflow) and the reduction of the overall structure into physical solutions to include: <ul style="list-style-type: none"> a. Numerical methods b. Mathematical methods c. Physical models d. Control flow e. Control logic f. Data flow g. Data structures h. Process structures i. Applicable relationships between data structure and process structures. • For model development activities, ensure any unique applicable requirements addressed in the QAPD Section 2.7.29 (Model Development, Use, and Validation) are addressed in the design document. • Ensure a unique number is assigned for each software design element that corresponds to the specific requirements it satisfies. • Implementation deliverables are completed.
2.13	Unit and System Testing	Unit and System testing are informal and not required to be documented for Utility Calculation Software.
2.14	Technical and Peer Reviews	Any additional technical and peer reviews that were performed, excluding those reviews already described in Section 1.10 and 1.10.1 of the SMP are documented.
2.15	Software Installation Plan	Describes how the software application will be installed in a production environment and distributed for use.
2.5	Contingency Plan	If required, the Contingency Plan is described.
2.16	User Qualification	If required, describes what qualifications a User must possess prior to using the software.
2.17	User Training	Describes any training the User must complete prior to using the software.
2.18	User Documents	User documentation is developed (e.g. Software Installation Procedures, Operating Manual, and maintenance manual.)
2.8	Software Safety Plan	Not applicable for Utility Calculation Software per TFC-PLAN-02.

ATTACHMENT A – SOFTWARE QUALITY ASSURANCE TABLES (cont.)

Table A-2. SMP SQA Deliverables Stage 1 Review Checklist.

Section	Heading	Key Points for Reviewer to Consider
2.9	Acceptance Test Plan	The Acceptance Test Plan is completed based on the RTM and ensures: <ul style="list-style-type: none"> a. It adequately and correctly performs all intended functions. b. Properly handles abnormal conditions and events, as well as credible failures. c. Does not perform adverse unintended functions. d. Does not degrade the system either by itself or in combination with other functions or CIs.
2.20	In-Use Tests	In-Use test cases are defined as well as the periodic or triggering conditions that if met, would drive performance of an In-Use test.
2.10	Retirement Plan/Checklist	The methods for retiring the software, including prevention of routine use and preservation of software files, data, and documentation are described.

Table A-3. SMP SQA Deliverables Stage 2 Review Checklist.

Section	Heading	Key Points for Reviewers to Consider
2.19	Acceptance Test Report	<ul style="list-style-type: none"> • All CIs are under configuration control. • Test results are documented in the Acceptance Test Report section and should include: <ul style="list-style-type: none"> a. Computer program version being tested. b. Computer hardware tested and its configuration during the test c. Test equipment and calibrations, where applicable d. Simulation models used, where applicable e. Date of test f. Tester or data recorder g. Reference to the applicable test plan and test cases, and a description of any changes in evaluation (validation) methods, inputs, or test sequence h. Tests and test results that supported reviews of software implementation. i. Software security measures in place, as required. j. Test results and conclusions that the reported results adequately address the specified test requirements and acceptance criteria for the software. k. Observation of unexpected or unintended results and their dispositions. l. Actions taken in connection with any deviations. m. Independent reviewer evaluating test results by a responsible authority to ensure all requirements have been met.

ATTACHMENT B – UTILITY CALCULATION SOFTWARE STANDARDS AND PRACTICES²**1.0 STANDARDS****1.1 DEVELOPING SOFTWARE REQUIREMENTS**

Requirements describe what a UCS is supposed to do in detail. The software requirements specification (SRS) is the collection of all requirements (i.e., functional, technical, software engineering, performance, security, user access control, interface, safety, installation, and design constraints) that define the performance of a UCS. A good requirement is:

- *Correct* – the stated performance satisfies an actual need.
- *Consistent* – does not negate, conflict with, or override another requirement.
- *Unambiguous* – has only one interpretation.
- *Verifiable* – is stated in concrete terms/measurable quantities; can be objectively tested.
- *Ranked for importance* – assigned a priority level, such as:
 - *Essential* – it must be met before accepting the software.
 - *Conditional* – it enhances the software if met but does not render the software unacceptable if absent.
 - *Optional* – it implies a class of functions that may or may not be worthwhile, such as functionality that exceeds the customer’s requirements.
- *Traceable* – its origin is clear and explicitly referenced in documentation; two types of traceability are prescribed:
 - *Forward traceability* – requirements traced to all documents spawned by the SRS.
 - *Backward traceability* – the source of the requirement is explicitly referenced in documents written earlier in the software development life cycle.

