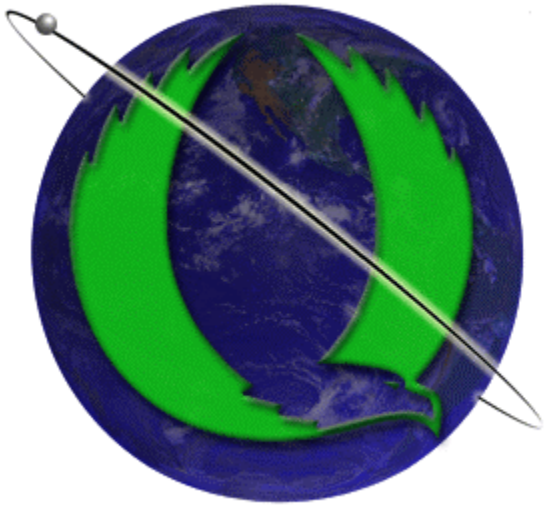
SAT-NET: Satellite Acquisition &Tracking with Network Enabled Telescope
Florida Gulf Cost University
Fort Myers, Florida, USA
http://satnet.fgcu.edu/



Florida Gulf Coast University

Team Members	Dustin Felton	dgfelton@eagle.fgcu.edu
	Sean Kerwin	spkerwin@eagle.fgcu.edu
	Anders Petersen	ahpeters@eagle.fgcu.edu
	Ray Rhode	rarhode@eagle.fgcu.edu
Project Mentor	Janusz Zalewski, Ph. D	zalewski@fgcu.edu
Mentor Assistant	Andrew White	aswhite@eagle.fgcu.edu

Abstract

As the population of the world grows and tends much more to travel, we are becoming increasingly dependent on communication, navigational aide, and weather forecasting. As such, protection of the infrastructure of these ends becomes irrevocably more important as well. The SAT-NET system is a means to help in making such protection a reality. SAT-NET allows for distributed monitoring of communication satellites and other orbiting bodies, thus giving anyone with a computer and Internet connection the ability to monitor these increasingly important resources.

This report documents our experiences in implementing this vision. It quantifies efforts of the Project Team in creating SAT-NET, with respect to requirements specification, design, implementation and testing, and summarizes the project status for today:

1) The design methodology applied draws significantly from the IEEE Std 1074-1997 "Developing Software Life-Cycle Processes" and its counterpart ISO/IEC Publication 12207 "Software Lifecycle Processes", with additional focus on quality assurance activities according to the IEEE Std 730 "Software Quality Assurance Plans".
2) The software design architecture applies the David Parnas' principle of separation of concerns, which divides software functions based on the I/O responsibilities, regarding telescope control, web camera control, database interface, and user interface, among others.
3) The implementation method uses object-oriented technologies, including the latest available version of Java and design patterns, of which one was created in this project (Virtual Singleton Pattern).
4) The testing process relies on following IEEE Standards [8] and focuses on performance measurements, with respect to computational accuracy and server's robustness.
5) Tools developed during this project, which include a new communication protocol, a new design pattern, `perl` package for dumping SQL database contents, and Java package for command and communication with external devices using serial port, are made available for the public and may serve general population of designers.

The *beta* version has been released. It can be accessed from `satnet.fgcu.edu` and by making prior email arrangement with team members. Full testing is still being pursued and was conducted successfully during the FGCU Research Day Student Poster Competition, where the Team won first price (`www.fgcu.edu/orsp/`). Further work is needed to fix minor bugs and coordinate operation of the webcam with telescope control software.

1. System Overview

1.1 Technical Requirements

The objective of the SAT-NET (Satellite Acquisition & Tracking with Network Enabled Telescope) project is to develop software providing Internet connectivity to a telescope allowing satellite tracking [9]. SAT-NET can be used by organizations to enhance safety of their air space and track intruding vehicles. It also allows users around the world to view, track and learn about satellites from a Java-enabled web browser. Connected by the Internet, the system allows for expandability, so that many servers can be set up to view a satellite or other space object, utilizing multiple site locations, as it moves across air space or makes an entire orbit around the Earth.

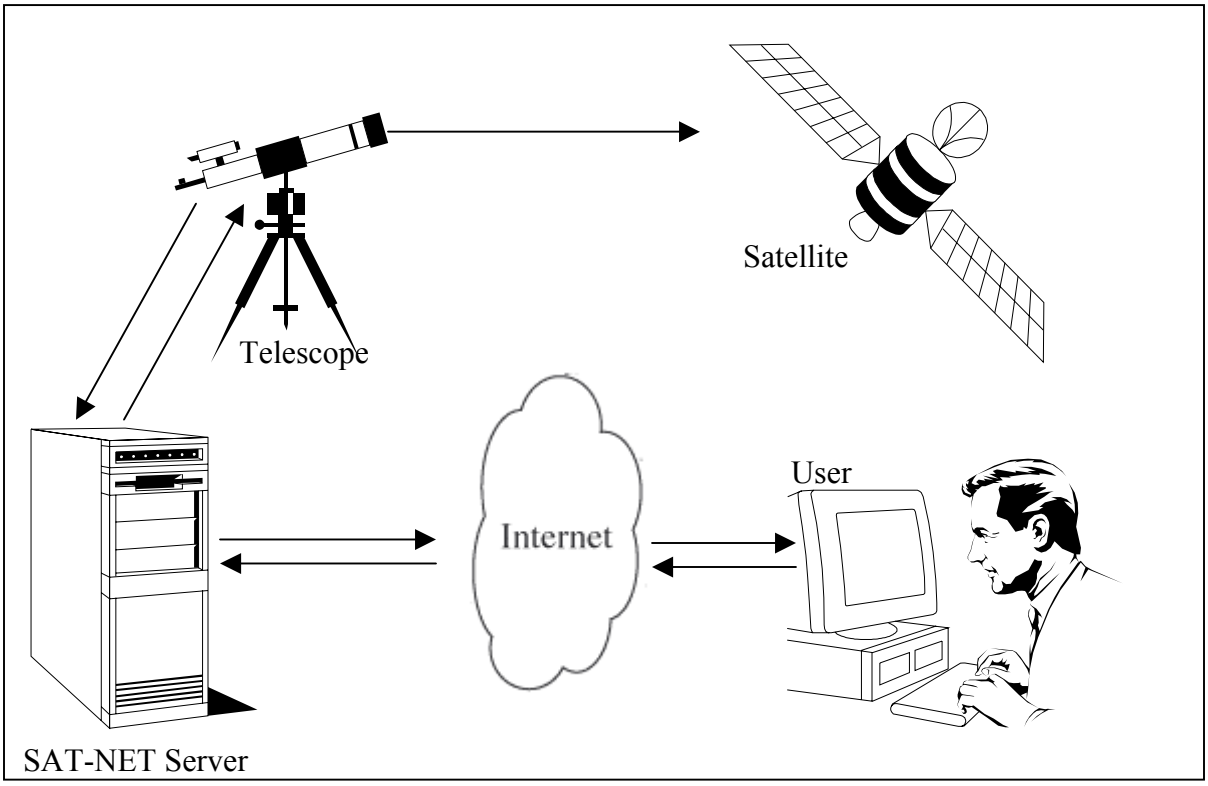


Figure 1. SAT-NET Overview.

As shown in Figure 1, the basic implementation of SAT-NET is simple. A user, using a Java-enabled web browser, connects through the Internet onto the SAT-NET server. The server then relays commands to the telescope, and tracks the satellite. Simultaneously, a web camera sends the real-time image back to the user for viewing. Specific functional requirements are listed in Section 2.1. Performance requirements include: accuracy of computations of satellite positions and server robustness to increasing load. They are defined in Section 2.2.

1.2. Design Methodology

Design methodology follows the logic of IEEE Std 1074-1997 "Developing Software Life-Cycle Processes" and its counterpart ISO/IEC Publication 12207 "Software Lifecycle Processes", with additional focus on quality assurance activities according to the IEEE Std 730 "Software Quality Assurance Plans", all involving pre-development, development and post-development activities conducted with object-oriented technologies, as presented in Table 1:

Table 1. Processes and Activities in Design Methodology (adopted from IEEE Std 1074 [8]).

