ISIS Test Data Reduction. K. Rodriguez[1], K. D. Lee[1], A. C. Paquette[1], A. R. Sanders[1] and J. R. Laura[1], [1]USGS Astrogeology 2255 N. Gemini Dr. Flagstaff, AZ 86001; krodriguez@usgs.gov.

Introduction: The Integrated System for Imagers and Spectrometers (ISIS) uses continuous integration testing with a proprietary test suite to monitor the overall quality of the code base and avoid code regressions. The current approach to application (e.g., cam2map or spiceinit) testing requires a large quantity of test data. The ~72Gb of test data requires complex synchronization with the ISIS source code and poses a large developer burden due to its size. Additionally, working with ISIS test data increases developer time performing tertiary operations and increases the probability of an error when making changes to ISIS. Ineffective or redundant tests against this large dataset increases test run time with the current turnaround time for tests requiring approximately 2.5 hours on an in-house Kubernetes cluster. Data volume and synchronization requirements are the largest burdens for enabling continuous integration testing, enabling continuous deployment and supporting community contributions, therefore, reducing the velocity at which improvements can be contributed to ISIS. By adopting a well documented testing framework (GTest), outside contributors will not have to learn a proprietary system to contribute changes. Moving on-disk test data over to minimal representative data co-located in the USGS ISIS repository will eliminate the code-data synchronization problem and reduce computation time for tests. Herein, we describe the results of an effort to reduce the overall data volume while adopting a standard testing framework.

Approach: Large data volumes make running ISIS tests a challenge for outside users, prevent traditional source control methods, and create a large burden for developers. Most ISIS application tests are run using proprietary tools written using the Make programming language and uses diff tools as comparators against "truthData", files in the ISIS test data store that represent expected output. This requires test data to be written onto a disk drive or another form of non-volatile memory as output from another program. The dependence on diff tools in place of in-memory assertions written with native code means data needs to be serialized using OS variant system calls which may cause changes in test outputs often requiring duplicate data for OS specific comparisons (e.g. one copy of the test data for MacOS 10.9 and another copy for MacOS 10.11) creating additional developer burden of maintaining redundant data. ISIS originally had ~72 gigabytes of this test data, far above the limits for common source control hosting services such as GitHub. Therefore, ISIS testing data are located in a file structure parallel to the source code directory structure. These test data are available to users through our Rsync server (isisdist.astrogeology.usgs.gov) that also hosts data required to run ISIS commands (SPICE kernels, DEMs, etc.). However, locally storing 73 Gb is a high barrier of entry for potential ISIS contributors.

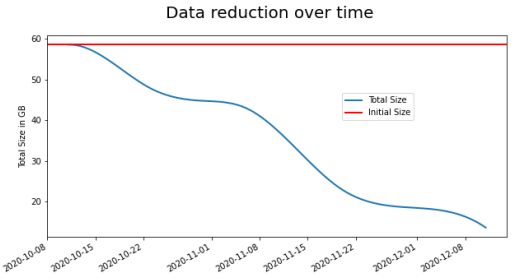
Some tests also need to be reconsidered as they test parts of ISIS functionality already tested elsewhere is the ISIS test suite or tested as part of a third party software suite (e.g. testing Boost library functionality already tested by the Boost maintainers) compounding redundancy. USGS developers have begun to replace the existing test automation tools with a combination of the GTest suite for writing tests in native code using in memory assertions and Jenkins for pipeline automation. The process requires altering ISIS applications that need tests converted to (1) be a callable function to be distributed with the ISIS library and (2) be called as part of the test. We leverage GTest fixtures in order to create representative datasets that can be reused in multiple tests. Therefore, the high level test conversion workflow:

1. Convert an ISIS application from code in a Main function, over to a callable functions with parameters as input params;
2. Ensure previous tests still pass after callable conversion;
3. Identify what tests need to be converted and which are redundant and can be deleted without reducing coverage;
4. Identify what representative data can be reused or created procedurally to meet testing needs;
5. Rewrite Makefile tests in GTest;
6. Delete old tests from repo;
7. Submit a pull request for review to the ISIS repository.

Some disruptions to this workflow include tests that require novel fixtures (e.g. application test requires image with specific binary metadata, how do we best solve this problem while reusing fixtures as much as possible) which were discussed within the team as needed. Applications were then sorted by highest to lowest test data volumes and extracted.

Results: After an initial adjustment period, each developer was averaging one full iteration of the test conversion workflow a day. The development team took a greedy approach, converting tests with the highest test data sizes first. Some preliminary work,

with 58.5 Gb of test data remaining in November, focused on applications with multiple Gbs of data for tests before quickly falling to <1-2 Gbs per application. The 1-2 Gb test data size plus the developers' capacity to iterate through the workflow about once per day, meant a predominantly linear reduction in test data over the length of the 9 week sprint. In total, the ASC team converted ~40 applications to the new GTest format reducing the total test data volume from 72.1 Gbs to 9.7 Gbs, a total reduction of 72% of the initial size.



Future Work: Orphaned test data is still archived and flagged for removal. The next step is to upload the remaining 9.7 gigabytes to an Amazon AWS service with appropriate metadata which will be publicly available for users to download as part of deprecating the RSync server favor of a USGS approved mechanism for data release. Although the top 40 apps consisted of 72% of the total data area, ISIS has more than 300 applications that still need to be converted. These applications need to be prioritized considering objectives secondary to minimizing files sizes, such as reducing test run times and complexity. This work is being considered in coming years.