

Satellite Testing: Test Results Database and Python GUI

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Connor Moon

Spring, 2023

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Briana Morrison, Department of Computer Science

Abstract

NASA engineers testing modems for satellites needed a backend database to store test results, as well as an intuitive GUI to be able to access and query said database. This problem was solved by creating a Microsoft Access Database and connecting it through Python pandas code to a Python GUI. To understand the problem, research into how these engineers ran tests and what kind of results they receive had to be performed. Once I completed my research, I created a UML to plan out the intended solution. After the planning stage, I created a database and wrote code to execute the plan. The result was a functioning database and linked GUI which could be used to add tests, remove tests, and query the current database. Future plans following the conclusion of this internship project were to integrate this database and GUI tool into a larger testing GUI in development.

1. Introduction

For decades, NASA engineers and contractors have been using a decades-old GUI to run modem tests for their satellites. At the onset of my summer internship, I was introduced to a programmer on the team who had been working on developing a completely overhauled GUI for their testing purposes. This project had been in development for over a year, and had significantly progressed. The other intern on my team was to develop the test creation side of the GUI; I was assigned to the test results database backend and GUI frontend. These tests include wildly different parameters, based on the wavelength and modem hardware/firmware, among others. I had to research these parameters, because the type of data that results from modem tests is different, depending on the input parameters.

2. Related Works

In the beginning of his book on GUI programming, Harwani (2018) provides a lot of helpful tips on the most important parts of creating a GUI. In particular, chapters 1 and 2 really influenced the design process for my GUI [3].

Siahaan (2019) also introduces a lot of PyQt concepts which were helpful, but it goes beyond that and discusses a lot of beginner concepts for Microsoft Access databases. This mirrored what my project was doing, and offered helpful tips on how to integrate the database into a python GUI [4].


3. Process Design

The design of my project required some initial research to be performed. After I understood the project requirements, I began construction on my backend database, before finally moving on to my GUI.

3.1 Research

Without delving too much into proprietary or government-sensitive information, my research during the early stages of the project mostly involved understanding TDRS (Tracking and Data Relay Satellites) as a whole, and more specifically the capabilities and specific tests run on the ground station modems [1]. These modems and satellites currently support the Mars missions, and the testing my team was working on was for the upcoming Artemis missions.

Specifically, regarding the ground station modems, some of the most important test parameters include polarization, carrier modulation, carrier data rate, subcarrier modulation, and decoding, seen in Figure 1 below [2]:



Characteristic	Value
Frequency	2200 – 2400 MHz
G/T	22.8 dB/K (clear sky & 41° elevation angle)
Polarization	RHC or LHC
Antenna Beamwidth	0.85 deg
Antenna Gain	42.8 dBi
Carrier Modulation	PM/PCM, FM/PCM, BPSK, or QPSK / OQPSK
Modulation Index	PM: 0.2 – 2.8 Radians (peak)
Carrier Data Rate	1 Kbps – 10 Mbps (FM/PCM)
(High Rate Telemetry Channel)	100 bps – 20 Mbps (PM/PCM, BPSK, QPSK)
	1 Kbps – 40 Mbps (QPSK) (< 20 Mbps per channel)
Carrier Data Format	NRZ-L, M or S; Bi-L, M or S; DM-M or S; DBP-M or S; PMS?
Subcarrier Frequency	5 kHz – 2 MHz
Subcarrier Modulation	PSK, BPSK, PCM/PM for high BW telemetry
Subcarrier Data Rate	100 bps – 800 Kbps
Subcarrier Data Format	Passes all NRZ or Bi-L or DM
Decoding	Derandomization: Viterbi and/or Reed-Solomon (Ref Para 1.3.5)

Figure 11.6: S-band Telemetry Characteristics for the WGI Antenna at NASA Wallops Flight Facility
Credits: NASA

Figure 1: Modem Parameters

3.2 Database Approach

After I fully understood these tests and their input parameters and output data, I began to plan out how my database and GUI would function and interact with each other. Because my end goal was an isolated product, which would be later incorporated into a larger project, we decided to use a Microsoft Access database over something much more powerful databases like a standard SQL database purely for simplicity. While Microsoft Access databases are not like databases typically used in industry, they can still be queried by SQL commands, which would allow for this prototype to be easily transitioned into a different SQL database.