Processes and Activities	Support Tool	Deliverable	Quality Assurance
1. Pre-development			
a) Mental Preparation	Kidder's book [1]	Internal discussion	
b) Domain Knowledge Acquisition	Consulting experts [2-7]	Record in minutes	
c) Team Organization	Organizational theory	Assignment of responsibilities	
d) Feasibility Study	IEEE Standards Collection [8]	Preliminary design decisions	
2. Development			
a) Software Requirements	UML Uses Case Diagrams	S/W Requirements Specification	Walkthrough (wrt. clarity, completeness, correctness, consistency, traceability, testability)
b) Software Design	UML Class and Sequence Diag.	S/W Design Description	Design Review (wrt. coupling & cohesion)
c) Coding	JDK 1.4.2	Code	Code Inspection
d) Testing	Test Execution	Test Plan and Test Report	Test Review
e) Documentation	MS Word	User Manual	English Faculty
3. Post-development			
a) Report Preparation	MS Word	Final Report	
b) Beta Release	JDK 1.4.2	Code beta version	
c) S/W Modifications	UML diagrams & JDK 1.4.2	Code production version	
d) User Training	HTML	Web documents	

1.3 Technical Innovations

1. **Remote operation of a telescope for satellite tracking via the Internet.**
   Satellite tracking is a rather standard operation in scientific community, but thus far no system has enabled a telescope with Internet accessibility; SAT-NET is the first to make this integration.
2. **Visual satellite tracking with a real-time video feed.**
   Real-time satellite tracking is available but on a text-only basis. SAT-NET will allow visual tracking of a satellite in real time.
3. **Potential to track a satellite through the entirety of its orbit.**
   SAT-NET will allow for tracking of a satellite around the globe, utilizing multiple SAT-NET stations.
4. **Exapandibility, allowing SAT-NET's inclusion as module of a larger system.**
   Modularity of the SAT-NET software and its open network protocol permits its use by a system with full-scale functions of greater importance, such as air traffic control system or satellite ground control station.

2. Implementation and Engineering Considerations

The SAT-NET has two functioning modes, a User Mode and a View Mode. The User Mode logs a user into the system, allows them to schedule time when they can control the telescope and allows the user to take control of the telescope. The View Mode, allows all viewers to watch a real-time video stream of a web camera attached to the telescope. The operation of the SAT-NET software is defined in the Software Requirements Specification (Section 2.1 below). The Design Description is presented in Section 2.2, followed by the list of major design decisions and tools developed (Sections 2.3 and 2.4, respectively). Test results are presented in Section 2.5.

2.1 Software Requirements Specification

Software requirements have been derived with help of Use Case diagrams, which are not shown here for the lack of space. For better clarity, the requirements are split into Functional Requirements and Non-functional Requirements (safety, security, and performance).

2.1.1 Functional Requirements

1. The SAT-NET software shall allow free access to satellite viewing for anyone on the Internet from a computer equipped with a Java enabled browser.
2. The SAT-NET software shall allow scheduled access from the Internet to telescope control, with registration and verification for regular users.
3. The SAT-NET software shall allow registration based on valid email addresses and only for email addresses not previously used.
4. The SAT-NET software shall require the following data for user registration: first name, last name, date of birth, email address.
5. The SAT-NET software shall allow entering optional data, such as occupation, reason for registering, latitude, longitude, country, city, zip code, and affiliation.
6. The SAT-NET software shall allow users to request their passwords to be emailed to the address they provide.
7. The SAT-NET Software shall allow clients to connect to the server and view telemetry relating to the satellite currently being tracked.
8. The SAT-NET software shall allow scheduled access for controlled tracking of the satellite, based on logon procedures and, after logon, on prioritizing users and first-come-first-served principle.
9. The SAT-NET Software shall allow connected clients to request authentication: if the credentials sent correspond to a registered user who is not currently banned, the request shall be granted and the client shall become a logged-in client; otherwise the authentication request shall be denied.
10. The SAT-NET Software shall grant control privileges to a single logged-in client at a time, allowing that client to select a new target for the software to track.
11. The SAT-NET Software shall allow logged-in clients to view a list of currently visible satellites.
12. The SAT-NET Software shall allow logged-in clients to request a reservation of time at a future date during which they will have control over the software, which shall be granted if and only if the requested time period is of duration within specified tolerances, does not conflict with any preexisting reservations, and does not exceed a specified per-user per-day time.
13. The SAT-NET Software shall allow logged-in clients to view the time and duration of current reservations.

14. The SAT-NET Software shall allow logged-in clients to request control at any time, which shall be granted if either a) the client has a reservation for the current time b) no client currently has control or c) the client currently in control does not have a reservation and has been in control for longer than a specified duration; otherwise the client's request shall be denied.
15. The SAT-NET software shall allow a superuser access to telescope control with preemption of other users.
16. The SAT-NET software shall allow superuser assign rights to users, administer user accounts, and overwrite schedules.
17. The SAT-NET Software shall allow logged-in superuser clients to request control at any time, which shall be granted if control is not currently granted to a superuser client; otherwise the superuser client's request shall be denied.
18. The SAT-NET software shall provide access to explanation of site policies on the front page of the Satellite Tracker website.
19. The SAT-NET software shall provide guidelines explaining the principles of operation of the satellite tracker, accessible from the Satellite Tracker website.
20. The User Interface shall allow a regular user to preselect a satellite with predefined Keplerian elements or enter their own Keplerian elements.
21. The SAT-NET software shall allow all users to view the data from the satellite that is currently being tracked.
22. The SAT-NET software shall allow all users to view the currently available schedule for the satellites being tracked.
23. The SAT-NET software shall allow automatic booking of satellite tracking time for regular users, within available time for the telescope.

2.1.2 Non-functional requirements.

24. *Safety.* The SAT-NET software shall take precautions against rotating the telescope too far to cause damage to power cables running from the base of the telescope to the side motor.
25. *Security.* The User Interface shall be required to verify to the server it is a valid instance of the interface before it is allowed to send or receive any information
26. *Security.* User information sent by the User Interface to the server shall be encrypted before being sent to prevent the interception of personal and login information and, upon receipt by the server, the information shall be decrypted for verification.
27. *Security.* All passwords shall be at least six characters long and must contain at least one digit and one letter.
28. *Security.* In case a user has failed to login to an account five successive times, the SAT-NET software shall have the account locked for one hour, during which even if the correct password is entered the user cannot login.
29. *Security*. Passwords sent to the server by the client during authentication shall be transmitted only in an enciphered form such that potential interceptors would be unable to determine the deciphered form.
Note. The enciphering process should be dependant upon a pseudo-random seed generated by the server, so as to prevent simple playback for a recorded authentication exchange.
30. *Performance.* Accuracy of satellite position computations (azimuth, elevation, latitude, longitude, and range) shall be better than 0.01%.
31. *Performance.* Responsiveness of SAT-NET server shall not degrade more than 5% per each 10 clients requesting service simultaneously.

2.2 Software Design Description

Overall system architecture is presented in Figure 2, in a form of a context diagram. It includes all external devices: telescope, web camera, database server, and Internet access to clients and time server (NTP). From the architectural standpoint, two major components in the SAT-NET software were distinguished: the Satellite Tracker (playing the role of a SAT-NET Server), which runs on a server machine and is responsible for telescope interaction, and the Remote User Interface, which runs as an applet within any browser and handles all human interaction. These two main components are built out of separate modules, each of which performs specific functions of the SAT-NET software.

