Conceptual Validation Plan for OSPExplorerApp

SiaMirza

June 17, 2025

1. Introduction & Purpose

The `OSPExplorerApp` is a modular Shiny application designed for interactive exploration, visualization, and basic analysis of PBPK/QSP (Physiologically Based Pharmacokinetic/Quantitative Systems Pharmacology) data. It leverages an associated R package, `ospapputils`, for core data handling, validation, and advanced calculations.

The purpose of this document is to outline a conceptual validation plan for the `OSPExplorerApp` and `ospapputils` system. In a pharmaceutical research and development environment, any software used to generate, process, store, or transmit data that may be submitted to regulatory agencies (e.g., FDA) or supports critical scientific decisions must comply with **Good Practice (GxP)** principles. This validation plan ensures the system's fitness for its intended use, maintains data integrity, and adheres to relevant regulatory guidelines such as **21 CFR Part 11** (Electronic Records; Electronic Signatures) where applicable for data provenance.

2. Scope

This conceptual validation covers the `OSPExplorerApp` and its underlying `ospapputils` R package, encompassing their: * **Development:** Coding, modular design, version control. * **Testing:** Unit, integration, end-to-end (E2E), and user acceptance testing. * **Deployment:** Considerations for production environment setup (e.g., Shiny Server). * **Maintenance:** Future updates and re-validation.

Out of Scope: This plan does not cover the validation of the foundational R language, Shiny framework, or operating system, which are typically addressed through vendor qualification, foundational IT infrastructure validation, or established industry best practices.

3. Conceptual Roles & Responsibilities

For a robust validation, various stakeholders would be involved:

- **System Owner (e.g., Lead Scientist/Analyst):** Defines user requirements, approves validation deliverables, ensures system continues to meet business needs.
- **Business Owner (e.g., Department Head):** Provides overall business direction and resource allocation, ensures compliance with organizational policies.
- **Developer/IT (You):** Designs, develops, tests, and maintains the software; generates design specifications and test protocols.
- **Quality Assurance (QA):** Reviews and approves all validation documentation, ensures adherence to GxP and internal procedures, performs independent oversight.
- **Testers:** Execute test protocols (IQ, OQ, PQ) and document results.

4. System Description (High-Level)

The `OSPExplorerApp` system consists of:

- **Front-end:** A modular Shiny web application (`OSPExplorerApp/`) providing an interactive user interface for data upload, filtering, visualization (concentration-time plots, GOF plots), and raw data display.
- **Back-end/Utilities:** An R package (`ospapputils/`) containing core functions for mock data generation, robust data validation, and specific analytical calculations (e.g., goodness-of-fit metrics).
- **Version Control:** Git managed in a GitHub repository, ensuring full traceability of all code changes.
- **Environment Management:** `renv` is used for both `OSPExplorerApp` and `ospapputils` to manage and snapshot R package dependencies, ensuring reproducible environments.
- **Continuous Integration/Continuous Deployment (CI/CD):** GitHub Actions automate testing (`testthat` , `shinytest2`) and build processes, ensuring code quality and deployment readiness.

5. Validation Life Cycle Approach

The `OSPExplorerApp` is best categorized under **GAMP 5 as a Category 4 (Configured Product) or Category 5 (Custom Application)** software, requiring a risk-based approach to validation with significant testing. The validation lifecycle follows a phased approach:

1. **Planning:** Define the validation strategy, scope, roles, and responsibilities. (This document serves as a conceptual starting point).
2. **Specification:**
    - **User Requirements Specification (URS):** Defines what the user needs the system to do.
    - **Functional Specification (FS):** Describes *how* the system will meet the URS.
    - **Design Specification (DS):** Details the technical design and architecture (e.g., module breakdown, data flows).
3. **Configuration / Development:** Actual coding and system assembly.
4. **Testing:**
    - **Installation Qualification (IQ):** Verifies correct installation of the software and its environment (e.g., R, packages, Shiny Server).
    - **Operational Qualification (OQ):** Verifies that the system functions according to its functional specifications (e.g., all buttons work, plots render correctly, data validation rules are applied). This is where `shinytest2` E2E tests play a crucial role.
    - **Performance Qualification (PQ):** Verifies that the system performs as expected under real-world conditions and meets user acceptance criteria (User Acceptance Testing - UAT).
5. **Reporting:** Generation of the Validation Summary Report, compiling all evidence and confirming successful validation.
6. **Maintenance:** Ongoing monitoring, change control, and re-validation for significant changes.

