A FRAMEWORK FOR DEVELOPING ARTIFICIAL INTELLIGENCE FOR AUTONOMOUS SATELLITE OPERATIONS

Jason L. Anderson[1], **Dr. Franz J. Kurfess**[2], **and Dr. Jordi Puig-Suari**[3]

[1]*California Polytechnic State University, 1 Grand Ave, San Luis Obispo, USA, jander06@calpoly.edu*
[2]*California Polytechnic State University, 1 Grand Ave, San Luis Obispo, USA, fkurfess@calpoly.edu*
[3]*California Polytechnic State University, 1 Grand Ave, San Luis Obispo, USA, jpuigsua@calpoly.edu*

ABSTRACT

In the world of educational satellites, student teams manually conduct operations daily. Educational satellites typically travel in a Low Earth Orbit allowing communication for approximately thirty minutes each day. Manual operations during these times is manageable for student teams as the required manpower is minimal. The international Global Educational Network for Satellite Operations (GENSO), however, promises satellite contact upwards of sixteen hours per day by connecting earth stations globally through the Internet. This large increase in satellite communication time makes manual student operations unreasonable and alternatives must be explored. This paper introduces a framework to conduct autonomous satellite operations using different AI methodologies. This paper additionally demonstrates the framework's usability by introducing a sample rule-based implementation for Cal Poly's CubeSat, CP3.

Key words: Autonomous/Automated Operations; Lights Out Operations; Earth Station.

1. INTRODUCTION

There are many different operational satellites in orbit at the moment with missions ranging from scientific payloads provided by NASA's Jet Propulsion Laboratory (JPL) [Lab09b] to commercial communication missions such as Direct TV services [TV09]. Each of these satellites, however, requires an operations team to monitor the spacecraft and solve potential problems. These operations are well understood, repetitive tasks making spacecraft operations a perfect candidate for automation [HLS96].

Satellite operations can also be expensive to maintain for any sustained period of time. NASA operation centers are typically staffed 24 hours a day, 7 days a week which adds up over time [HLS96]. For instance, LandSat 7 requires approximately $20 million per year for operations [Tho03]. If automation made it possible to reduce this budget by even 5% ($1 million), the direct savings alone would be enough to adopt an automated system.

In order to provide students with the necessary skills to work in the Aerospace industry, Stanford in coordination with the California Polytechnic State University (Cal Poly) has developed the CubeSat standard [Chi09]. CubeSats are small $10cm^3$ satellites weighing less than a kilogram [Too07]. The concept is that small satellites can be developed in approximately 2 years, allowing students to be involved in the design, development, testing and operations of a complete spacecraft. There are currently over 30 CubeSats in orbit at various mission stages [KAL09].

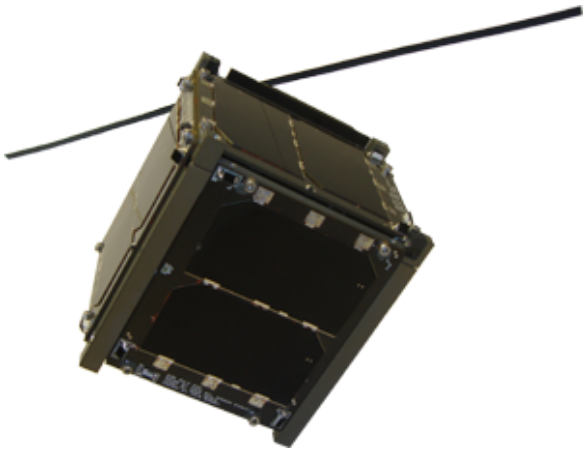

Figure 1. CP3, Cal Poly's Third CubeSat

Unlike some NASA missions which have near constant contact with their spacecraft, CubeSat orbits are such that only 30 minutes of contact is available per day [Hue06]. CubeSat operations therefore require a small amount of manpower and currently does not warrant a fully automated system as CubeSat teams often have 10 or more members. A simple rotation schedule is enough to ensure that all satellite passes are utilized.

