Using Modular Software to meet Dynamic Customer Demands

Dale P. Massey[1]
Universal Space Network, Horsham, Pa, 19044, USA

Thomas D. Shult[2]
Huntington Beach, Ca, 92647, USA

Universal Space Network (USN) found that most customers wanted one of kind solution for their satellite data needs when utilizing USN's commercial satellite ground network. Each application was not completely unique, but was some variation on a common theme. Time was being spent inefficiently attempting to create similar but different solutions. All of the software had certain things in common, such as custom formats, real-time connections, etc. USN decided to build a system that would provide the necessary flexibility required to satisfy the myriad of dynamic customer requirements. This system is composed of an outer framework and a set of single purpose modules. These modules can be linked and configured, in almost unlimited ways, to perform various tasks. The following are some example uses of this software: split Telemetry (TM) into virtual channels, both real-time and/or non-real-time data delivery, generate Tracking Data Message (TDM) or Universal Tracking Data Format (UTDF) files, and process Tele-commands (TC). This paper will describe the software and the future direction.

Keywords

Modular Applications, Application Framework, Software

I. Introduction

In the early days of the company, USN was working to establish a reputation as a trusted satellite ground services provider. Many of the customers wanted to use the USN sites as if they were an extension of their previously contracted ground stations. This meant that all of the USN interfaces would have to perform identically to the customer's other stations. If they wanted tracking angle data in the format of their particular antenna, USN would have to convert the format from our antenna to their format before putting the data on the line. This resulted in the generation of many unique data conversion applications. This approach was both time consuming and expensive, so a new approach was developed to improve efficiency of implementing new customer interfaces. During the transition from a startup to a well-respected ground station services provider, cost-effective customer data formatting requirements were always a dominant issue. The ideal solution was to develop a modular/configurable system to handle the variety of unique data conversion demands. Following a ranging proof of concept test, the requirement to produce an operational ranging system presented the opportunity to create an Application Framework with multiple single purpose modules that could be linked together to build a customer solution.

II. The Design

The operational ranging system is composed of several smaller tasks, data collection, merging, formatting, delivery and task management. Each of these tasks can be broken down into even smaller tasks. For example, the Data Collection task is composed of receiving, reformatting recording and transmitting data. The Data Collection

[1] Senior Software Engineer, 417 Caredean Drive Suite A, Horsham, Pa 19044.
[2] 6701Mason Drive, Huntington Beach, Ca 92647.

Receiving task, as shown in Figure 1, is composed of basic tasks, such as device I/O, initialization and frame processing, translation and output. A module was created to support each one of these basic tasks. For example, a socket module was created to provide the functionality of the device I/O task, a framing module to support data alignment. Once all the required modules were created, a configuration file integrating all the modules in to a complete task was built.

