Opportunities for Improving Anuket Reference Conformance (RC) Work Stream Transparency

This wiki page is based on an email sent to the Anuket TSC on Sept 6, 2021.

Introduction

An important objective of the release process is to make the development status of deliverables visible to the TSC, and the community. The following text discusses some ways to improve RC transparency.

RC Mapping to RA

RC is frequently described as being dependent on RA. However, during the Lakelse release, it became apparent that RC is, at best, loosely coupled to RA.

Milestone 2 (M2) in the release process requires a description of the coverage of RA by RC. However, RC does not provide a clear mapping of test cases to RA, making it unclear what the coverage is, or how the coverage has changed from release to release, or how RC has changed in response to changes in RA.

Not only does this obfuscate coverage, but it also contradicts the requirements set forth by the RC documentation, itself. For example: "*The conformance specifications **must** provide the **mapping** between tests and requirements to demonstrate **traceability and coverage**.*"

In addition, the lack of mapping creates confusion for RI development, since RI depends on both RA and RC. For example, if RC includes test cases not explicitly mapped to RA, then RI may spend time on test cases that are not actually required.

In order to meet the requirements documented in RC, I suggest that the RC work stream team create and publish a simple table that displays RA requirement references in one column, and corresponding test cases in another. This table should be updated for every release.

Test Case Information

The RC documentation includes lists of test case names. However, there are no links to detailed information about the test cases.

For example, a requirement author, or other stakeholder, might want to understand how a requirement is being tested. As another example, when the engineering team conducts a review of RA documentation for each release, they occasionally want to see how a requirement is being tested.

This issue could be corrected by adding links to the test case names that would take readers to expanded information about the test case. In addition, test cases should include reference numbers to facilitate mapping to RA, and to help differentiate similar sounding test cases.

RC Validation Testing

RC is currently lacking a validation plan, per the requirements of Milestone 1 (M1) of the release plan. This means that the TSC, and the community at large, have no documented way to understand how one of the project's most important release artifacts is being validated.

During the technical discussion meeting on Sept 6, we discussed validation testing. Based on that discussion, my understanding is that RC is validated by simply running the test cases on known good infrastructure and verifying that there are no failures. Unfortunately, this leaves out a key aspect of validating testing, which is confirming that the tests also fail when they should. In addition, the validation is only run on one environment, which means that over time, test cases become optimized to that environment, potentially leading to inaccurate results on different, yet still valid, infrastructure.

Validation testing could be improved by inserting faults and verifying that test cases fail appropriately. It would also be helpful to run the tests in multiple environments to avoid optimizing tests on a single environment.

Test Results

Although the test cases are run in a continuous integration (CI) environment, the community does not have readily available access to the results. The TSC, or a community member, should be able to independently see the results and determine the status of RC testing.

This issue could be resolved by creating a publicly available dashboard of RC testing results, and by providing regular status updates or reports to the TSC.