Public



Building Apps Made Easy: From R Shiny to Organizational Impact

Yasutaka Moriguchi, GSK
Santhosh N, GSK

Public

Disclaimer

- *The views and opinions expressed in this presentation are those of the author(s) and do not necessarily reflect the official policy or position of GSK.*



Public

The real challenge is not creating the app

Coding barrier ↓

Adoption ≠ Guaranteed

Sustainable use ↑

- Generative AI + Open-Source Technology make prototyping dramatically easier.
- But an organizational tool must also be understandable, testable, supportable and replaceable.
- Today: how far should we go for non-GxP applications used in our business?

Public

3 Application classes

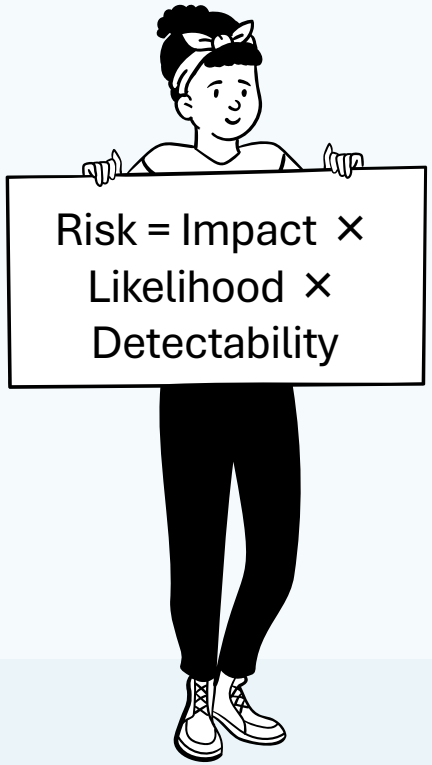
Class	Example	Scope	Approach
1. Study-specific / Small group use / Non-GxP	Prohibited concomitant medication viewer for Study-001	One study / Small group use	Simple validation / QC
2. Cross-study support / Organizational use / Non-GxP	CDISC QC, data exploration, utility tools	All studies / Organizational use	More testing + UAT
3. GxP / IT-managed	Production analysis systems	All studies	GxP + CSV + IT governance - OUT OF SCOPE

Key principle: validation effort should follow intended use + risk, not the fact that it is written in R.

Public

Risk-based question: “What can go wrong?”

- Could a wrong result change a clinical conclusion or a business decision?
- Is the application the authoritative source, or only a convenience / support tool?
- Will the result be independently verified before it is used?
- How broad is the impact: one study, every study or your organization?
- How easy is it to detect and recover from an error?