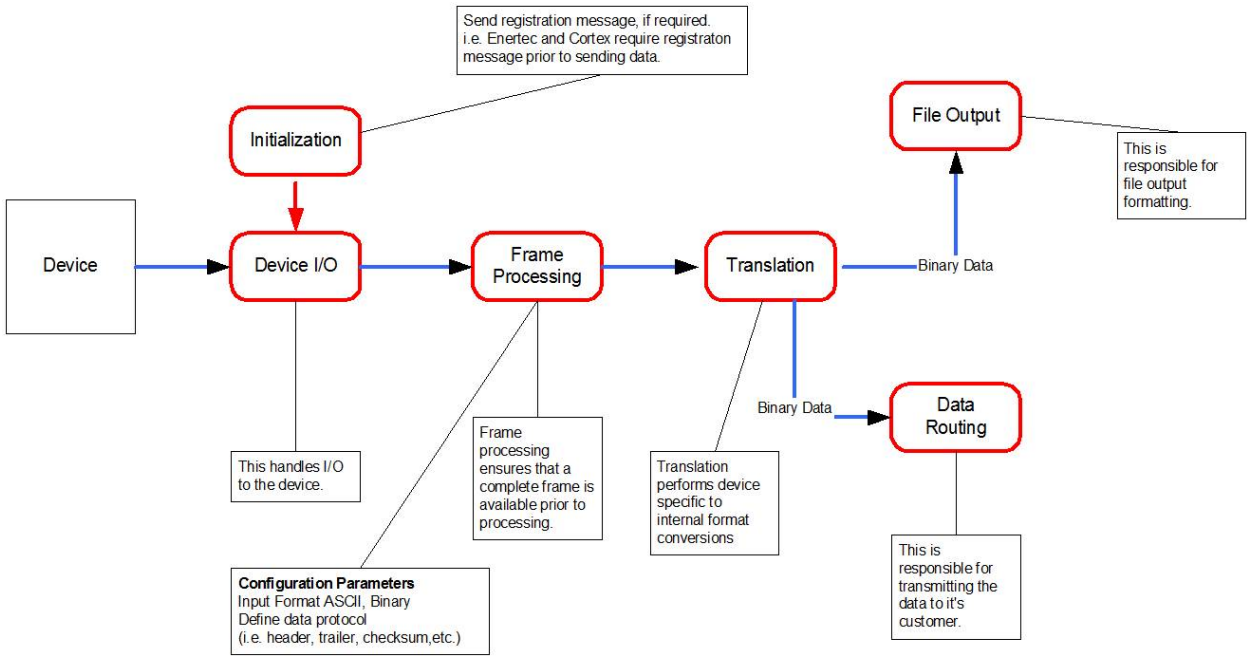

Figure 1 Data Collection Receiving Task

III. Software Overview

The software is a client/server application, consisting of four parts, the framework, the modules, the control GUI, and the configuration GUI. An instance of the framework is continuously running at one or more dedicated machines at each antenna location. The framework consists of a server application and several single purpose modules.

IV. The Framework

The framework, known as "*SNApplication Framework*", is a software application with plug-in modules that allow multiple software solutions to be built by connecting the modules together in different configurations. These configurations only require new software development if a new module is required. A configuration file defines what type of processing occurs. Each configuration file defines the interconnections, time/event based processing and specific configuration for each required module. Each configuration file is divided in to the following sections *Application*, *Load*, *Connections*, *Modules* and the optional section *Module Control*.

A. Application Section

The *Application* section defines general parameters, such as status rate, log output, and application name, associated with the configuration.

B. Load Section

The *Load* section provides the association between the actual module name and the internal module identifier.

C. Connections Section

The *Connections* section specifies the interconnections between modules.

D. Module Control Section

The *Module Control* Section provides timing and synchronization between modules. In some configurations, there are timing issues that must be accomplished, such as writing a header on the eventual output file that contains the total number of records recorded. So the header cannot be written until all data has been recorded.

E. Modules Section

The *Modules* section contains a configuration entry for each loaded module. A module configuration defines general and specific configuration parameters used by the defined module. For example a socket module, has the following configuration parameters, IP address, port, type (i.e. client or server), etc.

The final configured item is the global parameter table. The framework manages and maintains the global parameter table, which contains defined parameters from all modules and the framework. These parameters are available to all the loaded modules. There are times where a module or an outside source needs to feed some data outside of the flow of customer data between modules. This is accomplished through global data. An example would be the total number of frames processed for the TDM header may be recorded by the framing module and used in the header generation.The framework also gathers status from the modules and outputs on a socket, processes incoming commands (e.g. load configuration, unload configuration) on a control port, and logs important data.Those are the main functions of the SNApplication Framework. The next section will discuss the design of the modules.

V.The Modules

Each module is a Microsoft windows dynamic link library file. These files are designed to be single purpose. The motto for modules is, “do one thing and do it well.” Each module has some configuration to allow the program to be as flexible as possible. If a module has to be modified, or a new module is generated, that can be accomplished without changing the framework or the other existing modules.

Each module receives data from one or more input connections, processes, and transmits the data on one or more output connections. Received data can be internal or external to the framework. For example, a socket module can receive data, from an input socket, and transmit it to other connected modules. It can also receive data, from connected modules, and transmit it to an output socket. The modules will feed status and log information back to the framework. They can also read/set global parameters managed by the framework.

VI. Control GUI/Automation

There are two GUI applications for this system, one is the control/monitoring GUI, and the other is a configuration building GUI. The system can also be run in automation mode using commands sent to the control port.

A. Control GUI

The control GUI, known as “SNApplicationGUI”, is the monitor/control application for the system. The GUI can manage multiple framework applications simultaneously. What is displayed (plots, progress bars, data fields, etc.) are controlled by a script for each framework application. There are three main parts to the GUI, the tree showing what is running at each site, the output section that displays important information from each framework application and dock-able pages for each framework application. All control messages can be sent from this GUI to any applicable framework application. Monitoring data from all framework applications is processed and if specified, displayed.

.

