LDD Testing Techniques

Jesse Stone, PDS Small Bodies Node

Recap of our goals with testing

- Ensure that the dictionary does what we want it to do
- Ensure that the data that is required is actually captured
- Ensure that the schematron rules work
- Prevent regressions
- Detect non-backwards compatible changes

What should we keep in mind while writing tests?

- Tests should be maintainable and understandable
- Tests should be documented and well organized
- Tests should provide good coverage
- Tests should communicate the right amount of information

Principles

- Uniform labels
- Granular or Monolithic tests?
- Good test coverage
- Minimizing Redundancy
- Documentation
- Organization

Keeping labels uniform

- Unnecessary variations in the label will make it more difficult to track down errors.
- Sometimes variations are necessary when a discipline area can apply to different data types

Monolithic tests vs granular tests

- Two possible styles of invalid label test are monolithic tests and granular tests
- Monolithic tests have few labels with many introduced errors per test
- Granular tests have many labels with few introduved errors per test

Keeping tests granular

- Each label is invalid in only one way
- This simplified interpreting the results, both for you and for the testing framework.

Drawbacks to granular tests

- Granular tests will increase the number of labels that the LDD is tested against
- Each of these labels will need to be maintained individually

Demonstration - Survey Dictionary Tests

https://github.com/pds-data-dictionaries/ldd-survey/tree/main/test

Objectives

Demonstrate granular tests

The tests in the survey dictionary each have one thing wrong with them,

This is enough the trip the validator. When a label fails, it's for a single reason.

ldd-survey

Producing Test Labels

- Hand writing labels
  - Labels are just XML files, and can be written in any text editor.
- Injecting discipline area fragments into label templates
  - In addition to making the labels easier to generate, the parts of the label that are being tested are separated from the rest of the label.
- Mutating existing labels
  - Keep a mapping of *XPaths* and operations to perform on a location

Demonstration - LDD Test Generator

https://github.com/sbn-psi/ldd_utilities/tree/master/LddTestGenerator

Objectives

Demonstrate a template/mutation-based approach to generating test labels

Mention how the framework could be expanded to mutate test files

generator

Monolithic tests

- Multiple tests can be packed into a single label
- Document each point where the test is expected to fail
- Examine the output of the test run to determine if there are any missed failures

Demonstration - Spectral Dictionary Tests

https://github.com/pds-data-dictionaries/ldd-spectral/tree/main/test

Objectives

Demonstrate monolithic tests

The tests in the spectral dictionary have more than than one error introduced.

The errors are documented within the file.

This reduces the number of tests that need to be written.

You will need to check the output yourself occasionally to verify the overall test suite result.

ldd-spectral

Interpreting the test output for monolithic tests

- Monolithic tests require more interpretation that granular tests.
- The testing system could miss labels that fail for the wrong reason
- Log files are built with each push, so check in there for more information

How many tests?

- *Any tests are better than no tests*.
- There are a lot of things that you can test for your dictionary.
- Some tests are more valuable than others
- Which tests are the most important to write?

The case against too many tests

- Too many tests can cause problems (This does *not* mean don't write tests)
  - The biggest problem with too many tests is that they need to be maintained
  - Maintenance can be necessary when either your dictionary changes, or when the dependencies change (IM changes, upstream dictionaries, etc)
  - A test should have its own job – it shouldn't just functionally duplicate another test
    - More simply, keeping the same structure, but switching between valid values is not a high-value activity.

Testing classes

- You could write many tests to use each combination of classes, but this is not necessarily valuable.
- It *is* more valuable to test the minimal description that you can include in each public class.
- It is better to write tests specifically for this, so that other tests still have only one job.

Exercising Schematron rules

- An invalid label test could fail on Schematron rules.
- Valid label tests could pass on Schematron rule, or not trigger the rule at all
- Exercising schematron rules is valuable, since they represent exceptions to the exisitng system.

Demonstration - Nuclear Spectroscopy Dictionary Tests

https://github.com/pds-data-dictionaries/ldd-nucspec/tree/main/test

Objectives

Demonstrate tests for each Schematron rule

Each Schematron rule in the Nuclear Spectroscopy dictionary has a corresponding test that fails the rule.

- Additional tests could be written to illustrate cases that pass each rule.
- Cases could also be written to illustrate cases where the rule does not apply.

There are multiple passing labels, which collectively exercise a variety of classes within the dictionary.

ldd-nucspec

Document the tests - What to document and why

- Each test should somehow document what is being tested.
- This will remind you how each test is expected to fail, or what each test is intended to exercise.
- If writing a monolithic test, this can be further developed into the expected output for comparison in a future version of the EN testing tool.

Document the tests - Where documentation goes

- Documentation can be as simple as a file that lists the test name and what it is testing.
- Documentation can also be written inline. It would be valuable to note precisely which line should fail.

Organize the tests

- A simple way to organize tests is by sorting them into valid and invalid label tests.
  - Although this is embedded in the name, sorting them will make it easier to find the test that you need, especially as the number of tests grows.
- If there are a large number of schematron rules, a good alternative would be to sort tests by schematron rule.

Access this presentation

HTML

https://sbn-psi.github.io/dmsp/LDDTesting/LDDTestingTechniques

HTML

PPT

https://github.com/sbn-psi/dmsp/raw/main/LDDTesting/stone-LDDTestingTechniques.pptx

PPT

PDF

https://github.com/sbn-psi/dmsp/raw/main/LDDTesting/stone-LDDTestingTechniques.pdf

PDF