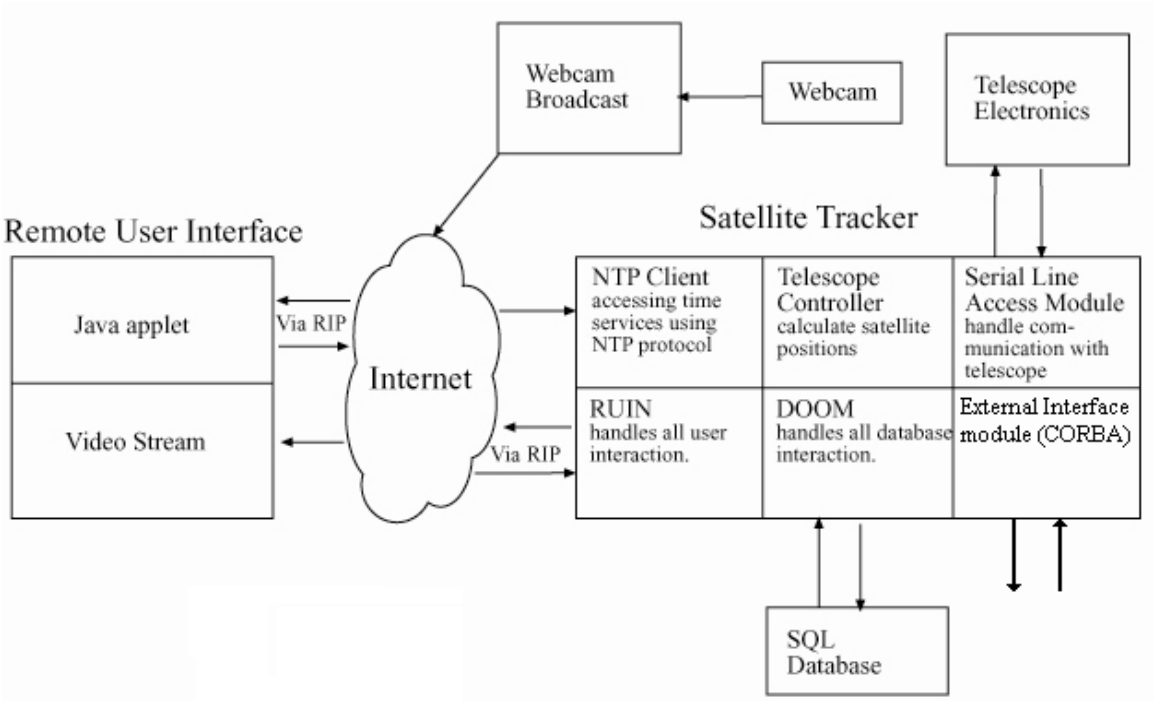


Figure 2. System Architecture/Context Diagram.

The basic software architecture of SAT-NET is based on *the principle of separation of concerns*, which divides software functions based on the following I/O responsibilities: user interface, equipment control, database access, network connectivity, precise time keeping, webcam control, and internal computations. Accordingly, these functions have been mapped on their respective modules as follows:

- **user interface** is handled by a *Java Applet* executing within a browser, passing login, telescope scheduling and other information to the *Remote User Interaction Negotiator* (RUIN) module, which has access to the database manager
- **equipment control** (telescope) is handled by *Serial Line Access Module* (SLAM) receiving commands from the computational module, Telescope Controller
- **database interface** is handled by the *Database Object Operations Manager* (DOOM) module, which take relevant data from the Java Applet and provide these data, upon request, to the computational module, Telescope Controller
- **computations** are concentrated in the *Telescope Controller*, which is responsible for calculating satellite positions and interacting with: the Database module to receive and

send data; the SLAM to communicate with the telescope; and the NTP Client to access precise time information

- **time keeping** for calculating satellite positions, is handled by the *NTP Client*, synchronized with an NTP server and providing time to the Telescope Controller
- **network connectivity** other than Internet is provided by an *External Interface* for future expansion
- **webcam broadcast** is provided by a public domain version of the QuickTime software.

2.2.1 User Interface Design

The basic appearance of `satnet.fgcu.edu` website is shown in Figure 3. From there, the use can register and log on to satellite tracking functions, which is illustrated in Figures 4 and 5, respectively.

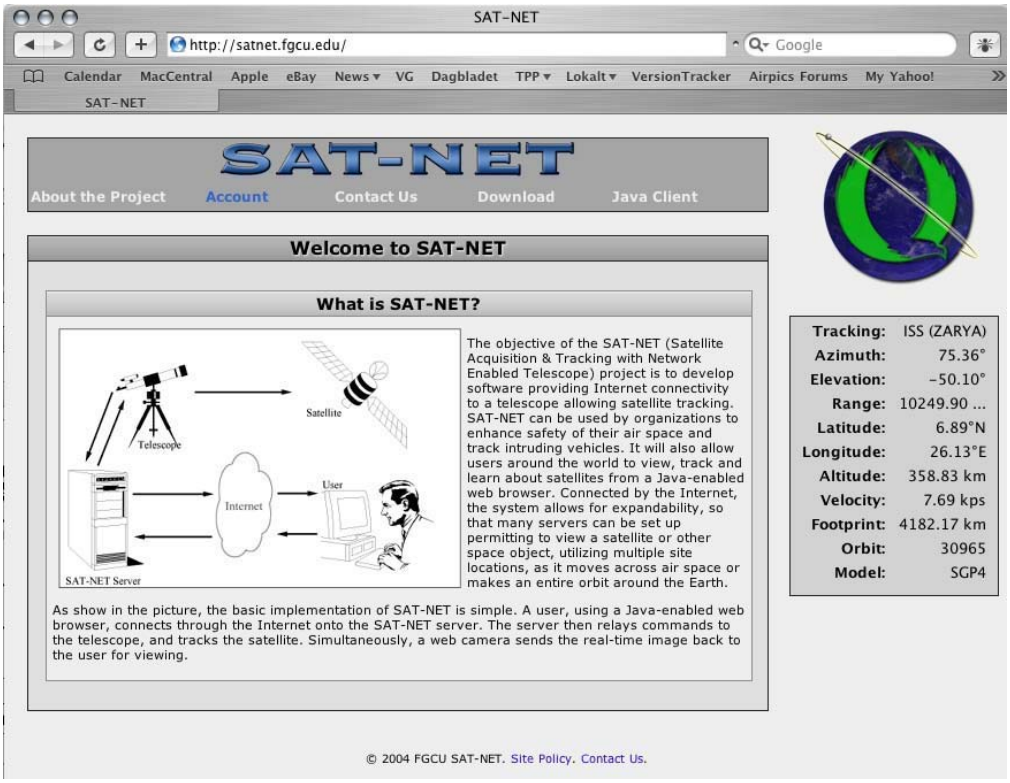


Figure 3. Main Webpage for the Satellite Tracking.

Upon browsing to the main SAT-NET interface page, the user is presented with Figure 3. The left half of the page (referred to as the control pane) is the SAT-NET Java client applet, while the right half is the live video stream coming from the telescope. The telemetry data on the upper half of the control pane will be continually updated to reflect the current focus of the telescope. Upon clicking the 'Log In…' button the user will be presented with the 'Log In' dialog box.

If the credentials entered are valid, the 'Log In' dialog box will close and the control pane will update. The status message in the upper-right hand corner will change from 'Not Logged In' to 'Logged In'. If the user is logged in with special credentials, e.g. as an administrator or priority user, that status would be reflected in this message as well.