Consider the following when writing requirements for UCS:

- What calculation or analysis needs to be performed by the spreadsheet?
- What data/units will be used as input data?
- Does the input data need to be validated and how will it be validated?
- What results/output units are to be produced?
- What equations or formulas are to be applied to the input data?
- How is it to be tested; what acceptance criteria determine a test is successful?

A good software requirements specification (SRS) is:

- *Complete* – identifies all functionality, performance, design constraints, interface, limits, or bounds for the spreadsheet.
- *Modifiable* – it can be easily changed while preserving the characteristics of a good SRS.
- *Consistent* – the requirements do not conflict with or defeat another requirement.
- *Unambiguous* – every requirement stated therein has only one interpretation.
- *Traceable* – the requirements can be traced to their origin throughout the SLC.
- *Verifiable* – every requirement stated therein can be objectively tested or observed using concrete terms and measurable quantities.

² Attachment content adapted from IEEE Std 830, 1993, IEEE Recommended Practice for Software Requirements Specifications.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)**1.2 DEVELOPING TEST PLANS AND TEST CASES**

A test plan identifies all the tests needed to demonstrate (verify) the stated UCS requirements have been met, the developed UCS is suitable for its intended use (validation), and nothing has gone wrong during its use. A test case identifies conditions or variables to be used when performing a test. Test cases can be used when testing all branches of a decision logic or may involve introducing wrong data types or out-of-range data values to test error trapping and conditional formatting.

Verification of all UCS by testing is required prior to their use or reliance on their results. Verification testing consists of three stages: (1) reviewing and verifying the assumptions/ limitations of the spreadsheet are reasonable and adequate; (2) verifying cell formulas accurately reflect and perform intended functions, equations, and calculations; and (3) verifying input data used in calculations is entered correctly. The following represent good practices for developing spreadsheet test plans and test cases.

General Test Plan Considerations

1. A Test Plan identifies:
 - The scope of testing (i.e., unit, integration, acceptance, installation, and in-use).
 - When testing needs to be performed including frequency of in-use testing.
 - Acceptance criteria to be used to determine if tests pass and requirement is met.
 - Acceptance criteria for agreement when using alternate calculations to verify results.
 - How to document test results.
 - How to disposition unexpected or unintended test results prior to test approval.
 - What constitutes completion of acceptance testing.
 - The applicability of Installation testing.
 - When to use in-use testing in response to changes or triggering events.
 - When problem reporting can be used in lieu of in-use testing.
 - The level of review and in-use retesting required in response to a change in the operating system (for safety spreadsheets).
2. A Test Plan for a UCS revision includes selective retesting of the UCS to detect unintended adverse effects introduced by changes and demonstrates requirements are still met.
3. When preparing a Test Plan, ensure:
 - Each requirement corresponds to at least one test case.
 - Each test case corresponds to at least one requirement.
 - Testing covers the full range of intended operations.
 - Links to external workbooks are valid and function correctly.
 - Labels used in the workbook are correct and provide the correct units.
 - Input Data is appropriate and consistent with its original source.
 - Formulas implement the intended functions or calculations correctly.
 - Error checking/formula auditing tools in Excel[®] are used to detect obvious errors.
 - WRPS Calculation Checklist (A-6006-215) is reviewed for applicability to test.
 - Initial or permanent input data included in “master” multiple-use UCS is verified.
 - Macros perform their intended purpose.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)

- Macros do not introduce any unintended consequences (for safety spreadsheets).
- Testing demonstrates the spreadsheet properly handles abnormal conditions and events as well as credible failures (for safety spreadsheets).
- Tests demonstrate encoded mathematical models produce correct solutions within the defined limits for each parameter employed and valid solutions to the physical problem being addressed (Design Analysis spreadsheets).