While the current CubeSat operations situation does not require automated operations, the Global Educational Network for Satellite Operations (GENSO) will soon

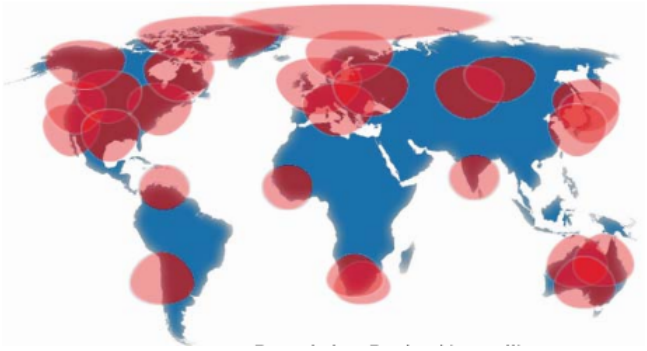

Figure 2. LEO Ground Coverage Using the GENSO Network

greatly increase the potential operations time [For09]. GENSO is a project which promises increased educational satellite (ie. CubeSats) communication time by connecting earth stations all over the world through the Internet [SK07]. For example, when Cal Poly's CP3 is within communication range of the University of Aalborg, Denmark's earth station, Cal Poly can command CP3 via the Internet through Aalborg's station. An important point is that earth station sharing is bidirectional so that when Aalborg's satellite AAUSat-II is within communication range of Cal Poly, Aalborg can command their satellite using Cal Poly's earth station.

When GENSO is completed, it will increase satellite communication time from 30 minutes per day to approximately 16 hours per day which will proportionally raise the amount of manpower required for operations by 32 times. Since this substantial increase cannot be feasibly compensated by adding more student labor, another solution, automation, must be explored for educational satellites.

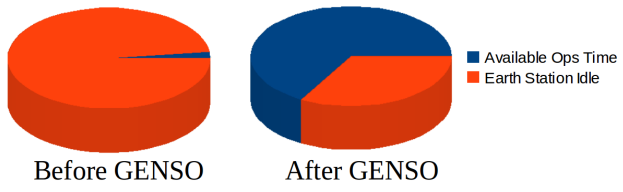

Figure 3. Time Available per Day to Conduct Ops Before (left) and After (right) GENSO

There have been many spacecraft automation systems developed over the years but none specifically for CubeSat satellites. In order to have the flexibly to quickly develop different AIs for automation, an easy to use framework is required. In order to develop a framework for conducting autonomous spacecraft operations, knowledge possessed originally by a human operator needs to be encoded into the computer. One method of accomplishing this is using a rule based system. This paper's implementation encodes the information required to operate Cal Poly's CP3 satellite into JESS (Java Expert System Shell). The type of information encoded includes what low level commands are needed to accomplish high level satellite tasks, expected satellite responses as well as actions to correct unexpected satellite behavior. The last enables the system to recover from errors which occur onboard the satellite.

This implementation also explores the possibilities of a learning knowledge base. That is when the earth station receives an unexpected response which it does not know how to handle, it tries to determine the most probable solutions and executes them. Once the problem is believed to be resolved, the system then resumes and resends the originally requested command. It is important to note that the knowledge base knows what is unsafe for the satellite and never executes commands which are potential harmful.

Autonomous spacecraft operations enable earth stations to operate spacecraft with minimal human intervention. The autonomous spacecraft operations field has greatly progressed over the last fifty years but has yet to be fully realized.

2. LITERATURE REVIEW

One of the first attempts to increase autonomy and reduce the manpower required for operations is NASA Goddard's Generic Spacecraft Analyst Assistant (GenSAA) [HL91]. GenSAA is an expert system that advises human operators about potential spacecraft issues and probable causes/solutions [LK03]. The system is written using a rule-based system called CLIPS [Ril09]. GenSAA's novel idea is to abstract the rule-based system behind an intuitive GUI so that domain experts can graphically create the rule base by defining the satellite's model and specifying restrictions.

The next step in automation came once again from NASA Goddard in the form of the Generic Inferential Executor (GENIE). GENIE is a tool which allows individuals to easily create pass script templates to encode the tasks necessary to mimic a flight operations team [HLS96]. Using a similar graphics library to GenSAA's, GENIE continuously shows the user the pass script's current execution step.