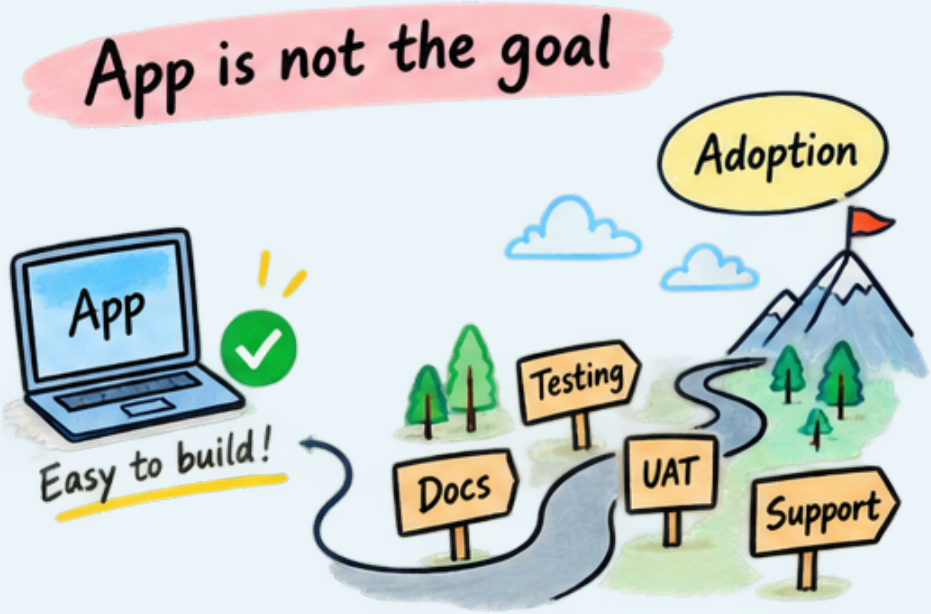


Public

Recommended lifecycle for R Shiny applications

1. Define intended use
2. Risk assessment
3. Requirement
4. Develop
5. Test
6. UAT
7. Release
8. Maintain, Support

- Ensure the evidence is proportionate.
- Enough to show that the intended use works.
- Not a miniature GxP validation package.



Public

Testing vocabulary: Software testing vs. CSV

Activity	Typical target	Main question
Unit Test	Individual function/module	Does each building block behave correctly?
Integration Test	Interfaces between modules	Do components work together?
System Test	Whole application	Does the complete system meet requirements?
Acceptance Test / UAT	User/business acceptance	Can intended users perform real tasks and accept the result?
CSV (GxP)	Regulated computerized system	Documented evidence that a system is fit for intended use in its regulated context

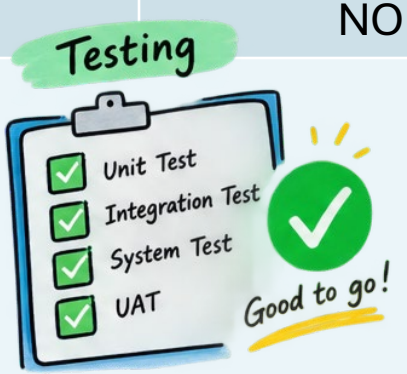
For non-GxP tools, software testing + documented user acceptance can be sufficient.
CSV terminology is not automatically required.

Public

Test strategy: What should we test?

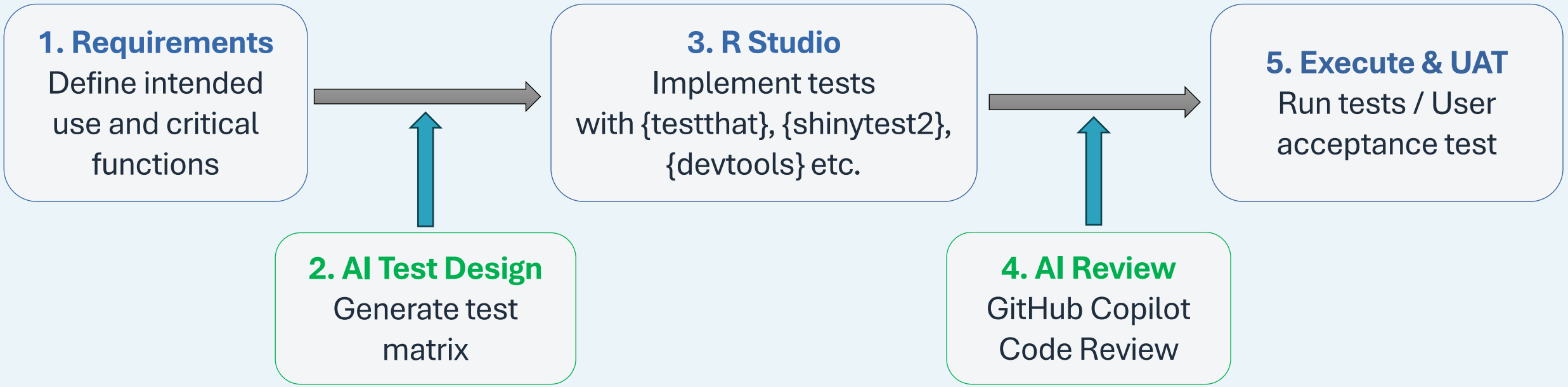
Test	Focus	Class 1	Class 2
Functional / Unit	Core calculations, filters, transformations	YES	YES
Integration	Module-to-module interfaces	Usually targeted	Targeted
System	End-to-end user workflow	YES, lightweight	YES
Acceptance / UAT	Real user scenarios	YES	YES
Regression	Changes do not break existing functions	As needed	As needed
Performance	Load/stress/response under many users	As needed*	As needed*
Security	Penetration / threat testing	Usually NO	Risk-based
Cross-browser / OS	Different platforms	Usually NO	Usually NO
Monkey / random testing	Unexpected arbitrary interactions	NO	NO

