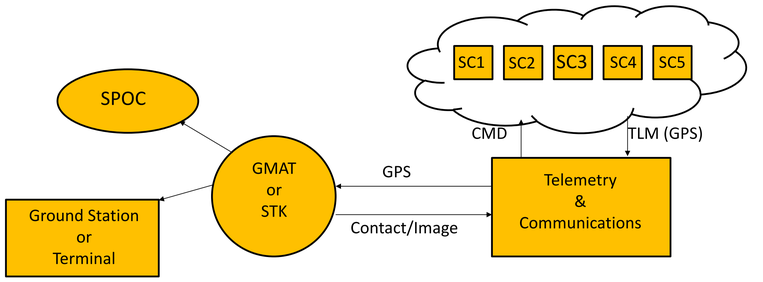
Station Observation & Locating Optimizer

Ahmed Woodson
Marcel Mabson
Angela Walters

Background

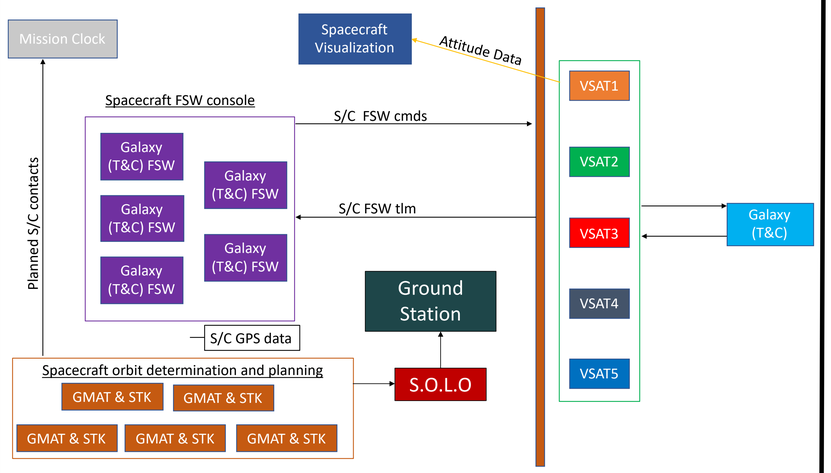
A mission planner for the Space Flight Operations Training Center (SFOTC) generates a ten day spacecraft contact prediction using the General Mission Analysis Tool (GMAT). This contact list contains all the expected contact times and duration between the various spacecrafts that are being observed and by each ground stations. The generated contact list contains various forms of information such as station and spacecraft name, and contact duration. For many of the predicted spacecraft contacts, the duration of the pass is not long enough to transmit commands or receive telemetry (or provide enough time for a control center to receive science downlinks) in a substantive manner.

Therefore, most predicted contacts have no usefulness to a mission planner and should be removed prior to processing. These contacts that fall below a threshold duration length will be removed and the mission planner will only be presented with contact durations that meet mission requirements.



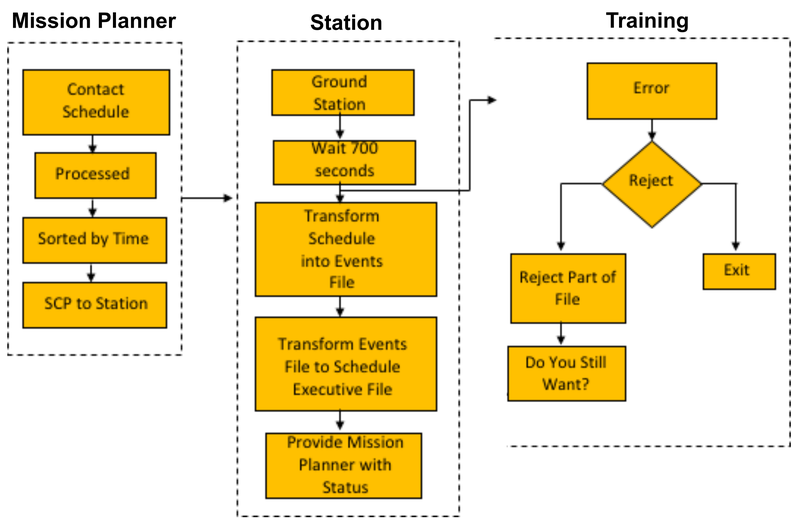
What Does The Program Do?

The code takes in a ten day ground station predication file from the General Mission Analysis Tool (GMAT), a software package that predicts orbital trajectories, spacecraft eclipses, available solar array power and spacecraft station contacts, and begins to manipulate the file. Once the file is accepted by the code. The code then sifts through all of the contacts in the file and deletes anything that is not of value to a mission planner (i.e. anything that is not a date). The code then prompts for user input. It asks the user what minimum duration it should display. After the input is given, the program then sifts through the file once again and finds all of the durations that are greater than or equal to the users input. After all of the durations are found, a new file is created with all of the contacts of the appropriate durations. The file also outputs how many contacts there are in that file. The file is now sent to mission planner; an 11 minute wait begins to simulate the actual contact between the mission planner and station. After that the user on the mission planning side is prompted with a Y/N question of whether or not this file is adequate or not an events file is created and the program.



Running The Program: An Overview

1. Ensure that the program, S.O.L.O. is on the hard drive of the computer you are using.
2. Ensure that the program is on the Desktop or easily accessible.
3. Open a new terminal (user should automatically be in the Desktop directory).
4. Navigate to the appropriate directory that houses the code.
5. If the program is somewhere other than the Desktop, the user must access the directory that the file is housed in.
6. Once that directory is accessed, compile the program (g++ “program name”)
7. Then run the program.



Methodology/Testing

There were several methods of testing that occurred before the final iteration of the program was developed:

- *Requirement Testing*: ensuring that the program undergoes a pass/fail assessment that maps the requirements to the programs ability to complete the task.
- *Software Testing:* simply made sure the program met the requirements of the scope of the project.
- *Error Testing*: giving the program incorrect or corrupt input information to ensure that the program can properly adapt to “bad” information.
- *Edge Case Testing*: attempting to push the program to its limits to see how much operating power it contains.

Conclusion

The development of the code, S.O.L.O. was a success. It passed all tests and has been completely integrated into the Space Flight Operations Training Center (SFOTC) operating system. The code will be used for the training purposes of Capitol Technology University’s aerospace engineering students. Hopefully the program can be further developed and integrated into mainframe within NASA’s mission planning operations.

Future Work

If allowed to continue work on the Station Observation & Locating Operative, I will go into the base code of the program and use more efficient methods of achieving the same goals. I would like to make it more flexible in the type of files in inputs and outputs. Lastly, I would create an overall more fluid and free ranging system for the code to operate on.

Acknowledgements

I would like to extend a very personal thank you to the following:

• The Maryland Space Grant Consortium
• Dr. Mary Bowden
• Professor Angela Walters
• Professor Marcel Mabson
• Mom & Dad