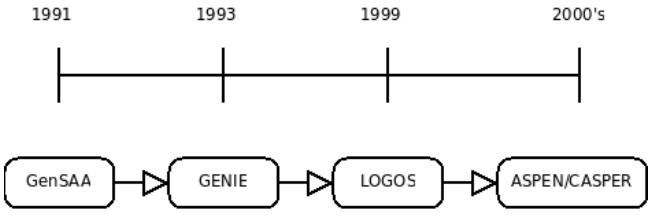

Figure 4. Timeline of Relevant Automated Operation Systems

Another innovative pass automation system is NASA Goddard's Lights Out Ground Operations System (LOGOS) which is a more direct attempt at reducing manpower through the use of software agents to automate all human processes for operating a spacecraft [THK99].

These ground-based agents work in coordination with onboard agents to complete the autonomy. The agents representing different functionality were kept separate rather than combined into one large agent in the hope of discovering emergent behavior. LOGOS utilizes the power of GenSAA/GENIE in a single agent whose sole job is to determine the next command to send and handle any possible anomalies.

The final system which automates satellite operations is NASA JPL's Automated Scheduling and Planning Environment (ASPEN). ASPEN's primary role is to accept a number of high-level goals and generate a set of low-level activities that satisfy the goal [CRK+00]. To allow a user expressive control of the spacecraft model, an improved modeling language was introduced. The modeling language is more expressive than GenSAA/GENIE's and includes constraints such as parameter dependency.

Additionally, ASPEN consists of a real-time component called the Continuous Activity Scheduling Planning Execution and Replanning (CASPER). CASPER is responsible for modifying the current activity plan as time progresses based on events which occur after the initial plan has been created. For instance, CASPER was used on the spacecraft Earth Orbiter 1 (EO-1) to allow the satellite to change its current task if another task becomes more beneficial to execute at that moment [CST+03]. For example when EO-1 detects a rare phenomenon occurring such as a volcano eruption, EO-1 will suspend its current task and focus on the volcano eruption. This ability to tweak generated plans has been key to EO-1 success.

While there have only been a few successful ground-based automation systems created, this research has been minimized due to an increased focus on onboard autonomy. There are many benefits for having autonomy onboard the spacecraft such as quick mission adaptability and higher quality scientific data [LJO+]. The only reason for earth station automation is to reduce the required operations manpower [EFS+04] and with NASA's large budget, earth station operators are cheap compared to the increased risk introduced by fully automating operations.

3. FRAMEWORK DESIGN

In order to provide an extensible framework which can be used to quickly develop automated operation systems, the Autonomous Spacecraft Operations Framework is implemented in Java and consists of a few modular components. These components are the Agent, the Knowledge Base, the Task File, the Terminal Node Controller (TNC) and the Line of Sight Executive. These components are individually described in this section.

3.1. The Agent

For all space operations, a certain set of functionality is always required. These actions include data logging and spacecraft communication. This core set of required functionality is contained within in the Agent. The main functions provided are