- For example: If the application loads CDISC datasets with more than 10 GB.



Public

Test Workflow: GitHub + R Studio + AI



AI is useful for

- Generating many plausible test scenarios
- Finding boundary / invalid-input cases
- Explaining unfamiliar code and test failures

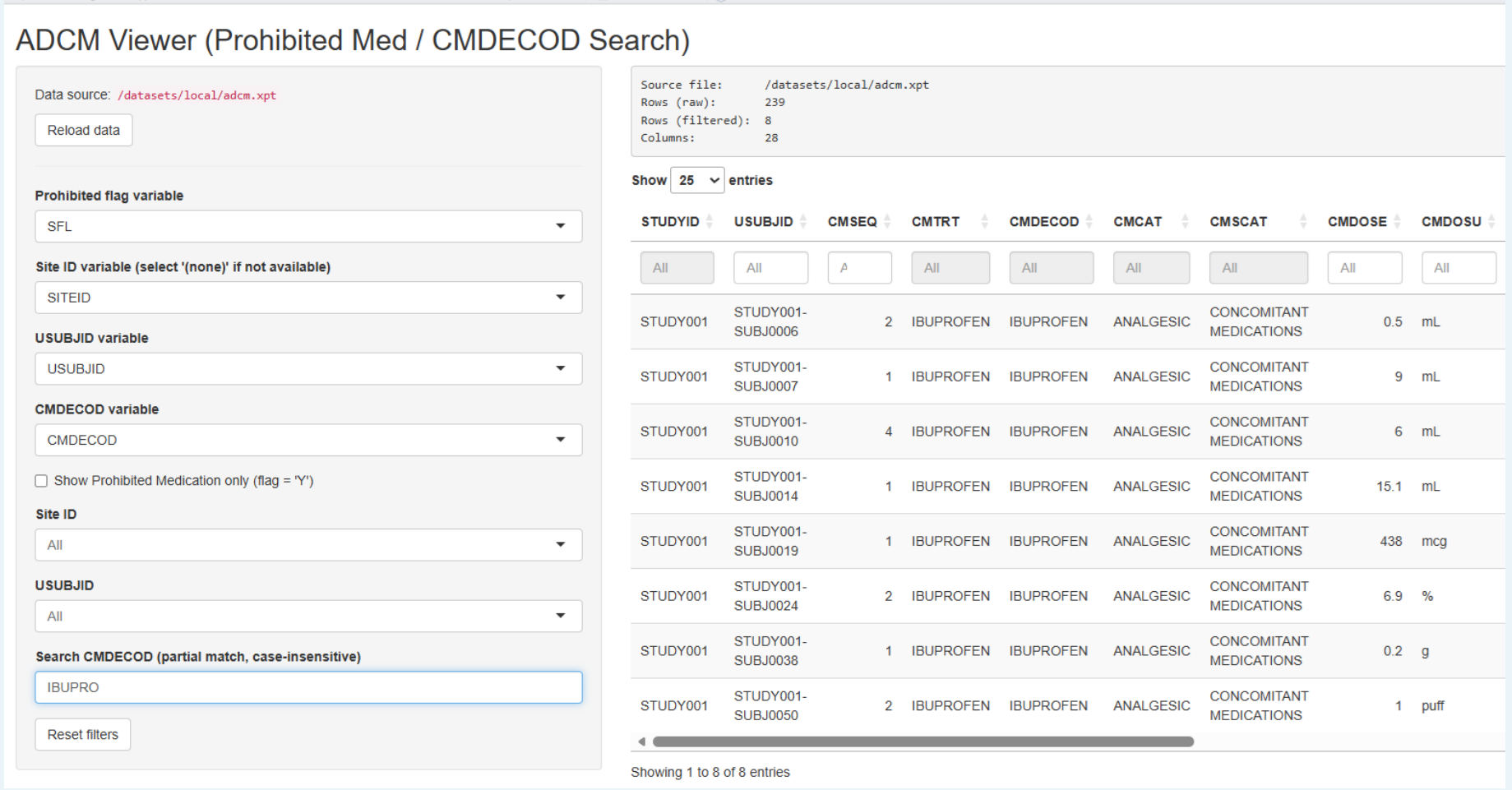
Human responsibility

- Define what “correct” means
- Independently verify important expected results
- Approve test evidence and release decision

Public

Class 1: Study-specific / Small group use

Class 1 application example: Data viewer for prohibited concomitant medications



Public

Example of Class 1: Data viewer for prohibited concomitant medications

- Example: Data viewer for prohibited concomitant medications.
  - Compared to Excel, the advantages of R Shiny Application are: 1. Ability to manage access permissions, 2. No possibility of data modification or manipulation, 3. Once deployed, only data updates are required.
- Formal identification is performed elsewhere (e.g., double programming, data review meeting).
- Therefore, the Shiny app is a viewing and navigation aid, not the decision engine.
- Minimum evidence should demonstrate: correct data load, correct display, filtering/search, key labels, missing values, and obvious error handling.