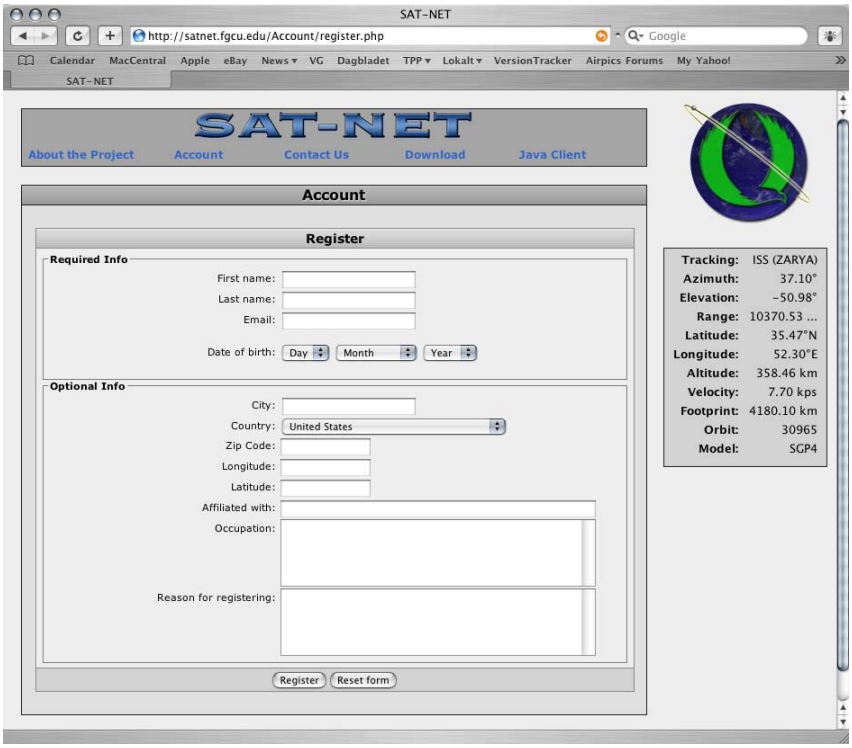


Figure 4. SAT-NET User Registration Page.

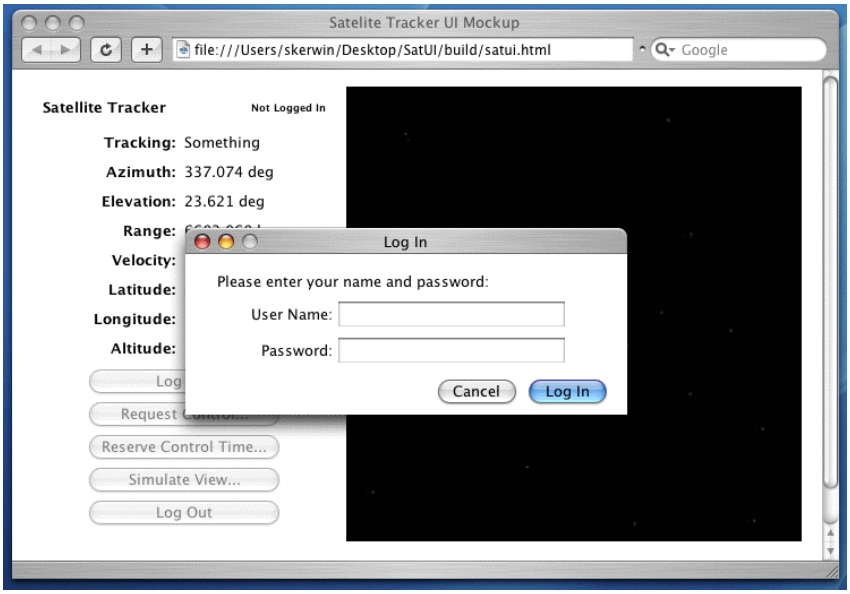


Figure 5. Logging in to SAT-NET to Access the Scheduling and Control Features.

If the user wishes to reserve control time for the SAT-NET system, they may do so by clicking the 'Reserve Control Time…' button on the control pane. This will lead to the appearance of the reservation window (Figure 6). The user may select a date and time by manipulating the popup buttons at the top of the window; as these values are altered, the time bar across the middle of the window will update correspondingly. The blue section of the bar represents the time the user is currently attempting to reserve, while red sections show times for

which reservations already exist. If the user were to alter the value of the Duration popup to two hours, the blue box would overlap the rightmost red section. The overlap would be colored in purple to indicate a conflict, and the 'Request Reservation' button would be disabled until the conflict was removed.

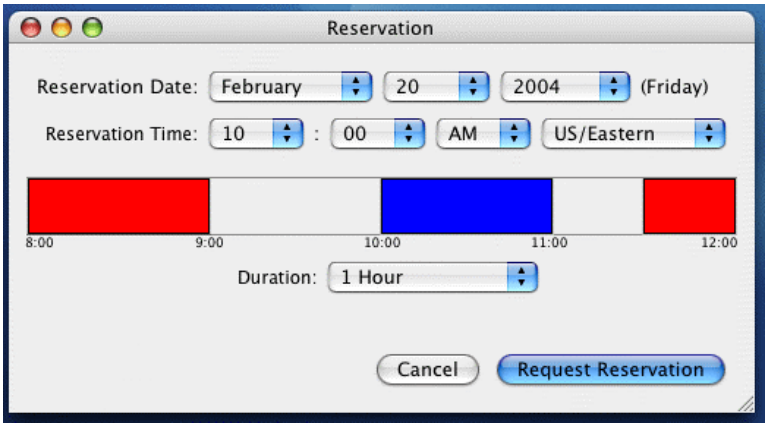


Figure 6. Requesting a Reservation to Control the Telescope at a Future Date and Time.

2.2.2 Class Design

The overall structure of SAT-NET software, as derived from the System Architecture of Figure 2, is illustrated in a class diagram shown in Figure 6. It represents the relationships between a main class, BOSS, and other classes controlling, respectively, the telescope (SLAM), computations (MADAM), user interface (RUIN), database access (DOOM), and time (NTPC).

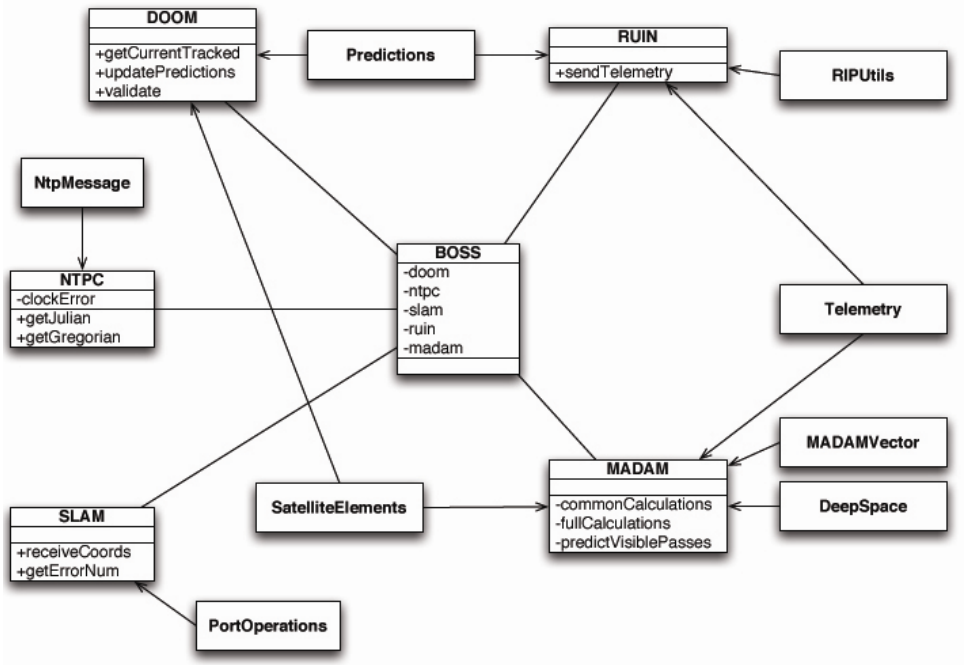


Figure 7. SAT-NET Class Diagram.

Detailed designs of respective classes are discussed in the following subsections only partially, due to space limitations.

2.2.2.1 Java Applet Design

A Java applet, named WRAK, is the core of the SAT-NET remote interface. It is responsible for coordinating the various user interface elements and communicating with the RUIN (Remote User Interaction Negotiator) module residing in the SAT-NET server. WRAK is entirely event-driven, receiving both local events generated by the various user interface elements and remote events originating with the server. It communicates with the server by sending similar events/messages to the RUIN. State is enforced through the dynamic display/hiding and enabling/disabling of user interface elements.

The bulk of WRAK's functionality is controlled by the Java event model; the main WRAK class is an implementation of the ActionListener interface and is registered to listen to pertinent UI elements. WRAK also receives events from WRAKComm, the networking module. Once instantiated, WRAKComm launches a new thread to receive information from the server. Information received is parsed and, if it represents a known event, relayed to WRAK.