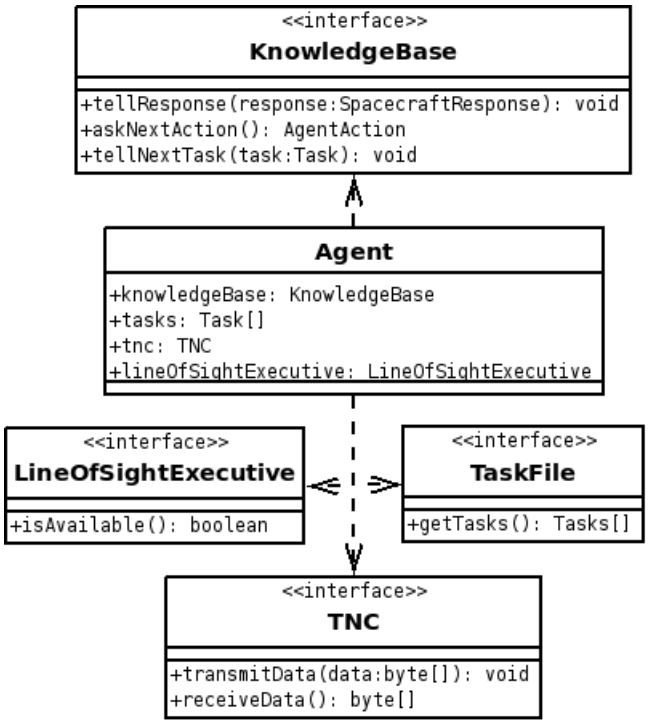

Figure 5. The Overview Framework Design

1. Transmit and log uplink packets,
2. Receive and log downlink packets,
3. Read a task list from a file.

The agent can also be further extended via Java's inheritance to incorporate additional features.

3.2. The Knowledge Base Interface

All of the intelligence for spacecraft operations is located in the Knowledge Base. This is the primary component to extend for a given spacecraft to implement different autonomous systems. The following functions provide the interface between the Knowledge Base and the Agent (See Standard Program Execution for function usage).

1. **`tellNextTask`**`(Task) : void`

   Tells the Knowledge Base which spacecraft Task should be completed next as listed in the Task File.

2. **`askNextAction`**`(void) : AgentAction`

   Queries the Knowledge Base for the next agent action to execute (ie. next command to send).

3. **tellResponse**(SC_Response) : void

   Tells the Knowledge Base what response was returned by the satellite (can be a timeout response).

Currently, a Knowledge Base must be implemented for every satellite the framework will track. That is if the framework is going to track both CP3 and CP6 (which have very similar commands), there needs to be two different Knowledge Bases. The Knowledge Base however is written in Java and therefore can be constructed using polymorphism to reuse code between different satellites of the same CPX brand.

3.3. The Task File

The spacecraft manager is able to define what tasks to complete over a period of time using the Task File. This Task File is currently structured with each line representing a single task followed by a parameter list. The following represents an example Task File.

```
DumpCDHData()
RunADCSExperiment(1, "MagTorque1.test")
DumpADCSExpData("MagTorque1.test")
RunADCSExperiment(2, "ExpTemps1.test")
DumpADCSExpData("ExpTemps1.test")
```

3.4. The TNC Interface

Since most spacecraft communicate over a radio link, operations software must be able to modulate/demodulate the digital data sent between the spacecraft and the earth station. The modulating and demodulating of data is commonly done using a terminal node controller (TNC). A TNC can be a physical device such as the KPC9612+ [Kan09] or implemented in software with a program such as MixW [Fed09]. Since each earth station is different, a TNC driver must be created by implementing the Java TNC interface. A default Serial TNC class is provided since most hardware and software TNCs are implemented using this method of communication.

3.5. Line Of Sight Executive

While external tracking software can control the Doppler shift of the spacecraft's frequency and control the pointing of the earth station's antennas, the agent must still know when the spacecraft is available for communication. The Line of Sight Executive encapsulates this information in an interface which can be implemented by any user defined class. A SGP4 implementation is provided by default [Joh09].

3.6. Standard Program Execution

When the agent starts, it parses the entire input task file into a serializable in-memory structure. The agent then begins its basic control loop which is represented in the following pseudo code.

Listing 1. The Agent's Algorithm

```
1  For each task t in input file do:
2    Tell Knowledge Base t via tellNextTask(t)
3    Ask  Knowledge Base next action a
4
5    While a in not null do:
6      Execute a
7      Tell Knowledge Base execution results
8      Ask Knowledge Base next action a
```

This sequence is executed until all the tasks in the input file have been completed. Once the agent has finished all the tasks found in the Task File, the agent reads a similarly formatted default Tasks File which defines what the agent should do once all of its tasks have been completed. This functionality is useful since most CubeSat missions would not prefer an idle spacecraft. At the very least, health and status data can be downlinked for later analysis.

4. SAMPLE RULE BASED IMPLEMENTATION

All of the previous automated operations research has been built using rule based systems (RBS). In order to provide a baseline for this framework, a RBS is built and integrated with the framework. The RBS is developed using the Java Expert System Shell (JESS) [Lab09a].