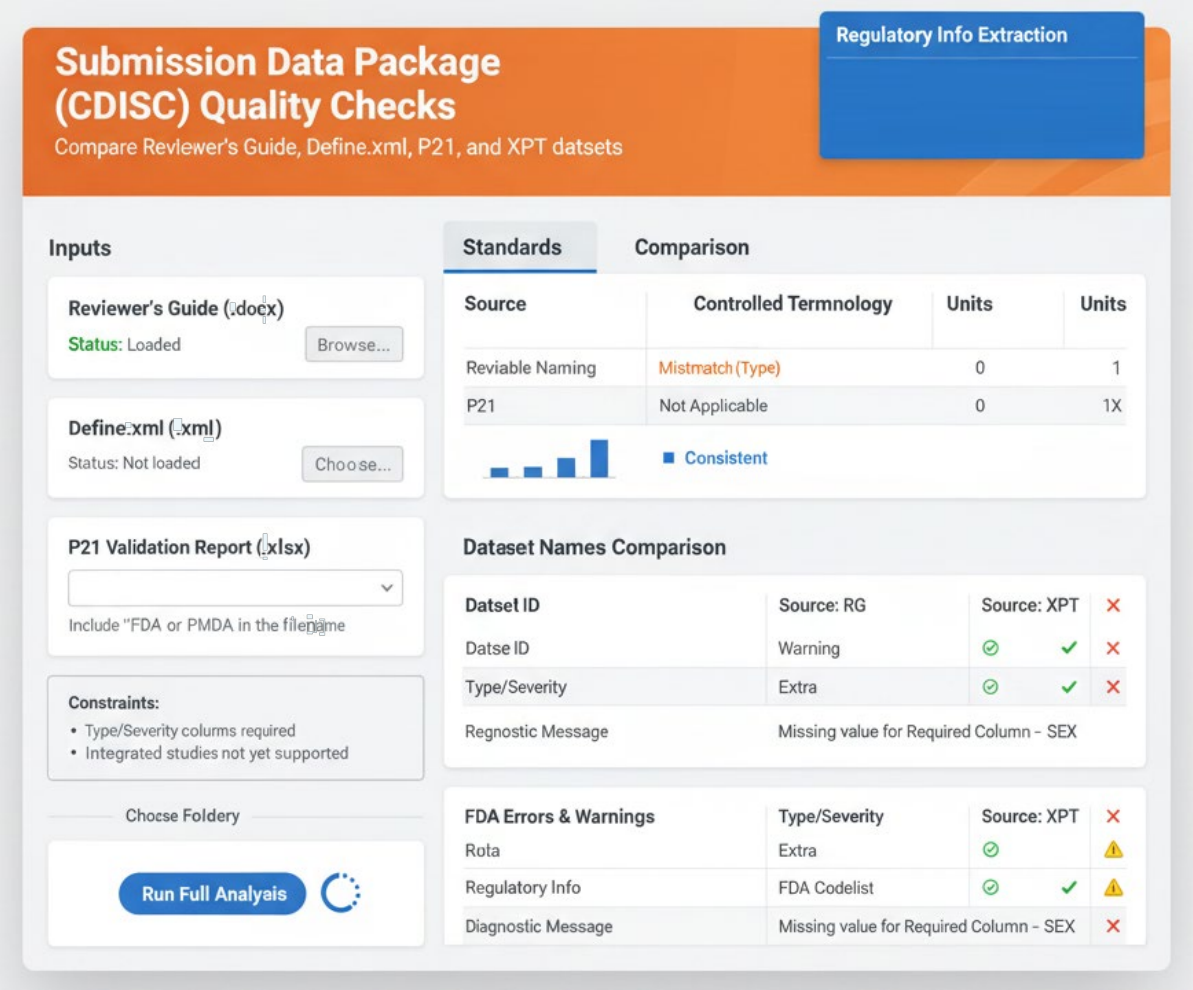


Suggested evidence: Developer test checklist + Small set of expected-result test cases + User confirmation

Public