Table 2. Externally-Generated (Network) WRAK Events

Event Name	Handling
`kREMOTE_CONNECTED`	1. Enable log in button 2. Set status display to 'Not Logged In'
`kREMOTE_FAILED`	1. Disable all buttons 2. Display alert 3. Construct retry thread and start
`kREMOTE_DROPPED`	1. Disable all buttons 2. Display alert
`kREMOTE_LOGIN_YES`	1. Change status display 2. Enable request control button 3. Enable reservation button 4. Enable logout button 5. Dispose WRAKLogin
`kREMOTE_LOGIN_NO`	1. Call WRAKLogin.credentialsWereInvalid( )
`kREMOTE_LOGOUT`	1. Enable log in button 2. Disable request control button 3. Disable reservation button 4. Disable logout button 5. Set status display to 'Not Logged In'
`kREMOTE_TELEMETRY`	1. Call WRAKControl.updateTelemetry( )
`kREMOTE_CONTROL_YES`	1. Instantiate new WRAKTargeting
`kREMOTE_CONTROL_NO`	1. Display alert 2. Enable request control button
`kREMOTE_VISIBLE`	1. Call WRAKVisibleDataModel.updateList( )

The main class of the WRAK applet, WRAK is created automatically by the web browser and is tasked with responding to all remote and local events. Unlike most subclasses of `Applet` or `JApplet`, WRAK does not implement its own user interface, instead using the WRAKControl class for that purpose. As special method is written, as dictated by the ActionListener interface, to which all buttons in the various user interface components ultimately report. It is implemented as a large switch statement, with a separate case for each supported event. Corresponding events are split into internal (generated by user interaction with user interface elements created by one of the WRAK classes) and external (generated by the WRAKComm module in response to messages from the server) events. In this report, only external events are presented (see Table 2).

2.2.2.2 RUIN (Remote User Interaction Negotiator) Design

RUIN is the server element of the SAT-NET software. It is tasked with accepting connections from clients, authenticating clients, and responding to commands and requests from clients. The design of RUIN is intended to be maximally scalable; care was taken to minimize per-client resource allocations. Most notably, RUIN makes use of the Java's newest I/O facilities, the NIO package, to ensure that the number of threads is constant regardless of client load (efficient use of Java's previous I/O facilities often required one thread for each connection).

RUIN contains three threads of execution. The first, the acceptor thread, is tasked with listening for incoming socket connections. When a connection occurs, the acceptor thread instantiates a Connection object (a nested class within RUIN) and adds it to a vector member containing all current connections. Each connection object maintains a buffer of outgoing data, which is written to that connection's socket as its write buffer frees space. This writing is accomplished by the second thread, the I/O service thread. The I/O service thread is also charged with reading data from sockets as it becomes available, parsing the data into RIP packets using the RIPUtils class, and handling them appropriately. The final thread maintains a packet representing a list of the currently visible celestial objects; because this list is request by clients quite frequently, it is more efficient for it to be done once in a separate thread of execution than to be done many times as clients enquire.

RUIN communicates only with the MADAM and DOOM modules. The MADAM module periodically calls RUIN's sendTelemetry( ) method, which is used to inform clients of the currently tracked satellite's position, and DOOM module is called by RUIN to validate user credentials, determine visible objects, and retrieve and store information about reservations.

2.2.2.3 BOSS and MADAM

As the name implies, BOSS is the module in charge. It creates and handles all the instances of all the other modules. All communication between the other modules goes via BOSS to ensure that there only is one instance of a module, thereby creating a virtual Singleton design pattern. In addition, BOSS is intended to have several maintenance tasks such as downloading and parsing TLE files, maintain the database and so forth. Also, reading and parsing the configuration file is the responsibility of BOSS.

The MADAM () is the heart of the satellite tracking. This is where all the calculations are done and the data is then forwarded to the user via the RUIN and WRACK. Data is also sent to SLAM to move the telescope to point at the satellite. It is composed of three threads:

- TelescopeController Thread. As the name implies, this thread is responsible for sending freshly calculated azimuth and elevation for the satellite currently being tracked. The thread uses a simplified set of calculations since only azimuth and elevation is needed. SLAMs `receiveCoords`() method is then called with azimuth and elevation. As soon as `receiveCoords`() returns (after the telescope is done moving), the calculations are repeated. This goes on continuously.
- RUITelemetry Thread. This thread is responsible for supplying the user with accurate telemetry data. That means that it is a continuous thread that calculates a full set of data, azimuth, elevation, range, longitude, latitude, altitude, velocity, orbit number, visibility score, SLAM error number, calculation model (SDP4, SGP4 - SDP8 and SGP8 future options)), footprint1). Visibility status (visible, in sunlight, eclipsed), sun azimuth and sun elevation will most likely be added to the telemetry object sometime in the future. The RUITelemetry thread will call the `outputTelemetry()` method in the RUIN at the end of the iteration.

- Prediction Thread. The prediction thread will only run every 24 hours or so (specific times can be entered in the configuration file). It is recommend that this is scheduled for times when the server has a light load, as it may require significant resources and time depending on the number of satellites in the database. It will loop through all the satellite in the database, and try to calculate when each of these satellites will be visible during the next 7 (can be changed in the configuration file) days. When a visible pass (required score can be set in the configuration file) is found, `DOOMs updatePredictions`() method is called with `SatelliteID`, `OrbitNumber`, `AOSTime` and `LOSTime` as the parameters.

For all three threads there is a common set of calculations that calculate azimuth, elevation and range. The specifics of the calculations depend on whether or not the satellite is considered a deep space satellite. A deep space satellite is defined as a satellite with a period (time of one orbit) of more than 225 minutes. First, the satellites position (in a Earth-Centered Inertial (ECI) coordinate system) is calculated based on the data given in in TLE files. These calculations are quite complex and are explained in detail in [11]. Second, the ECI position of the ground station (where the telescope is located) is calculated with special care taken to adjust as the Earth is not a perfect sphere [11]. Third, the difference between these to ECI coordinate sets is used to calculate azimuth, elevation and range [11]. This is needed to determine if the satellite could be visible. MADAMs implementation is based on the one used by PREDICT, but a description of the mathematics used to determine this can be found at [12].