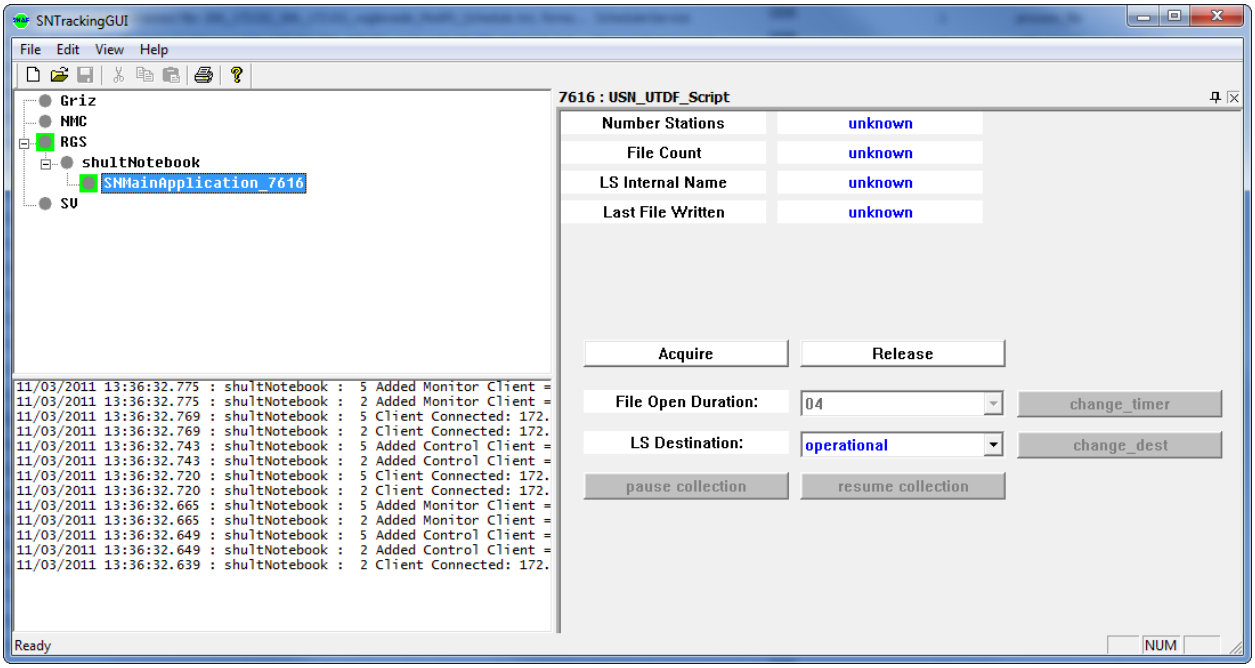

Figure 2 Example SNApplicationGUI screen

B. Configuration GUI

The configuration GUI provides a graphical tool for the creation of configuration files. The configuration files can be several thousand lines long. For example the configuration file, shown in Figure 3, is 2221 lines, manually editing this file can be an arduous task. Therefore this tool is invaluable. In Figure 3, each block represents a module. The lines between blocks represent the module interconnections. In this configuration the data flows from left to right. For example, data is received on the module Angle Socket, Doppler Socket and Range Socket, and following each path until the data is output on the module Output Socket.