Class 2: Cross-study / Organizational use

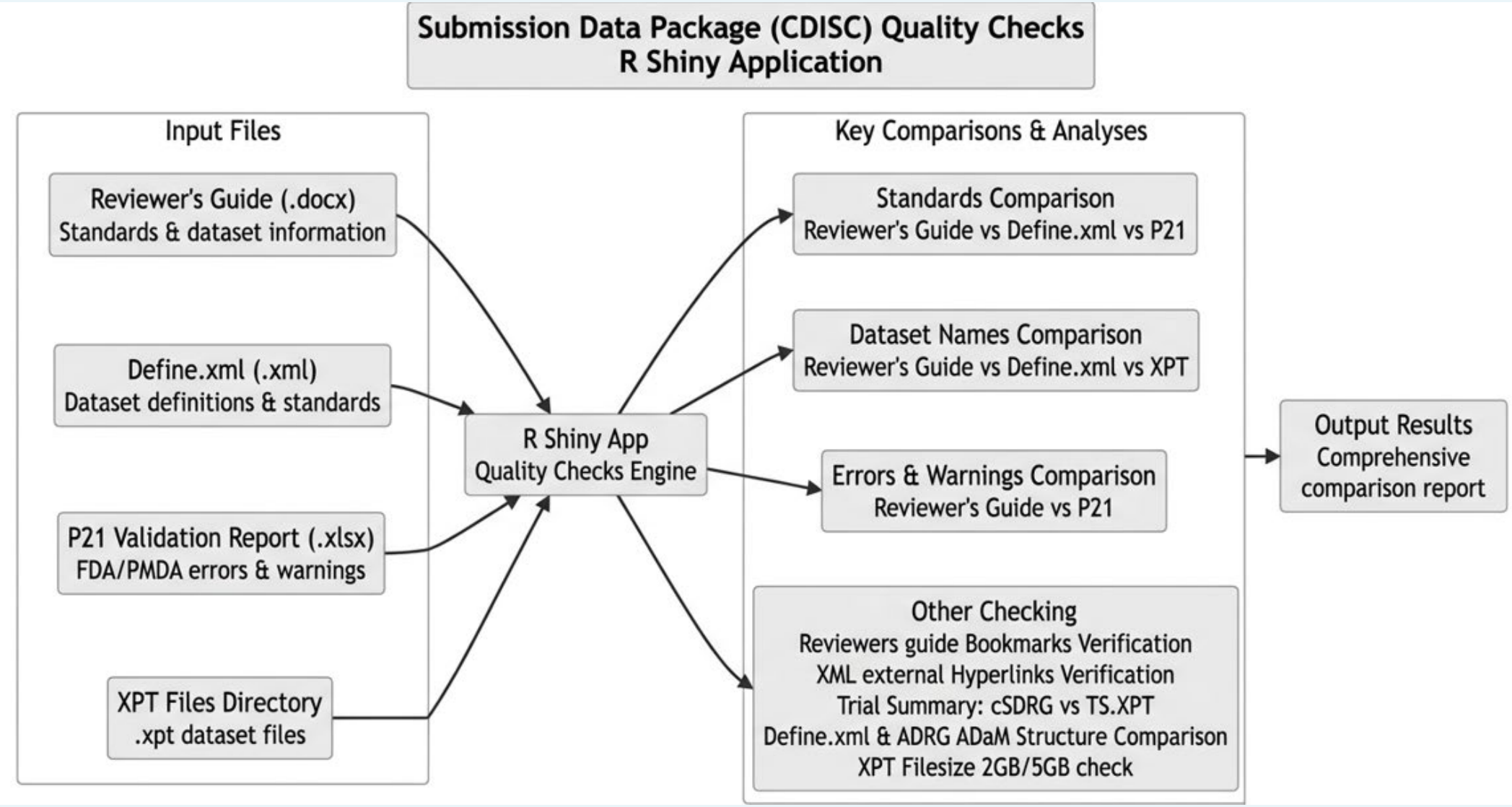
Class 2 application Example: CDISC package QC application used across multiple clinical studies.