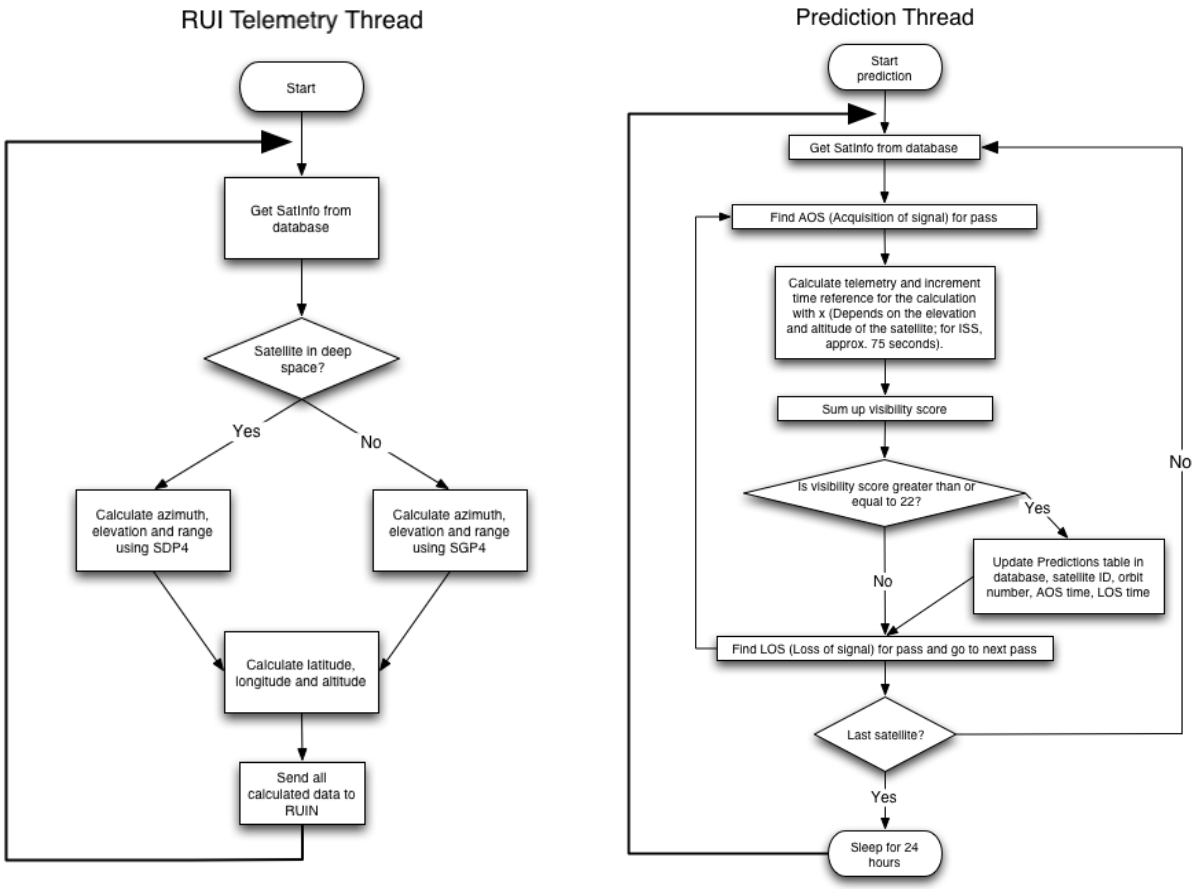


Figure 8. Telemetry and Prediction Threads.

2.2.2.4 DOOM Design

DOOM, which stands for Database Object Operations Manager, is a passive class designed to respond to requests for database access. It operates on a number of fields in an SQL database, which are illustrated in Figure 9. Specifics of each field are omitted due to report space limitations.

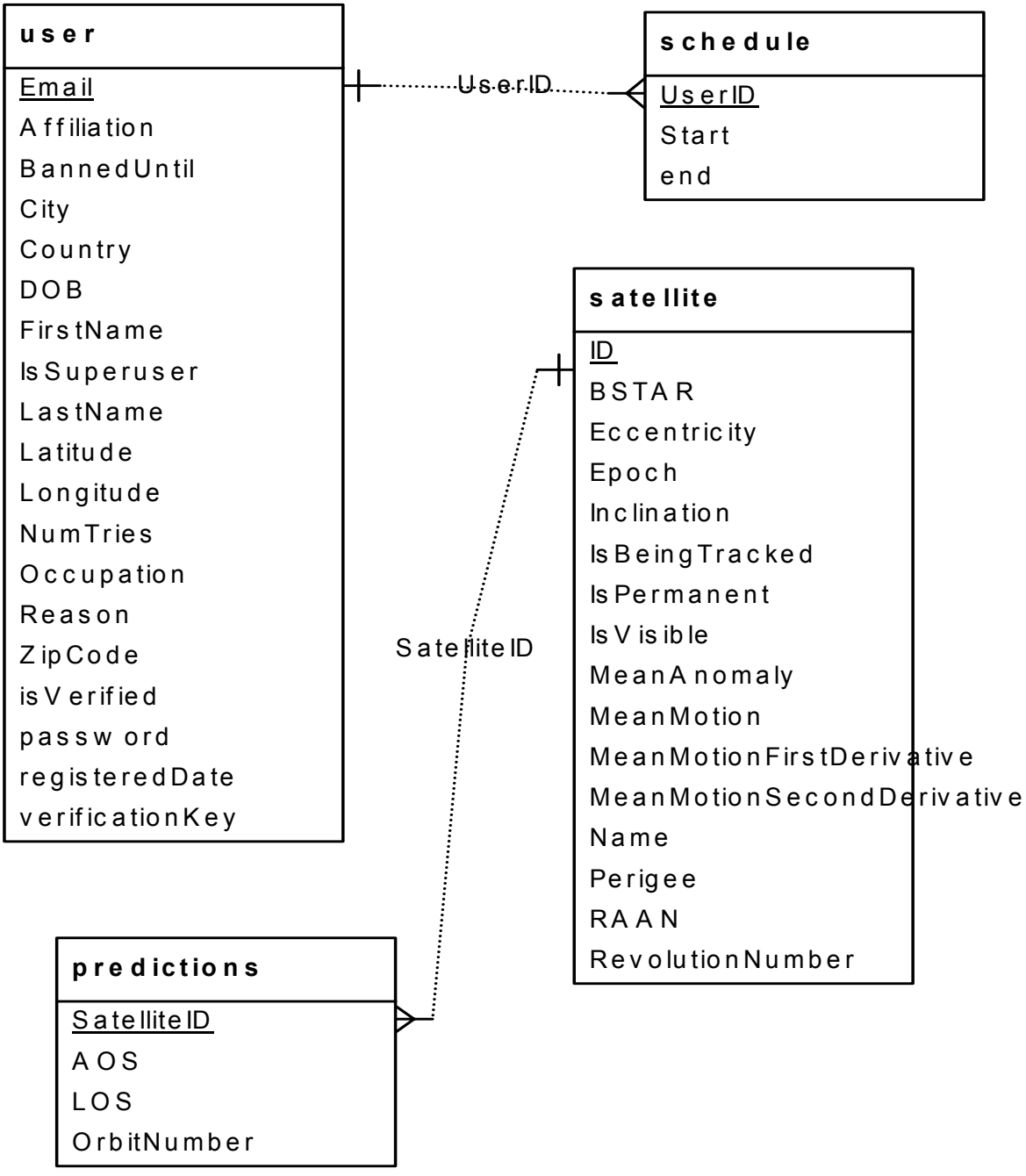


Figure 9. Entity Relationship Diagram for DOOM.

2.2.2.5 SLAM Design

The SLAM module provides serial communications with the telescope to and from the server. This is the module, which allows the server and indirectly, the user, to control and operate the telescope. SLAM uses the standard DB9 pin serial (232) port on the computer to link with the telescope and issues the commands in A-Synchronous mode. The module accepts input from the Telescope Control module (MADAM). These inputs are the current azimuth and elevation and the predicted azimuth and elevation after 5 seconds and the current Right Ascension and Declination of the satellite. The module is composed of two classes: Slam and PortOperations. A brief description of module's functionality provided below.

```
public void recieveCoords(double azimuth, double elevation)
```

Whenever a new object of the Slam class is created, it performs the following actions in the respective order. The actions it takes determine if any action is needed and if so, what action and then preforms that action. Partial functions include the following:

- Saves the time at invocation in member variable 'startTime'.
- Calls function ChangeState with no arguments, changing variable 'state' to end the current thread moving the telescope if there is one.
- Calls function GetTelecsopePosition with no arguments, stores the values in TeleAz and TeleEl
- Determines if the telescope is already moving and or not and if the module is active or not and moves the telescope if it is not moving is it is active.
- Calls function SendCommand with 1 argument which is the command for current telescope Azimuth.