6. Key Validation Deliverables

The following formal documents would typically be generated during the validation process:

- **Validation Plan:** (This document's full version)
- **User Requirements Specification (URS)**

- **Functional Specification (FS)**
- **Design Specification (DS)**
- **Installation Qualification (IQ) Protocol & Report**
- **Operational Qualification (OQ) Protocol & Report**
- **Performance Qualification (PQ) Protocol & Report (including UAT)**
- **Traceability Matrix**
- **Validation Summary Report**
- **User Manual/System Administrator Manual**

7. Testing Strategy

A multi-layered testing strategy ensures the reliability and accuracy of the `OSPExplorerApp`:

- **Unit Testing (`testthat` for `ospapputils`):** Verification of individual functions and components within the `ospapputils` package (e.g., `load_osp_data`, `validate_osp_data`). Ensures foundational logic is sound.
- **Integration Testing:** Testing the interactions between different modules within the `OSPExplorerApp` and between the Shiny app and the `ospapputils` package.
- **End-to-End (E2E) Testing (`shinytest2`):** Automated testing of full user workflows, simulating user interactions with the UI (e.g., uploading data, applying filters, verifying plot rendering, validating messages). This forms a significant part of the OQ.
- **User Acceptance Testing (UAT):** Performed by actual end-users to confirm the application meets their specific business needs and requirements, as defined in the URS. This is the core of the PQ.
- **Test Data Management:** Use of predefined, documented, and traceable test data sets (including positive and negative test cases) for all qualification phases.

8. Traceability Matrix (Conceptual)

A Traceability Matrix is essential for demonstrating that all user requirements are addressed by system functionality, implemented in the design, and thoroughly tested.

User Requirement (URS)	Functional Spec (FS)	Design Spec (DS)	Test Case (OQ/PQ)	Code Module/Function
URS-001: App shall allow upload of OSP data (CSV/Excel).	**FS-001:** File upload shall accept .csv and .xlsx.	**DS-001:** `fileInput` in `mod_data_loader_ui`; `observeEvent` for file type handling; `readr` / `readxl` for parsing.	**OQ-001:** Verify successful upload of valid .csv and .xlsx files.	`mod_data_loader.R`
URS-002: App shall validate uploaded data structure and content.	**FS-002:** Implement checks for required columns, data types, and value ranges.	**DS-002:** `ospapputils::validate_osp_data` function.	**OQ-002:** Test validation with valid, missing column, wrong type data.	`ospapputils/R/data_validation.R`
URS-003: App shall display concentration-time profiles interactively.	**FS-003:** Concentration-time plot with filtering, log scale, points/lines toggle.	**DS-003:** `plotlyOutput` in `mod_visualization_ui`; `ggplotly` in `mod_visualization_server`.	**OQ-003:** Verify plot rendering, filter application, axis scaling.	`mod_visualization.R`
URS-004: App shall show goodness-of-fit metrics.	**FS-004:** GOF calculations (e.g., MAPE, R-squared) displayed for selected data.	**DS-004:** `ospapputils::calculate_gof`; `mod_comparison.R`.	**OQ-004:** Test GOF calculation with known data.	`ospapputils/R/gof_metrics.R`, `mod_comparison.R`

9. Conceptual Risk Assessment

A systematic risk assessment identifies potential failures and their impact, guiding validation efforts.

Risk Category	Identified Risk (Example)	Potential Impact (High/Medium/Low)	Mitigation Strategy
Data Integrity	Incorrect parsing/loading of uploaded data.	High	Robust `readr` / `readxl` usage; comprehensive `ospapputils::validate_osp_data`; `tryCatch` blocks.
	Invalid or out-of-range data used in calculations/visualizations.	High	Strict data validation rules (`ospapputils::validate_osp_data`); clear error messages; data cleansing/transformation steps.
System Reliability	App crashes due to large datasets or complex operations.	Medium	Performance testing (PQ); efficient R code; memory management best practices; error handling.
	Inaccurate calculations/ visualizations.	High	Unit tests (`testthat`) for `ospapputils`; `shinytest2` for E2E visualization verification; peer code review.
Compliance	Lack of reproducible environment for analyses.	High	Use of `renv` for strict dependency management and environment snapshotting.
	Inadequate version control of code.	High	Strict adherence to Git for all code changes; clear commit messages.
	Insufficient documentation of development/testing.	Medium	Comprehensive `README`s; in-code comments (`roxygen2`); validation documentation (URS, FS, DS, IQ, OQ, PQ).
Security (Basic)	Unauthorized access to the application or data.	Medium (if deployed)	Implementation of authentication/authorization if deployed on a Shiny Server; restricted server access.
	Malicious file upload.	Low	Input validation (`fileInput` accepts only specific types); server-side scanning (if enterprise deployment).

Disclaimer: This is a conceptual outline for discussion purposes. A full validation plan would be a formal, controlled document with much greater detail.