4.1. Satellite Model

In order for the RBS to make good operation decisions, it must always maintain a believed model of the satellite's state. To do this, a Java class is used to store the last satellite snapshot and the time it was taken. This class can then be queried for questions given the current time. For instance, if the RBS wants to know if the satellite is in Normal Ops, it would call

```
satModel.isInNormalOps(
   System.currentTimeMillis())
```

which returns true if the satellite is believed to be in Normal Ops and false otherwise. This model is updated every time a command is issued or a response is received. The following fields represent some of the common parameters monitored for the CPX brand of satellites.

1. Snapshot Time

2. Time Left in Normal Operations
3. Battery Voltages
4. Temperatures

An example of where these parameters are useful is when a payload command needs to be sent. All CPX satellites have the concept of Normal Operations (Ops) which is the high power state required for payload command execution. Normal Ops must be enabled by sending a `Go To Normal Ops` command. For safety reasons once the satellite is in Normal Ops, the satellite has a 3 day inactivity timeout which will put the satellite back into Pre Ops. Therefore, it is necessary to query the model before sending a payload command to make sure that the satellite is already in Normal Ops.

4.2. Task to Command List Rules

One of the simple uses for JESS is to translate a task from the Task File into its corresponding satellite commands. These JESS rules are straight forward to implement. The following is an example JESS rule used to translate the `DumpCDHData` task into its corresponding commands.

```
defrule DumpCDHData
  ?x <- (Task (TaskName DumpCDHData) (args ?))
  (not (lowVoltage))
  => (retract ?x)
  (assert (Cmd (pos 0) (CmdName ``12'')))
  (assert (Cmd (pos 1) (CmdName ``34'')))
```

4.3. Built-In Satellite Safety

In order to protect the satellite from executing any power intensive commands while its batteries are low, the RBS has implemented ground-based safety checks. That is when the Task List specifies a task which requires a power intensive command, the RBS checks via its rules to verify that the satellite can successfully complete the command without browning out the satellite. If there is not enough battery power, the RBS waits for 30 seconds and then sends a status command to see if the satellite has charged to a safe battery level.

4.4. Error Recovery

The previously developed systems implement error recovery as a basic fault decision tree. Similarly, the RBS identifies any NACKs it receives and then analyzes the model to determine what could be wrong. Once the RBS believes it knows the most likely problem, it sends the correct commands to resolve the issue. The RBS then resumes the command sequence.

The problem with this method of problem resolution is its inability to adapt to new problems. For instance if the RBS resends the original command and another NACK is received, the RBS will cycle continuously never solving the problem correctly. To add learning functionality, the decision tree algorithm would have to be adapted to randomly create new methods for solving a problem and test to see if the new solutions are effective. More research must be done to determine if rule based error recovery with learning is possible for spacecraft automation systems.

5. RESULTS

The Autonomous Spacecraft Operations Framework has been successfully deployed for use with Cal Poly's CP3 satellite. Most of the initial testing was conducted using a bench model of CP3, communicating across the room via an ICOM 910 H [Rad09] radio using the software TNC MixW [Fed09]. Normal operational modes work as expected as well as error handling which resolves no satellite response errors and Nack responses. While the system was used for real operations with the CP3 unit in orbit, the demonstration was only mildly satisfying due to CP3's Comm problem which keeps Cal Poly from reliably commanding the satellite.

Figure 6. Cal Poly's Earth Station Used For Testing

6. FUTURE WORK

As the autonomous spacecraft operations framework provides an extensible platform for developing AIs to conduct operations, the next step is to develop packages which can be used with the framework. A learning Knowledge Base is of immediate interest since the system would be able to adapt to the idiosyncrasies of each spacecraft. This package could be developed as an extension to the Knowledge Base API by creating a set of Java class which provide empty shells for different methods of learning (ie. genetic algorithms, artificial neural networks [RNC+95]). This work could then be validated by extending the CP3 sample implementation to demonstrate its learning features.

Other nice to have features include operator notification when all tasks have been completed, a web interface to track the progress and current state of the system remotely, and a HamLib [Gro09] wrapper which contains a number of already developed drivers for radio and TNC equipment.

