roxyreqs

Adding roxygen2-style documentation to testthat

Moritz Lang (moritz.lang@roche.com)
Roche

Mario Annau
Roche

Doug Kelkhoff
Domino Data Lab

Szymon Maksymiuk
Roche

useR! 2026 - Lightning Talk

The gap

roxygen2 documents functions, not tests

- *Functions get structured tags:* `@param`, `@description`, `@author`
- *A test gets one free-text string:*

```
1 test_that("parse_input handles edge cases", { ... })
```

The rest lives elsewhere:

- *Who authored or reviewed a test*
- *Which requirement it covers*
- *Recorded in commit messages, spreadsheets, or nowhere*

The idea

`@meta` tags above `test_that()`

```
#' @meta author Alice
#' @meta reviewer Bob
#' @meta review_date 2025-01-15
#' @meta requirement REQ-204
#' @meta description Validates input parsing.
test_that("parse_input handles edge cases", {
  expect_equal(parse_input("a,b"), c("a", "b"))
})
```

- *Parsed by roxygen2's own tag machinery*
- *Keys are free-form: add whatever your project needs*

→ Increasingly required in regulated fields (pharma, finance) where validation demands documented author/reviewer roles and requirement-to-test traceability

From tags to reports

A JUnit reporter exports the metadata

```r
testthat::test_local(".", reporter = roxyreqs::JunitReporterMeta)
```

```xml
<testsuite name="main" tests="23" skipped="2" failures="1">
  ...
  <testcase classname="main" name="parse_input_handles_unicode">
    <properties>
      <property name="author" value="Alice"/>
      <property name="reviewer" value="Dan"/>
      <property name="requirement" value="REQ-206"/>
    </properties>
    <failure message="length(parse_input(...)) not equal to 2. 1 != 2"/>
  </testcase>
  ...
</testsuite>
```

- *Test result and metadata, side by side in one report*
- *Standard JUnit XML, the R pendant to pytest's test metadata*

The same `@meta` tags work on functions

```r
#' @meta id REQ-204
#' @meta risk high
parse_input <- function(x) { ... }
```

- *Functions declare the requirement and its risk*
- *Share an `id` convention to link tests back to them*
- *Requirement, risk, and test, all captured in the code*

A check enforces the metadata in CI

```
roxyreqs::check_meta(print_error = TRUE)
```

- *Every test and function carries the required tags and unique `Id`s*
- *Missing or incorrect metadata fails the pipeline*
- *Documentation and code can never drift apart*

Why it matters

Traceability, generated from `@meta` tags

Requirement	*Risk*	*Test*	*Author*	*Reviewer*	*Status*
REQ-204	*high*	*parse_input edge cases*	*Alice*	*Bob*	✅
REQ-205	*low*	*parse_input empty input*	*Carol*	*Bob*	✅
REQ-206	*medium*	*parse_input unicode*	*Alice*	*Dan*	❌

- *Requirement and risk from the function tags*
- *Author and reviewer from the test tags*
- *Pass/fail from the test run*

github.com/mnlang/roxyreqs