Testing Input Data

1. Identify the type of input data (e.g., text, date, or number) and the valid range for the data values. Perform tests to check for unintended consequences from entering incorrect data types or out of range data values.
2. Identify the degree of data entry validation that will be performed:
 - a. For safety UCS, check 100% of manual data entries for accuracy by comparing the entered data to the data source.
 - b. For non-safety UCS, check manual data entries for accuracy by comparing a sampling of entered data against the data source, where sample size is based on the total number of input data entries for the spreadsheet as indicated below.
 - Less than 100 data elements, verify all input items and correct all errors.
 - For 101-500 data elements, verify a 25% random sample of input data is correct. If the observed error rate is less than 1% of the sample, complete the verification by correcting the errors. If the error rate exceeds 1%, correct the errors and draw another 25% sample and repeat the procedure. If the error rate for the second sample is less than 1%, complete the verification by correcting the errors. If the error rate for the second sample exceeds 1%, correct the errors and verify all remaining data elements are correct.
 - For over 500 data elements, verify a 10% random sample of the input data taken at is correct. If the error rate is less than 1% of the sample, complete the verification by correcting the errors. If the error rate exceeds 1%, correct the errors and draw another 10% sample, and repeat the procedure. If the error rate for the second sample is less than 1%, complete the verification by correcting the errors. If the error rate for the second sample exceeds 1%, correct the errors and verify all remaining data elements are correct.
 - c. For all spreadsheets, test:
 - Datasets transferred by an automatic process by an appropriate method from above depending on the amount of data transferred.
 - Datasets copied and pasted from another UCS or application, by checking the first and last values for appropriate placement and alignment in the spreadsheet cells and checking that the sum of copied cells matches the sum of the original cells.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)

3. Equivalent tests for large blocks of data include using conditional formatting, formula to compare two data values, and copying one set of data and pasting on a second set of data using Paste Special and selecting Difference.

Testing Formulas

Three methods exist for testing formulas. Method (1), if used, must be supplemented with method (2) or method (3) to provide adequate verification. Testing (2) and Alternate calculations (3) are the preferred methods to verify formulas.

1. For each unique cell formula, review the cell formula to determine it performs the intended algebraic equation or logical expression; for non-unique formulae (i.e., the same formula applied to a series of data values), review the overall series for consistency to ensure the formulae have not been corrupted or incorrectly copied.
2. Compare spreadsheet results with field measurements or analytical data, when appropriate, to confirm the results are in reasonable agreement with the measured values.
3. Perform an alternate calculation by hand, alternate spreadsheet, Mathcad, or other means, and compare the results to the spreadsheet results to verify the spreadsheet performs the intended functions and returns the correct results.

NOTE: The results of hand calculations need to be “close enough” to the spreadsheet results to give confidence in the reasonableness of the results. This qualification is added since discrepancies in rounding or methodology can result in small differences, up to 5%. Care needs to be exercised when such differences are observed to make sure the difference is due to rounding or methodology rather than an error in the spreadsheet.

Verifying and Testing Macros

Macro verification involves checking the macro performs its intended function correctly either by testing it or by using an alternate means to verify its correct performance; verification does not typically constitute checking the VBA code line by line.

Macro Testing demonstrates the macro:

- Performs the intended function correctly.
- Buttons, dialog boxes, and menus work as intended.
- Cell references are correct.
- Handles duplicate or non-existent files correctly (for macros managing files).
- Closes appropriately when the User presses [Esc] during macro execution.
- Handles incompatible data types, including error-trapping to avoid damaging crashes.
- Allows Users to change default macro settings, e.g., worksheet names.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)**2.0 PRACTICES****2.1 SPREADSHEET DESIGN**

A number of good standard practices for spreadsheet development are provided that help ensure spreadsheet products are easier to understand, verify, and use. While not mandatory, these practices should be followed to the extent practical to reduce the risk of errors in spreadsheet products.