7. CONCLUSION

This paper introduced the autonomous spacecraft operations framework which is a software package that provides basic support for autonomous spacecraft operations. The design philosophy behind this work was explained and a sample implementation was provided for the Cal Poly CubeSat satellite, CP3. This implementation has been thoroughly tested and successfully used with the flight model of CP3 currently in orbit.

REFERENCES

[Chi09] Alex Chin. Cubesat community website. World Wide Web, March 2009.

[CRK+00] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, et al. Aspen–automated planning and scheduling for space mission operations. In *Space Ops*, 2000.

[CST+03] S. Chien, R. Sherwood, D. Tran, R. Castano, B. Cichy, A. Davies, G. Rabideau, N. Tang, M. Burl, D. Mandl, et al. Autonomous Science on the EO-1 Mission. 2003.

[EFS+04] A. Ercolani, P. Ferri, A. Simonic, A. Kowalczyk, and T. Ulriksen. Operations Automation for the Rosetta Mission. In *The Space Operations 2004 Conference, Montreal, Canada*, 2004.

[Fed09] Nick Fedoseev. Mixw - multimode software for radio amateurs. http://www.mixw.net/, 2009.

[For09] Sandra Forsman. Genso. World Wide Web, March 2009.

[Gro09] The Hamlib Group. Ham radio control libraries, 2009.

[HL91] P. M. Hughes and E. C. Luczak. The Generic Spacecraft Analyst Assistant (GenSAA): A Tool for Automating Spacecraft Monitoring with Expert Systems. *NASA Conference Publication*, 3110:129–+, 1991.

[HLS96] J. Hartley, E. Luczak, and D. Stump. Spacecraft control center automation using the Generic Inferential Executor(Genie). In *International Symposium on Space Mission Operations & Ground Data Systems-'SpaceOps 96', 4 th, Munich, Germany*, pages 1007–1014, 1996.

[Hue06] Derek Huerta. Development of a highly integrated communication system for use in low power space applications. Master's thesis, California Polytechnic State University, 2006.

[Joh09] David Johnson. Personal correspondance with david johnson, software developer for black pepper software, 2009.

[KAL09] Bryan Klofas, Jason Anderson, and Kyle Leveque. A survey of cubesat communication systems. *AMSAT Journal*, 2009.

[Kan09] Kantronics. Kantronics kpc-9612+ radio modem/tnc. http://www.kantronics.com/products/kpc9612.html, 2009.

[Lab09a] Sandia National Laboratories. Jess, the rule engine for the javatm platform, 2009.

[Lab09b] Jet Propulsion Laboratory. Jet propulsion laboratory: California institute of technology. World Wide Web, March 2009.

[LJO+] D.B. LaVallee, J. Jacobsohn, C. Olsen, J. Reilly, and MS Starkville. Intelligent Control For Spacecraft Autonomy–An Industry Survey.

[LK03] D. LaVallee and W. Knopf. TIMED lights out operations. In *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, volume 7, 2003.

[Rad09] Universal Radio. Icom 910h amateur multi-mode vhf uhf transceiver ic-910h ux-910. http://www.universal-radio.com/CATALOG/hammulti/1910.html, 2009.

[Ril09] Gary Riley. Clips: A tool for building expert systems, 2009.

[RNC+95] S.J. Russell, P. Norvig, J.F. Canny, J. Malik, and D.D. Edwards. *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, NJ, 1995.

[SK07] Graham Shirville and Bryan Klofas. GENSO: A Global Ground Station Network. In *Proceedings of the AMSAT-NA 21st Space Symposium*, 2007.

[THK99] W. Truszkowski, H. Hallock, and J. Kurien. Agent Technology from a NASA Perspective. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 1–33, 1999.

[Tho03] R.J. Thompson. Correspondance with r.j. thompson, chief of the usgs earth resource observation systems data center, sioux falls, s.d., 2003.

[Too07] Armen Toorian. Redesign of the poly picosatellite orbital deployer for the dnepr launch vehicle. Master's thesis, California Polytechnic State University, 2007.

[TV09] Direct TV. Direct tv: Satellite television. World Wide Web, March 2009.