Software Best Practices at Crustal Dynamics Data Information System (CDDIS): Steps to Consider

Justine Woo (Justine.y.Woo@nasa.gov)
Sigma Space Corporation
Lanham, MD, USA
Patrick Michael (Patrick.Michael@nasa.gov)
NASA Goddard Space Flight Center
Greenbelt, MD, USA
Carey Noll (Carey.Noll@nasa.gov)
NASA Goddard Space Flight Center
Greenbelt, MD, USA
Rebecca Limbacher (Rebecca.i.Limbacher@nasa.gov)
Science Systems and Applications, Inc.
Lanham, MD, USA

The Crustal Dynamics Data Information System (CDDIS) has been incrementally transitioning to a new software system with the goals of increasing automation and quality control measures. In doing so, software engineering best practices were identified and implemented alongside the new system to ensure the integrity and sustainability of the system. Many of these best practices are applicable to different systems and this poster introduces them, how they have been implemented at CDDIS, the benefits CDDIS has reaped, and how they may be applicable throughout other science related systems.

Definitions

- Code Base: the code/software package
- Regression Test: code created to verify changes made to the Code Base (enhancements, bug fixes) still performs correctly
- Repository: a storage location for the Code Base and Regression Testing Code so that it can be retrieved and installed on another computer; it also allows for version control
- Error Database: Database or tables in a database dedicated to recording bugs/issues; necessary if running items on crontab

Steps:

1) The code base is created and outputs are correct
2) Create regression tests; there are several ways to do this. Most major languages have packages that you can download to aid you in this or it can be as simple as creating a key file with the outputs expected and checking the test results against the key file.
3) Once the code base and regression test are complete, save them to a repository (preferably on another system)
4) Create documentation on how to compile/build the code, the version of the compiler and OS, and the exact commands used to run the program
5) If the code is run on crontab, a database with error tables needs to be created so that any errors encountered are recorded and easy to access and find.
6) Whenever changes are made to the code base, the code needs to be tested through regression testing before being pushed to production and uploaded to the repository. Update the documentation as needed.

Implementation and Benefits at CDDIS

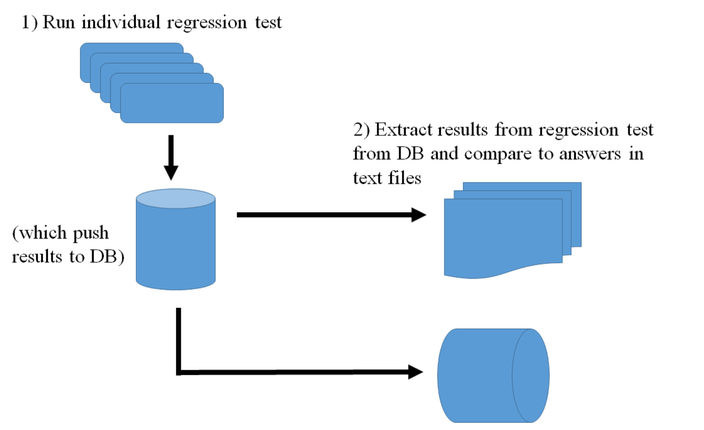
Regression Test

At CDDIS:

- An in-house solution was developed
- The tests check critical points required for correct completion and the outputs

Other Options:

- Most modern programming languages have packages that can be downloaded to support or enhance regression tests
- At it's base the outputs should at least be checked



Error Database

- Errors are pushed to the DB from the Code Base if any are encountered during runtime
- Warnings for files are also listed allowing for provider issues to be monitored and communicated
- A must if anything runs of crontab

filename	provider	error	location
allsat_20180912230O.npt	GLSL	Warning: noradID incorrect 00016908 16908	/slr/data/npt_crd/allsat/2018/
allsat_20180912230O.npt	THTL	Warning: noradID incorrect -1 07646	/slr/data/npt_crd/allsat/2018/
allsat_20180913000O.npt	MONL	Warning: noradID incorrect -1 07646	/slr/data/npt_crd/allsat/2018/
allsat_20180913000O.npt	MONL	Warning: noradID incorrect -1 08820	/slr/data/npt_crd/allsat/2018/
allsat_20180913000O.npt	AREL	Warning: noradID incorrect -1 43476	/slr/data/npt_crd/allsat/2018/
allsat_20180913000O.npt	AREL	Warning: satName incorrect:BEACONC BE-C	/slr/data/npt_crd/allsat/2018/
allsat_20180913000O.npt	AREL	Warning: noradID incorrect -1 01328	/slr/data/npt_crd/allsat/2018/
allsat_20180913000O.npt	AREL	Warning: noradID incorrect -1 22824	/slr/data/npt_crd/allsat/2018/
allsat_20180913000O.npt	AREL	Warning: noradID incorrect -1 16908	/slr/data/npt_crd/allsat/2018/
allsat_20180913010O.npt	GLSL	Warning: noradID incorrect 00016908 16908	/slr/data/npt_crd/allsat/2018/

Above is a snapshot of some information that is saved to the CDDIS database error tables. The information in these tables are used to generate reports so that we can contact the Ops centers with any issues we encountered from files sent.