Use A Modular Design

Building spreadsheets in which everything is contained on a single worksheet is neither recommended or the generally accepted practice for good spreadsheet design. As the number of rows/columns increases, the ease of managing, verifying, and using the spreadsheet is reduced. Reserve this practice for relatively small and simple spreadsheets. Utilizing multiple worksheets in a well-designed modular spreadsheet, with each worksheet dedicated to a specific purpose, is typically easier to manage, navigate, and verify:

- To avoid inserting columns between headings at the top of the spreadsheet and accidentally splitting different heading sets located in latter rows, use separate worksheets for each set of column headings to allow for simple spreadsheet expansion.
- To develop a spreadsheet quickly to perform functions against several similar datasets (e.g., data for different waste tanks), standardize the layout of a single worksheet to perform the functions and create a copy of the standardized worksheet for each dataset.

Separate Inputs, Calculations, and Results

Use separate worksheets to provide a clear separation between where the user enters data (input data), the spreadsheet calculations occur, and the final results are posted. This makes it easier to: input significant volumes of data while reducing the risk of overwriting calculations; verify and maintain the spreadsheet by keeping the calculations together; and print final reports/distinguish results from their basis.

For smaller single worksheet spreadsheets, separate input data, calculations, and final results using one of two arrangements:

- *Diagonal* – Each section is added below and to the right of the section preceding it, enabling addition/removal of rows/columns without impact to the other sections. (preferred approach)
- *Top-to-bottom* – the top of the worksheet contains the input data, the middle section contains the calculations, and the bottom section contains the results.

Always color coding data entry cells helps indicate the input data area of the worksheet.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)**Keep Formulas Simple**

Separate long formulas containing multiple functions into their intermediate steps, limiting each cell to one or two intermediate functions. Although more rows or columns will be required to achieve this, the resulting formulas will be easier for a verifier to follow and check, less prone to error, and result in a more robust final product.

Formulas often become too long by including multiple nested IF commands. Frequently, this problem is overcome using table lookup functions, such as VLOOKUP/HLOOKUP.

Use One Formula per Row or Column

As far as possible, formulas should be written so that a single formula can be copied across the entire row of calculations or down the entire column. Advantages to using one formula per row/column include:

- *Straightforward documentation* – only one definition for each row/column.
- *Quicker development* – formulas are built once; then copied across/down all other cells.
- *Effective testing* – especially when formula maps generated by software (e.g., Spreadsheet Professional) is used to distinguish between unique and copied formulas.
- *Robust updating and maintenance* – developers know to copy a formula across/down a row/column each time a formula is changed.

Avoid Creating Circular References

Design spreadsheets so the logic flows from front-to-back, top-to-bottom, and left-to-right to make the product easier to understand and reduce the risk of introducing circular references to the calculation.

For example, each cell reference in a formula should refer either to an input or calculation further up the same column or to an input or calculation on an earlier worksheet in the workbook, or to a calculation further down the worksheet but in an earlier column.

Include a Documentation Section or Worksheet

Provide documentation within the spreadsheet, the extent of which depends on the software grade and complexity of the spreadsheet. Documentation will provide key information about the spreadsheet such as operating system, spreadsheet functions, interfaces to other software applications, performance requirements, installation considerations, design inputs, and design constraints to guide future verifiers and users without recourse to the developer.

Incorporate Software Design Considerations

Follow software design principles or considerations when designing a spreadsheet. One example of a software design approach is to analyze the software to identify potential problems and then design into the software measures to mitigate the consequences of those problems.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)

The analysis should include an evaluation of external and internal abnormal conditions and events that can affect the spreadsheet. Another example of a good software design feature is the use of error trapping. These are described in more detail in Attachment D.

2.2 HINTS AND TIPS FOR DEVELOPING EASY-TO-USE SPREADSHEETS

While not being appropriate to all spreadsheets, the ideas in this section can be useful ways to make the spreadsheet easier to understand and use.

