2 Oct, 2017

Robert Trzeciak

Program Manager

Northrop Grumman IS

468 Viking Drive

Virginia Beach, VA 23452

757-498-5544 work

757-635-2628 mobile

**Progress Report – 1 Sep 2017 – 30 Sep 2017**

Contract Number: HSHQDC-06-D-00022

Contract Number 7500097279

Order Number: HSCG23-07-J-TED150

Task Order – Performance Work Statement (PWS) 1.12

Attachments: (1) SAROPS subcontractor financial reports.

1. **Jira meetings, e.g. Sprint Planning, Sprint retrospective, etc. Lots of meetings.**
2. **Finished Sprint 5 work:**
   1. **I’d accidentally introduced 2 problems with LOB/Flare.**
      1. **While cleaning up some logging code within SimLib, I introduced code that did an extra check for bad data. This extra check occurred before the logger had been initialized, and so when I tried to log the bad data, it crashed. Since I am always given bad data during intermediate steps of the GUI, this was a problem (at the end, I am given good data). I fixed this bug.**
      2. **The bad data was that I was being given was maxRange == 0. While debugging and tracing it down, I had changed the check on maxRange from “> 0” to “>= 0” to verify that that caused the problem. In one place I had changed “minRange >= 0” to “minRange > 0,” but minRange *is* allowed to be 0 and most often is. Hence, the bearings were all being ignored, and I had no scenario, which meant that I did not have a case.**
   2. **A 3rd problem was that for 2.2, I thought I had asked Young to put SimLand’s jarFile data file into a separate directory in my installed directory. But she was putting it into a directory that I intended to be for my use only, and therefore I thought I was free to delete, move, or rename. I went back to the old way.**
3. **Good progress with Judy on the Transit Optimization . She and I exchanged several phone calls and emails, and updated the example xml in . She produced one with independent PATTERN\_VARIABLEs (formerly called SRUs and now abbreviated as PVs) and a sequence of PVs. I modified it and used it in my demo.**
4. **Demo went well, even if I’m not sure who understood it. Using my GUI, I moved a PV so that the owning PvSeq was almost in violation or barely in violation. Then I (interactively) flipped a start point so that the PvSeq’s violation status flipped. A secondary display giving the amount of violation popped up as positive (or went away), as the transit legs on the display went from “ok color” to “violation color.” I asked people to trust me when I said that the Plan\_Result.xml would show the PvSeqV appropriately.**
5. **During this discussion, it seemed to me that everyone agreed that an answer is no longer a box, but rather a box PLUS cst and duration. I asked “do we agree that planner needs to send back cst and duration, or am I making things too complicated?” Robert responded, “that’s the whole point.” Judy added, “then I need to send back cst and duration when I send you a box,” and Jim noted that has a lot of ramifications outside of Planner. I feel now that any discussions about (eg) VS/SS patterns can wait until we are comfortable dealing with linked PVs (within a PvSeq), the total duration of a PvSeq, and cst/duration of a PV.**
6. **Made some progress on 465 which is where “the rubber starts to meet the road.” The trick is to modify the optimization and the code to handle PvSeqs but not to disrupt what happens to the Independents. There are 3 steps in an optimization pass and at least the first one must be done to satisfy 465. I’m not sure how the 2nd and 3rd tasks will play out, but my thinking is that I will simply mimic the logic that I introduced a long time ago, but was never used. I’m not sure how to handle this in Jira; each of the 3 steps requires significant code, and seems to be worthy of its own Jira task, and I would like to work on them, but we’re not allowed to create Jira tasks for the current sprint.**

**A “MyStyle” is my Java structure that stores center, orntn, length, width, ps/cs/ and firstTurnRight (the etymology is that I store this information in an easier way to work with than the above; those are stored directly in the Java structure “TheirStyle.”). I thought I would add cst and duration to that, but that didn’t work well. A “MyStylePlus” has a “MyStyle” and a PatternVariable, but putting it in the MyStylePlus didn’t work well either. Finally, I put it in a PvValue (formerly SruSolution), and I’m working through that. A PvValue can have an initialConfiguration (which happens when a box or a pattern is given to me), and these initialConfigurations, as per Judy’s comment above, WILL NOW include cst and duration. This scheme seems to be working out and once the PvValue and initialConfiguration have cst and duration, 465 should be considered done. I’m trying to put it in there by computing it in the first step of an optimization pass.**

1. **Did Sacramento Land. I was given a polygon, but it was intended as a “substitute path.” This input is called “mainWater.” A “substitute path” is when the user wants to modify a section of the shoreline (first application was NW Oahu). Therefore, I edited the input file to remove the “polygon nature” and make it a substitute path. But substitute paths are not supposed to cross any other polygon, and may cross the polygon being modified only between the start and ending points (such crossings involve only edges that will be replaced anyway). mainWater violated that principle because it intersected lakes that the user really wanted to turn into bays attached to mainWater.  
    Therefore I used the main operation “deletePolygon” to delete the 3 lakes that mainWater intersected, including the islands within them. I modified the delete code to capture the Lat/Lngs of these polygons in xml files. Only two of the lakes were intended to be added back and I set aside those two xml files, plus their islands. I modified the two xml files corresponding to the “add-back lakes” to remove their polygon nature and re-ordered the newly opened paths so that they started and ended close to mainWater. The final sequence of operations was:**
   1. **Create xml files corresponding to lakes and islands to be added back. Modify the lakes to be substitute paths.**
   2. **Then run the sausage maker. “Phase 4” is where all of the user-supplied edits are processed. So Phase 4’s data was loaded with all of the previous edits (eg Clatsop Spit, Hudson modifications, Oregon Inlet), and then:**
      1. **Delete lakes and islands**
      2. **Substitute\_Path mainWater.**
      3. **Substitute\_Path the files corresponding to lakes in step a.**
      4. **Add islands from input files**
      5. **Re-introduce islands from step a.**
         1. **Note; there was a bug in this code; the code was confused about adding polygons that it had already deleted. Had to chase that down and fix it.**

**For Sacramento (and the Hudson’s “stitching together” and Sacramento, considerable work was required to prepare Phase 4’s input files. Therefore, when Robert asked if the files he provided had any “errors,” I didn’t know how to respond. If, by “no errors,” he meant that they were ready to use, then they had errors. But I wouldn’t really classify these as errors. It’s just work that perhaps I should enter Jiras tasks for doing. I didn’t, but we should be aware that modifications that don’t fit nicely into the main operations (deletePolygon, addPolygon, substitutePath), are easy to underestimate.  
 As an afterthought, I built the jar files. The sausage maker builds shape files.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Activity Worked** | **Hours Worked** | **Hourly Cost** | **Total Cost** |
| Kratzke | Coding/Doc/Travel | 196.31 | -- | -- |
|  |  |  |  |  |
| **Totals** |  | 196.31 |  |  |
|  |  |  |  |  |