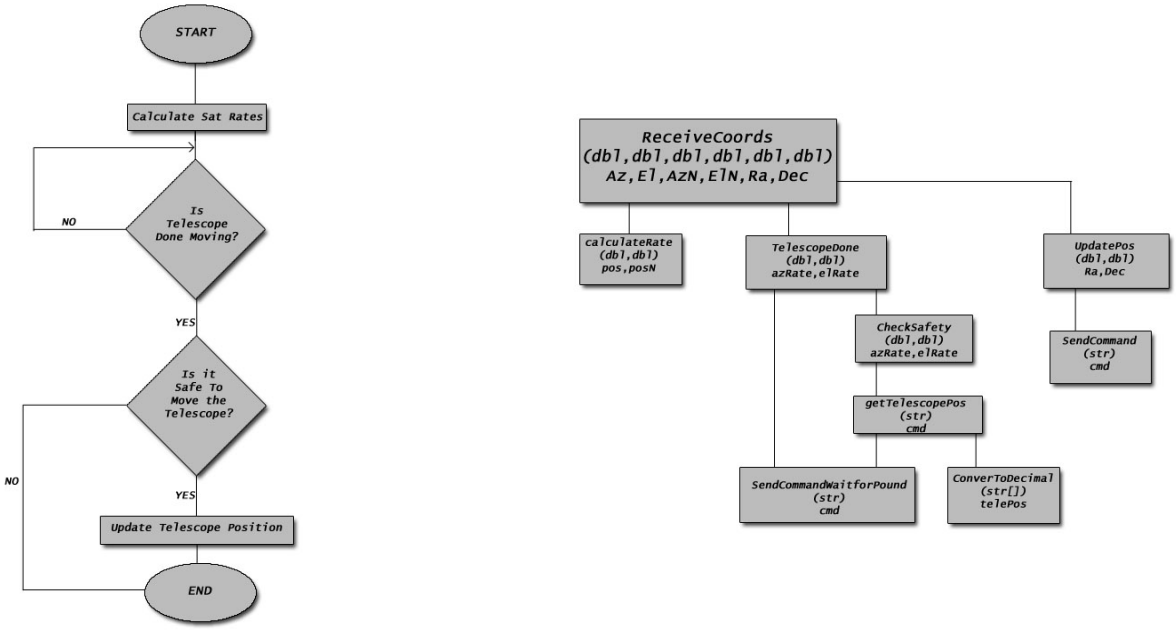


Figure 10. Details of SLAM Design (basic flowchart and structure chart).

2.2.3 Major Design Decisions and Optimizations

There were several major design decisions taken during the course of this project, which allowed for various optimizations. They are listed in Table 3, along with the criteria taken into account.

Table 3. Design Decisions and Optimization.

Design Decision	Criterion	Disadvantage
Use open source software (Linux, Java, MySQL, UML)	Portability	Java sacrifies speed and ease of access to hardware. Linux tends to cause difficulty in configuring. UML notation and tools are an overkill for simpler system.
Follow the principle of separation of concerns	Modularity	May cause undesired over-simplification for bigger systems.
Apply quality assurance at each development stage	Various: SRS - clarity, completeness, correctness, consistency, traceability, testability; Design – cohesion, coupling; Code – readability, structuredness.	Extremely time consuming, but very productive.
New Text-based Internet Protocol	Ease of implementation	Yet another protocol.
New (Virtual Singleton) Design Pattern for Main Module Implementation	Simplicity	Less robust than regular Singleton Pattern when reused (full implementation would require a `friend` feature.
Use of UDC Time	Accuracy and simplicity	Leads to confusion within the project
Manually manage version control (as opposed to CVS)	Simplifies development	Causes confusion, especially during module integration and testing.

2.2.4 Tools Developed

There were four new tools developed in this project, where a tool is understood as a vehicle assisting in performing a primary software function during development and operation. They are listed in Table 4 and the details can be found on project website [10].

Table 4. Description of Four Tools Developed in This Project.

Tool Name	Function
Virtual Singleton Design Pattern	Extension of a Singleton pattern to make it thread-safe and enable parameter passing.
Remote Interaction Protocol (RIP)	Meta-protocol - an additional transport stream than can layer atop TCP to provide increased convenience in implementing basic networking functionality.
Database Backup Tool	Perl script that connects to the MySQL database using the perl DBI interface to cycle through all the database and dump its contents.
Java Serial Link Tool	Communicates with the serial port devices allowing issuing all commands it receives to the serial device as strings.

2.2.5 Software Testing

Testing for this project has been accomplished in the following phases:

- preparation of test plans for each module, addressing specific requirements
- functional testing of each module, according to requirements 1-23
- safety and security testing, according to requirements 24-29, and
- performance testing, according to requirements 30-31.

For the lack of space, in this report we are only including results of performance testing. Results of other tests are made available from the project website [10].

2.2.5.1 Testing of MADAM Computational Accuracy

An extensive test was conducted on MADAM consisted of collecting a number of predictions from SAT-NET and comparing it to the baseline program PREDICT [2]. The latest available TLE file from `http://www.qsl.net/kd2bd/predict.tle` was used for both programs. The predictions were set to start at UNIX time 1082426819 (Tue Apr 20 02:06:59 UTC 2004) and calculate for every 5 minutes for 24 hours. The results were written to text files for later analysis. This analysis proved to be very good news as the only differences seem to be caused by slight differences in rounding. The largest difference we saw, was for the range. In some cases this was off by 0.01 km (10 m) and as mentioned, is most likely caused by rounding. Either way, when the range is in excess of 40,000 km, 10 meters is not bad. Due to the amount of data collected, we only present here graphs for azimuth and latitude, which show typical behavioral patterns [2].

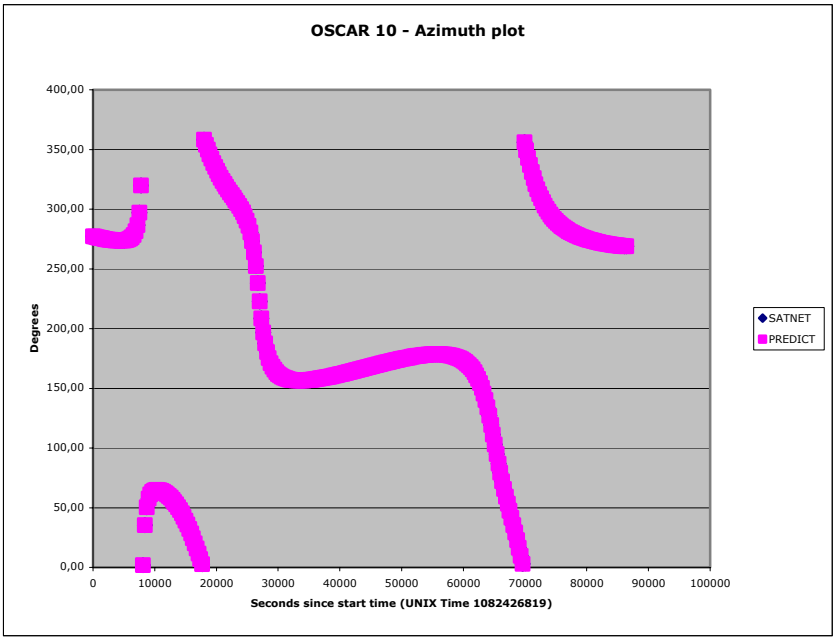


Figure 11. Azimuth Test for SATNET compared to PREDICT.

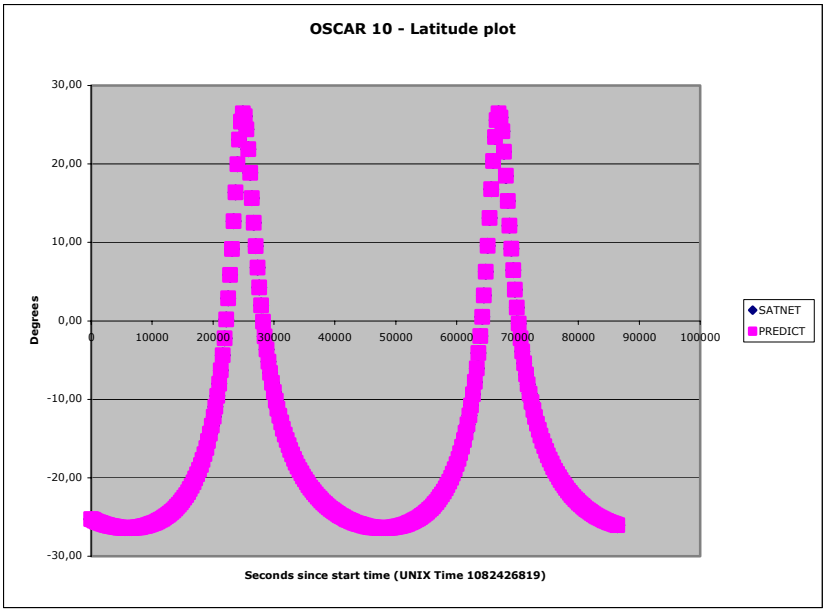


Figure 12. Latitude Test for SATNET compared to PREDICT.