Use Color Coding

Color coding is a useful technique to make spreadsheets easier to use and can be used to more clearly distinguish between separate sections of a spreadsheet as discussed below. If you use dark colors for text, then black and white printouts will be largely unaffected, but the difference will be visible on screen. Using pale colors for shading cells will be visible on printouts, but can become indistinct on photocopies or faxes.

Examples of using color coding include:

- Separating types of inputs
- Highlighting cells which are linked to external spreadsheets
- Separating inputs from calculations by making all cells requiring inputs the same color
- Highlighting the cells modified when you made a particular change to the spreadsheet, making it much easier to track changes or, if necessary, reverse the change at a later date.

Include Units

The inclusion of a units column on every worksheet will tell you what is expected in each input cell and what is being presented in calculations. Verification of spreadsheet formulas should verify the dimensional consistency of formulas.

Use Natural Language Formulas

For simple spreadsheets, include an extra column containing natural language translations of the formulas to help make the calculations easy to understand, make important logical assumptions more transparent to users, and make it possible for an ITR/Tester to check the coding in the spreadsheet.

When development and verification is complete, the column containing the formula can be safely hidden, if desired, but not deleted to keep it as a record of what the spreadsheet is attempting to do.

Alternatively, spreadsheet auditing software packages are able to translate cell formulas into natural language formulas based on row and column headings in the spreadsheet. While potentially very powerful, caution needs to be exercised when using these translators since poor spreadsheet design and the use of subheadings can confuse the translators and make the translations incorrect.

Spreadsheet Professional includes options that enable the user to specify the location of row and column headings and can be able to overcome this limitation.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)

Use Named Ranges

Consider using range names when developing a spreadsheet; include a table of range names and their associated cells on the documentation worksheet of the spreadsheet.

Allocating meaningful range names to areas or cells within a spreadsheet can make the formulas in a spreadsheet easier to understand and reduce the risk of errors made by referring to the wrong cell. This is most useful when a cell is referenced in a formula a long way from where it is calculated, particularly when the input cell is contained on a different worksheet than the formula.

When referring to nearby cells use cell references rather than range names; referenced cells close to their point of reference are relatively easy to trace and understand in the formula. Avoiding range names will keep the formula definition shorter.

Build in Error Traps

Include checks for errors in the spreadsheet such as checking for errors in the input data provided or in the spreadsheet formulas.

Cross casting is the process of checking the totals in the rows and columns of a spreadsheet are consistent. This is useful for detecting errors introduced by the insertion of rows or columns.

Use Data Validation

Data validation is a process of ensuring a spreadsheet operates on clean, correct and useful input data. Data validation testing looks for too much data, too little data, no data, the wrong kind of data, the wrong size of data, and inconsistencies in the data and notifies the User of an error or closes the spreadsheet process without crashing or without allowing unauthorized access to the User's computer.

- The data validation feature of Excel®, available under Data>Validation, allows the user to specify valid data ranges and types and the message to display if inputs lie outside the specified range or data type.
- Conditional formatting is an alternate method for data validation and is used to identify when a result is outside an acceptable range by changing the color of the cell.
- Use of error trapping code in macros or scripts can be used for data validation by looking for exceptions resulting from out-of-range data values and handling the exception in a controlled manner, preventing the potential for exploitation of vulnerabilities in unhandled exceptions.

Macros

Macros can be used to automate repetitive tasks within a spreadsheet and can improve usability. However, care needs to be taken to ensure macros are verified as functioning correctly.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)**2.3 COMMON ERRORS IN SPREADSHEETS**

The following represent some common and known spreadsheet errors. It pays to be alert for these common errors during verification and use of the spreadsheet.

Formulas Not Copied

One of the easiest ways to introduce an error into a spreadsheet is to update the formula in a cell and forget to copy the new formula across into the other cells in a row. By using a formula map and a design in which all formulas are copied across all rows, this mistake can be noticed very quickly.