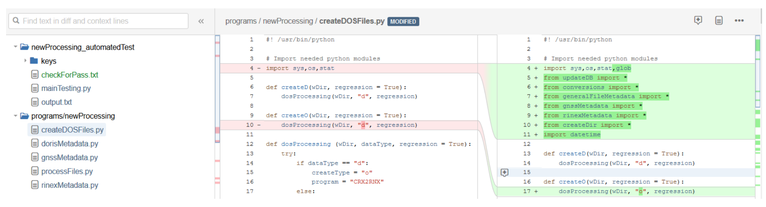
```
Error Report: 2018-09-04 - 2018-09-11


GLSL
Warning: noradID incorrect 00001328 01328
Warning: noradID incorrect 00007646 07646
Warning: noradID incorrect 00016908 16908
Warning: noradID incorrect 00027386 27386
Warning: noradID incorrect 00033105 33105
Warning: noradID incorrect 00036508 36508
Warning: noradID incorrect 00041240 41240

KUN2
Warning: satName incorrect:TECHNOSAT TechnoSat

HARL
Warning: noradID incorrect -1 07646
Warning: noradID incorrect -1 08820
Warning: noradID incorrect -1 16908
Warning: noradID incorrect -1 22195
```

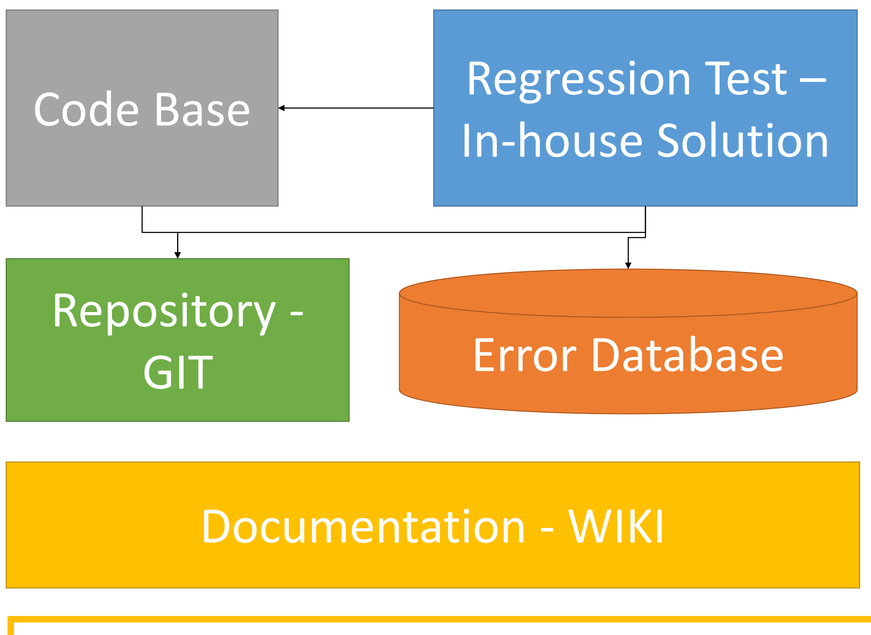
Repository

- Used to backup code base and regression test
- Provides version control that has helped us track changes, locate errors quickly, and revert to previous working versions
- Option to consider: Allows for replication if open source

Commit	Message	Commit date
31ae8fd6e3e	Update keys for GNSS and VLBI processing code	1 week ago
b494fefaf04	Correct IGS products and add initial doris products handling	28 Aug 2018
8fb1185d009	Update to Check only on technique	14 Aug 2018
9af063de222	GNSS products - latest types correction	03 Aug 2018
bd58acdf2e0	Regression Test Key Update	30 Jul 2018

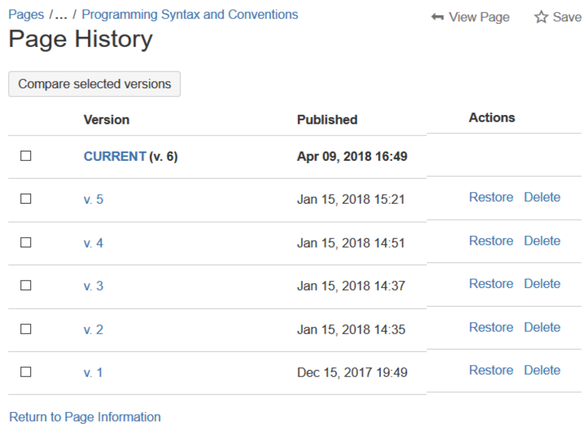
The commits shown above are a snapshot of the directory (and sub-directories under it) where changes were made to the code and committed to the repository. A commit can be selected and the changes made from the previous commit can be seen program by program.





Documentation

- Saved on a WIKI that tracks updates
- Accessible by everyone at CDDIS, especially useful when people go on leave



Encouraging Documentation Use: Brown-Bag Sessions

- CDDIS works to build familiarity with the documentation so that it is not a last resort for when an error occurs
- The CDDIS team meets and reviews new programs and their structure while going through the associated documentation. Participants are encouraged to add their understanding to the documentation to ensure it's clear.
- Additional Positive Effects:
  - Allows for code review to catch mistakes, for suggestions to be made, or to extend capabilities
  - Promotes team collaboration and support through various tasks
  - Shifting of existing and distributing of new tasks does not require a steep learning curve