Implementation of Automated Testing in a Continuous Integration Development Environment

By: Ashli Crookston, Derek Hampton, and Rachel Searle

Abstract
This poster introduces a unit testing system developed by Ashli Crookston, Derek Hampton, and Rachel Searle per the request of The Space Dynamic Laboratory’s Command, Control, Communications, and Computer Intelligence Surveillance, and Reconnaissance Division (C4ISR). This system consists of unit testing software for SDL’s image rectification functionality. The unit testing software tests the C4ISR Division’s existing software for newly introduced errors to the code. This software is automated so that it builds nightly and generates error reports that are sent to the developers. This poster demonstrates the background and need for our project, a short description of the design and its results, and finally the work breakdown structure for our project.

Design
In order to make sure the images were being geo-rectified correctly, it was essential to get three dynamics of flight rotation (roll, pitch, and yaw) calculations correct for both the plane and the sensor.
Roll: Rotation around the z-axis
Pitch: Rotation around the x-axis
Yaw: Rotation around the y-axis
Once these calculations were made, we were able to write our unit testing code. It consisted of 6 functions: initialization, sensor pitch, sensor roll, aircraft pitch, and aircraft roll.

Our project was completed successfully. The unit testing code is able to detect problems introduced into the original image rectification code and email the developers who have introduced these problems.

Contact
Authors: Ashli Crookston, Derek Hampton, and Rachel Searle
The Space Dynamics Laboratory—C4ISR Division.
Phone: (435)797-4275.
ashli.crookston@sdl.usu.edu
derek.hampton@sdl.usu.edu
rachel.searle@sdl.usu.edu

Supervisor: Mr. Pete Krull
The Space Dynamics Laboratory
Phone: (435)797-4275.
pete.krull@sdl.usu.edu

Phone: (435)797-4549.

Image Processing
- SDL develops an image processing software
- Aircraft with sensors capture aerial views of the earth
- Software processes, compresses, and manipulates images
- Imagery must be able to be "geo-rectified" so that north is at the top of the image

Image Rectification
The picture to the left is an example of a raw image taken by a sensor as an airplane flew overhead.
The picture to the right shows the same imagery after it has been geo-rectified. Geo-rectification adjusts an image so that north is facing up. This functionality is a key part of the C4ISR Division’s software.

Image Rectification

Line of Flight Imagery
The picture to the left shows the same imagery after it has been geo-rectified. Geo-rectification adjusts an image so that north is facing up. This functionality is a key part of the C4ISR Division’s software.

Unit Testing
Problem: In the past, the C4ISR hired students to manually test the software for bugs by pushing buttons and entering values. This is time consuming and costly to the company.
Solution: Create unit tests. Unit testing is the ability for software to test other software.
By using the Boost Test Libraries, we can implement test code that can be used to stress-test the existing image rectification code. This code can then be automated to run with a nightly build server, so that it is compiled each night. This nightly build also generates error reports if any changes to the image rectification code have introduced, bugs or errors, and the location where the errors were found. These reports are then automatically emailed to the developer who has introduced them so that the changes can be found and fixed as quickly as possible.

The System
The block diagram above shows our unit testing system. The original image rectification code and Boost Test Libraries are used to create unit testing code. This code is then sent to a C++ compiler and the nightly build server to be built.

Work Breakdown Structure

Project Impact
Global: Our project perfects the C4ISR Division software, which can and has saved many lives on both sides of war efforts. Automated testing will also produce higher quality software which will be used in future Department of Defense efforts.
Societal: Those working at SDL will be able to spend less time debugging (which is often boring and monotonous) and more time enhancing the software, which is typically a more enjoyable part of any software development job.
Economic: Automated testing will allow SDL to produce a better product. This will increase customer confidence. The developers will be able to use their time more efficiently because they will be able to focus on spending more time enhancing the software than trying to track down pesky bugs. It is also commonly known that the longer a bug goes undetected, the more expensive it is to fix. Our code will catch errors the same day they were created.

Sponsored by The Space Dynamics Laboratory’s C4ISR Division