2.2.5.2 Testing Robustness of the Server (RUIN/DOOM Combination)

The second performance test focused on testing the speed of response (latency) in case of increasing load due to access from multiple numbers of client. Multiple calls to RUIN were executed through its RIP protocol, emulating the actual SAT-NET events. The time, in milliseconds, was recorded it takes RUIN to call DOOM and DOOM to return the results back. This procedure was repeated for increasing number of clients (from 1 to 10), all making simultaneous requests. Additionally, artificial resource consumption on the server was added to test the effect of spurious activities. The resulting graph is shown in Figure 13.

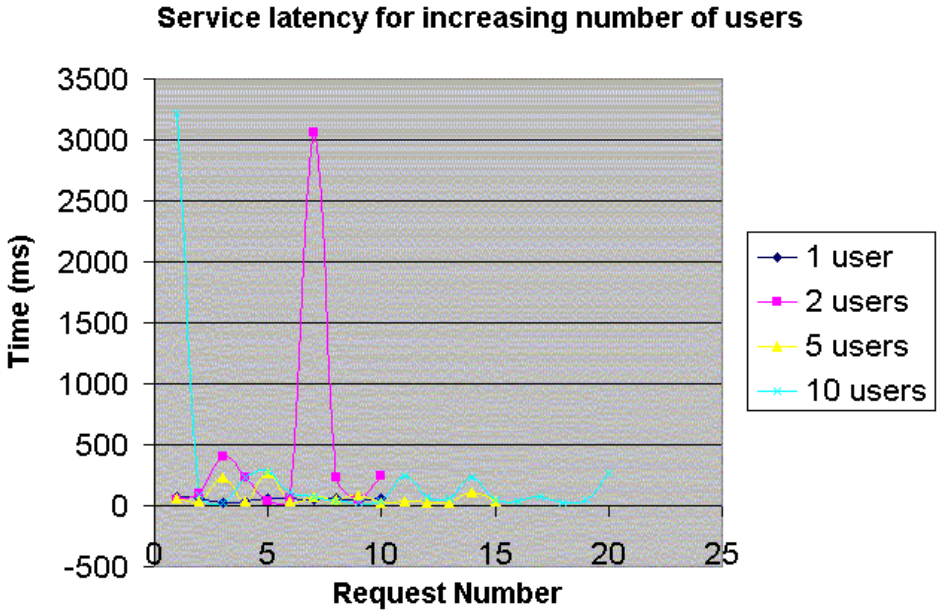


Figure 13. Testing SATNET Robustness: Service Latency vs Number of Users.

3. Summary

The project was completed successfully and the *beta* version has been released. It can be accessed from satnet.fgcu.edu and by making prior email arrangement with team members. Full testing is still being pursued and was conducted successfully during the FGCU Research Day Student Poster Competition, where the Team won first price (`www.fgcu.edu/orsp/`). Further work is needed to fix minor bugs and coordinate operation of the webcam with telescope control software.

We consider the outcome of this project to be important in three aspects: technical, societal, and personal for team members:

Technical. The primary goal of this project was the production of a functioning implementation of the SAT-NET system. Specifically, we intended to produce a web site at which users may remotely control FGCU's telescope and observe the results. Our intention is to leave this system in place, allowing for global access to this important educational resource.

Societal. Secondly, we aimed to open-source the SAT-NET software so that it is can be easily downloaded and operated by any organization with the appropriate hardware. SAT-NET's wide adoption would represent a significantly broader realization of its potential societal benefits, especially in improving safety of the air space and in education of the younger generation.

Personal. Thirdly, we hope to win the competition and not only further educational and research possibilities but increase our own understanding of software development, engineering teamwork, and professional challenges.

While being technically innovative, the SAT-NET project will benefit society in several different ways. In addition to its primary purpose, which is enhancing safety of air space, it will provide global access for non-scientists to scientific experiments over the Internet, as well as address the issue of technological underdevelopment in countries without the means to provide such educational and scientific tools to the population. In particular, the project will:

1. Provide students at all levels (Intermediate School, High School, College Students, etc.) with access to experimentation using scientific equipment, such as a telescope, which will result in popularization of Science.
2. Reduce the cost of science education by allowing expensive scientific equipment to be shared by schools, rather than requiring each school to purchase the equipment or go without the resource.
3. Allow for an exchange of knowledge and research, which extends throughout the globe to countries of minimal industrial development and world industrial powers alike.

Project costs are divided into Development Cost, which is incurred only during development, and Implementation Cost, which is related to every implementation. They did not exceed the limit set by rules of this Competition.

Development Cost	
Satellite Tracking Software for Research	$ 50.00
Null Modem Cable	$ 15.00
Total Development Cost	**$ 65.00**
Implementation Cost	
Serial Cable For Connection with Telescope	$ 26.00
Web Camera for Telescope	$150.00
Total Implementation Cost	**$176.00**
Total Cost	$241.00

4. References

[1] Kidder T., *The Soul of a New Machine*, Little Brown, Boston, Mass., 1981
[2] Magliacone J.A., Tracking Satellites with PREDICT, *Linux J.*, No. 75, July 2000.
[3] Steidler-Dennison T., Linux, Talon and Astronomy, *Linux J.*, No. 117, January 2004.
[4] Boshart B., *Satellite Tracker*, Atwood, Ontario, 2003, http://www.heavenscape.com
[5] SatTrack: Real-Time Orbit Simulation Program, http://www.amsat.org/amsat/
[6] Fauerbach M., PhD, College of Arts and Sciences, Florida Gulf Coast University, Ft. Myers, Florida, January 2004
[7] Johnson R., PhD, Florida Space Institute, UCF, Orlando, Fla., February 2004
[8] IEEE Software Engineering Standards Collection: 2003 Edition. CD-ROM. IEEE, New York, 2003.
[9] Meade Instruments Corporation, Irvine, Calif., 2004, http://www.meade.com
[10] SAT-NET Project Website, Florida Gulf Coast University, Ft. Myers, Florida, 2004, http://itech.fgcu.edu/faculty/zalewski/COP4931/COP4931projects.html
[11] Celestrak – Center for Space Standards and Innovation, Malvern, Penn., http://www.celestrak.com/NORAD/documentation/spacetrk.pdf
[12] Rusin D., Mathematical Sciences, Northern Illinois University, DeKalb, Illinois, http://www.math.niu.edu/~rusin/

Appendix: Definitions & Acronyms

Authentication – the process whereby a client is verified to represent a registered user and thus upgraded to a logged-in client.
Client – the remote software used to interface with the SAT-NET server via the internet. *Note.* This is not necessarily the official SAT-NET Client software; SAT-NET is designed to support alternate clients so long as they support the protocol.
Coordinates – latitude, longitude, elevation (from Horizon), azimuth, altitude and velocity.
Footprint - the geographic region on the earth underneath a satellite, which is in the appropriate range to receive that satellite's information.
Keplerian Elements – a coordinate set derived from Kepler's equations for celestial bodies.
Logged-In Client – a client which has completed the authentication procedure, verifying that the current operator of the client corresponds to a registered user.
NTP – Network Time Protocol
Priority User – a user who is allowed to control the telescope and has assigned priority to do so.
Regular User – a user who can control the telescope only if there are no priority users currently controlling it.
RUI – Remote User Interface
SAT-NET – Satellite Acquisition & Tracking with Network Enabled Telescope
SDP – Simplified Deep Space Perturbations
SGP – Simplified General Perturbations
Superuser – a user in charge of administrative functions for the Software.
TLE – two line element.
User – superuser, priority user, regular user or viewer.
Valid Email - An address of the form `username@domain.ext` which is capable of receiving and responding to emails.
Viewer – a user who is allowed to receive images and data from Satellite Tracker, but not allowed to control the telescope.