Referencing the Wrong Cell

Nearly every formula in a spreadsheet refers back to another input or calculation. With the quantity of references in any large spreadsheet, it is inevitable that you will make mistakes and refer to the wrong cell. Sometimes, the resulting formula will produce a meaningless result making it easy to spot with some simple numerical testing. If you are unlucky, the error in the result will be more subtle.

The only way to check for wrong references with any confidence is to check every unique formula in the spreadsheet. In Excel®, checking references is made easier if you use the auditing toolbar. This allows you to trace the precedent cells graphically.

Summing Over the Wrong Range

A similar mistake is to include the wrong cell reference in a SUM formula. It is particularly easy to introduce this error when you insert an extra row in a block of cells that are being summed. Insert a row in the middle of the block and the formula will automatically adjust to include the extra row, but insert a row immediately above or below the block, and the new row will be omitted from the formula.

You can build internal checks of the sums in a spreadsheet by using cross casting, as described in Attachment B. When testing a spreadsheet, trace precedents feature in Excel® helps find errors of this type by showing them graphically.

Confusion between Relative and Absolute References

Another commonly found error is caused by confusion between relative and absolute references. A cell reference in a formula of the form '=D4' will change if you copy the formula across the row to E4, F4 and so on. Copied down a column it will change to D5, D6, etc. If you use the reference '\$D\$4' it will not change when copied across or down. You can also use semi-absolute references of the form '\$D4 or D\$4.

The most common mistake is using a relative reference instead of an absolute one. This is quite easy to spot numerically, but if you use an absolute reference in place of a relative one it can be much more difficult to spot. Again, the only way to find this mistake reliably is to go through all of the unique formulas in the spreadsheet.

ATTACHMENT B – UCS STANDARDS AND PRACTICES (cont.)

Units Errors

Mixing up the appropriate units for the elements in a calculation is another frequently occurring problem. Finding such errors can be helped by features such as inclusion of a units column to make tracing through the calculation clearer. It is also good practice to try to avoid switching between units, except when absolutely necessary. Performing a dimensional analysis of formulas will ensure that errors in units are identified.

Misusing Functions

Certain functions such as the lookup and reference functions, VLOOKUP, HLOOKUP, INDEX, and MATCH are frequently used incorrectly. It is important to construct the VLOOKUP statement correctly to ensure that an exact rather than an approximate match to the desired value is obtained.

Complicated IF statements, especially nested IF statements, are also particularly prone to error. These functions are often very useful, but make sure that you understand exactly how they work before using them in your spreadsheet. By making the formulas in your spreadsheet easy to understand, you reduce the risk of introducing errors and increase the chances that a verifier will find your mistakes.

Spreadsheet auditing software is able to check for many function problems.

Corrupted Links to External Workbooks

Beware of making structural changes, such as inserting and deleting rows/columns, to linked workbooks. It is essential to have both workbooks open when making structural changes to a linked workbook to ensure the change is reflected in the workbook containing the link.

Hidden Cells/Pages

Hidden cells, pages, or ranges can cause difficulty both during the construction of a spreadsheet and during the verification. If you are developing or modifying a spreadsheet, hidden cells can result in the inclusion of more than is expected in a calculation, such as a SUM. Likewise, if you copy across hidden areas, you can overwrite formulas that are important to the spreadsheet.

Overwriting Data

It is possible to inadvertently overwrite one value with another when multiple worksheets are selected. The design of Excel is such that when multiple worksheets are selected and data are entered into one cell on the top worksheet, the data values will propagate to the same cell on other selected worksheets. The inadvertent overwriting of data entries can be reduced by protecting worksheets. However, the spreadsheet developer needs to consider the possibility of having an overlapping alignment of unlocked cells in separate worksheets when unlocking cells to permit data entry into protected worksheets. If the positions of the unlocked cells happen to coincide on several worksheets that are selected together, it is still possible to overwrite data on underlying worksheets. The spreadsheet needs to be designed and configured so data entry cells do not coincide and users need to be careful to only select one worksheet at a time.