For the actual design of the database itself, I came up with multiple approaches. One approach had a different table for each test type; other approaches had different tables based on other input parameters (test performer, date, etc.). After consulting with the engineers, I decided that utilizing different tables for different test types would be the most optimized approach, as that would represent their most frequent queries.

3.3 GUI Approach

Due to my familiarity with Python, Python Pandas, and PyQt from my classes at UVA, and also because that was the only programming language any of the engineers had experience in, that was the language and libraries I chose for the GUI code. I first had to design a high-level diagram (similar to a

UML diagram) which would help determine how many different tabs the GUI would have, as well as which of the other tabs each tab would interact with. After reviewing my design with the engineers, I moved forward with the coding.

I created a central landing page which housed the ability to add a new test, delete a test in the database, or to query the database. The add test result page would allow the user to select and test type and add all the input parameters to create a test type and send it to the appropriate database. This was a necessary functionality for this prototype, but would eventually be replaced via integration with the overall testing GUI (which would automatically insert results of a test into the database). The query page would bring up a new tab with a drop-down bar containing all the test types as well as all of the non-test-specific parameters. If a test type was selected from the drop-down bar, then the tab would also populate with the test-specific parameters. Last, the delete tab would bring up a similar window to the query page, except the end result would have an option to delete the specified test(s).

4. Results

By the end of my summer internship, I had completed most of my goals for the project. The database and GUI, which cannot be shown due to government-sensitive information, were both in a finished state, capable of adding tests to the database, deleting tests from the database, and querying the database all from the python GUI. After I returned to school, my manager informed me that a new team of software developers had been brought on to integrate the project into the larger GUI and finish it, marking the end of an updated GUI replacing a decades old-extremely outdated system.

5. Conclusions

The work I performed on this testing GUI and database was essential for the future of NASA's Artemis Missions. Thorough testing of all aspects of the missions, including the satellites used to communicate and monitor during those missions, is essential. My improvements to the GUI will make testing these modems and satellites easier as well as quicker. While not in a state to be fully integrated into the larger GUI yet, my project laid the foundation for the tool to be used in the future.

6. Future Work

Upon completion of my internship, other members of my team began to work on integrating my solution into the larger GUI. This involves creating a more practical database for long term use, such as a SQL database. While the Microsoft Access database was the best solution for my project, it is not ideal for long term use due to limitations. Fortunately, accesses to a Microsoft Access database through python are already formatted in SQL format, so changes would not need to be made to the GUI code to access a new SQL database.

7. UVA Evaluation

While much of the UVA curriculum prepared me for this learning event (using python to create GUI's and communicate with databases, among other things), there were also many important aspects of my learning event that UVA did not prepare me for. Many of the less technical skills, such as working in an agile environment, are not a large focus of the classes taught at UVA, despite being highly important to most industry work. The only class which spent any time attempting to teach agile development was Advanced Software Design, which also could benefit from a greater emphasis on agile design.

8. Acknowledgements

I would like to thank my excellent manager and mentor, David Schuchman, for teaching me so much about satellite communications. Additionally, I would like to thank my fellow intern Alexander Cochran, whom I collaborated with on the GUI and backend database.

References

- [1] Danny Baird. 2020. Space Communications: 7 Things You Need to Know. NASA's Space Communications and Navigation program (Oct 2020). <https://www.nasa.gov/feature/goddard/2020/space-communications-7-things-you-need-to-know>
- [2] NASA. 2021. 11.0 Ground Data Systems and Mission Operations. <https://www.nasa.gov/smallsat-institute/sst-soa/ground-data-systems-and-mission-operations#11.5>
- [3] B. M. Harwani. 2018. Qt5 Python GUI Programming Cookbook: Building responsive and powerful cross-platform applications with PyQt. Packt Publishing.
- [4] Vivian Siahaan. 2019. Access Database for Pragmatic Programmers: A Step by Step Guide to Create Database-Driven Application Using Python