This is just sample image.

Public

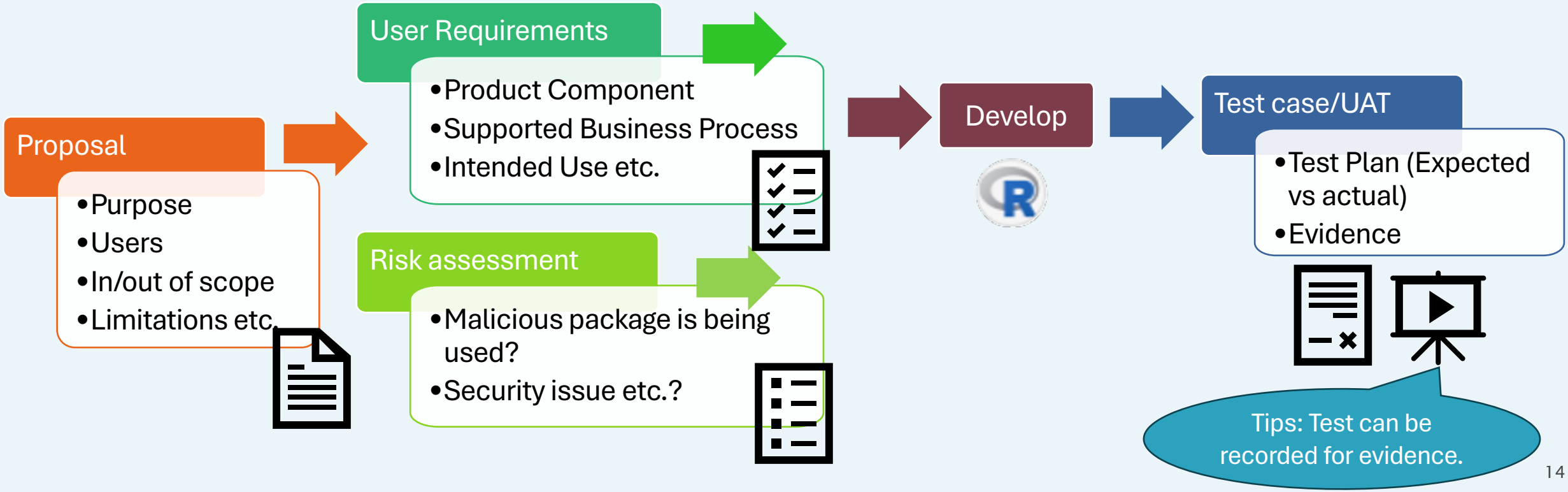
Mermaid diagram: CDISC package QC application



Public

Example of Class 2: CDISC package QC

- Example: CDISC package QC application used across multiple clinical studies.
- As one defect can affect many studies, test depth should increase even when the app is not GxP.
- Formal QC remains outside the app; the app should clearly state that it is a support tool.



Public

Impact and Lessons Learned

- Impact:
  - CDISC packages review time saving (e.g., 20 hours manual review/QC per study) / Accelerate Timelines / Boost Quality & Consistency
- Lessons Learned:
  - Processing and extracting .docx (ADRG/cSDRG) are not straight forward as its unstructured data.
  - Utilizing AI for this unstructured data might yield better results.
  - During the Errors & Warnings comparison, there are the issue of Rule ID not being in many studies because it is not included in the PHUSE template.



Public

The hidden risk: “The developer is the support model”

- Applications often start as personal productivity tools.
- Then they become widely used while knowledge remains in one person's head.
- Risks: staff turnover, package changes, expired credentials, undocumented business rules, no one able to rebuild the app.
- Organizational impact therefore depends on ownership, not just code quality.

Bus factor = 1 → Operational risk

The “Bus factor” is a measurement of the risk resulting from information and capabilities not being shared among team members, derived from the phrase "in case they get hit by a bus". “Bus factor = 1” means one person holds critical knowledge; the project is essentially failed if he/she disappears.

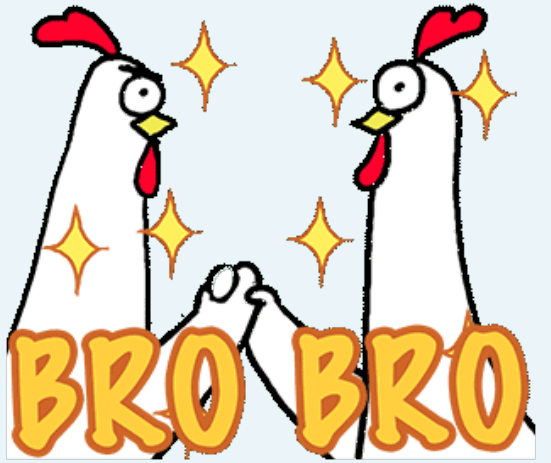
Public

A pragmatic support model for Class 1–2

Role	Responsibility	Minimum
Business owner	Defines intended use, priority, acceptance	Required
Technical owner	Code, dependencies, release, troubleshooting	Required
Backup owner	Can maintain/release without the primary owner	Strongly recommended
IT / platform contact	Hosting, access, infrastructure escalation	As needed
Users	Feedback, incidents, UAT participation	Required

Class 1: Business owner and Technical Owner may be a same person, but archive code + test evidence + handover notes for backup owner. IT contact may not be needed.

Class 2: Establish a small “application owner” community / registry and mandatory backup.



Public

Key Take-Home Messages



Easy development changes the bottleneck from coding to governance and adoption.



Define intended use and risk first; then choose the minimum effective test set.



Class 1 can be lightweight. Class 2 needs broader testing and UAT.



Document enough to reproduce, test and hand over the application.



Ownership, support and communication make the application sustainable.

Public

Thank you for your attention!



Public

What about DQ / IQ / OQ / PQ?

Term	Typical question	Context
DQ: Design Qualification	Does the design meet predefined requirements?	CSV / validation context
IQ: Installation Qualification	Was the system installed correctly?	CSV / infrastructure context
OQ: Operational Qualification	Does it operate as specified?	CSV / regulated context
PQ: Performance Qualification	Does it perform in the real process?	CSV / regulated context
Non-GxP Shiny	Software testing + UAT + appropriate records	Use proportionate controls

For this presentation, Class 1 and 2 are non-GxP applications: formal DQ/IQ/OQ/PQ is not the objective.