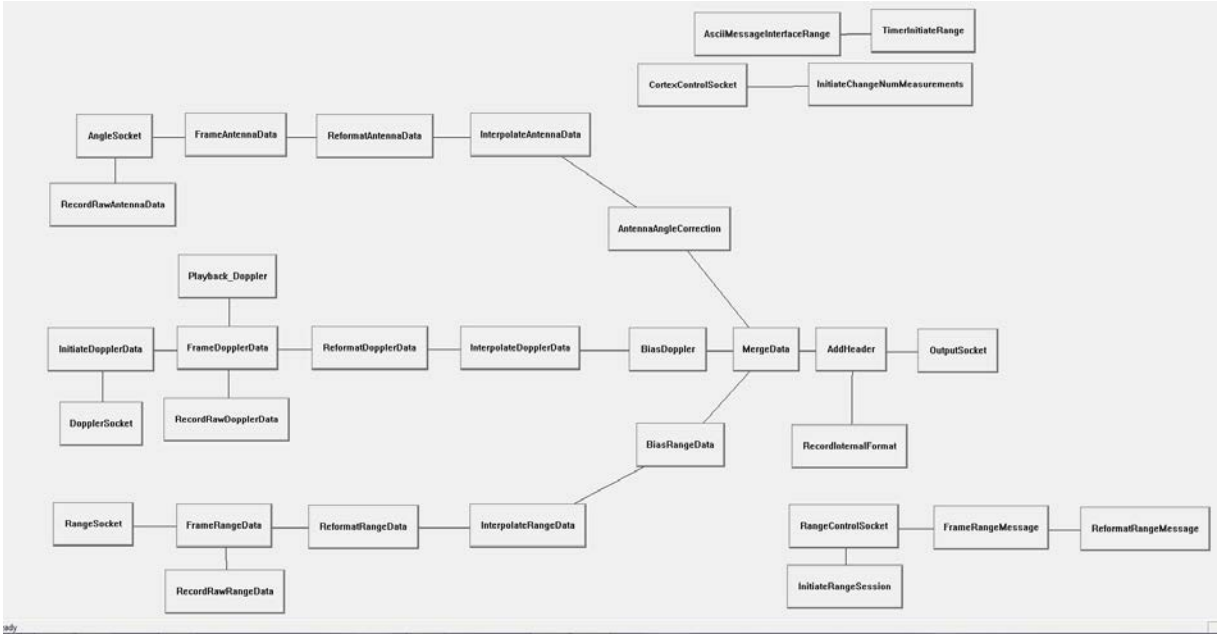

Figure 3 Example Configuration GUI screen

VII. FUTURE DIRECTION

The flexibility of the system allowed a similar affect to the original problem. Many engineers wanted special features to monitor/control for their mission and this was accomplished. Although the effort to do this was greatly reduced, the differences in the GUIs cause the operators confusion. Currently there are two efforts in progress, create an interface to control SNApplicationFramework from external control system, and standardization of mission configurations to reduce training costs and operator errors.

VIII. CONCLUSION

In conclusion, to simplify and create unique configurations for individual customers, USN used a modular framework to allow different applications without developing new software.

Appendix A
Acronym List

CCSDS	Consultative Committee for Space Data Systems
CDH	Command Delivery Header
CORTEX	Command Ranging and Telemetry Unit
GSC	Ground Station Controller
HDR	Cortex High Data Rate Receiver
IPDU	Inter-project Data Unit
NASCOM	NASA Communication System
NMC	Network Management Center
TC	Telecommand
TDM	Tracking Data Message
TM	Telemetry
UTDF	Universal Tracking Data Format

Appendix B
Module List

Here is the current list of modules along with a brief summary for each one.

ASCII Message Module
This module provides an ASCII to Binary and Binary to ASCII translation of internal messages. This is used to allow a module to send or receive control messages from the system.

Cortex Reformat
This module reformats the CORTEX Doppler and Range messages into individual frames. Range data is dumped from the CORTEX in one large frame with multiple readings. This module allows the software to treat each reading as a separate entity.

Current Value Table
This module saves specified parameters from multiple incoming frames from various sources into a current value table. The module can then send out a packet with the latest value for all configured parameters.

Data Checker
This module checks frames of data for the frame counter. It tracks the number of missing frames and the number of gaps in the frame counter. For each gap in the data, the gap and/or the frame counters will be output.
Data Queue
This module manages a queue of frames. It will process fields within frames and save them to a queue based on an optional queue size.

Demultiplexer
This module takes in frames of data (ASCII or binary), de-multiplex the frames to different connections identified by a set identifier. The module uses multiple fields specified in the configuration to determine how to de-multiplex the frames. The output is determined by the values of the fields used in a conditional statement to determine which type of frame is being processed.
File Playback
This module defines a file and a method of playback to replay data files through the system. It can control the rate at which data is played to mimic the original rates.

File Record
This module records all frames received to the specified file.

File System
This module performs actions on the local file system. It includes file and hot directory processing.
Framing
This module aligns the data based upon specified configuration parameters.

General Reformatter
This module allows conversion of the stream of data. It can take in framed ASCII or BINARY and it can output framed ASCII or BINARY. You can create fields based on data from the input stream or from current date/time information. The output can be optionally sent to a global parameter instead of continuing in the stream to the next module. This module works at the byte level. For bit level manipulation there is the Translate module.

Initiate Data
This module allows a pre-defined message to be sent out a set of connections to initiate some data. The received data will then be sent out a different set of connections. The message could have a reply and if so, the reply message will be validated against the configured message.

Interpolate
This module performs linear or parabolic interpolation based upon a key field.

Merge
This module will merge multiple input data streams into one output data stream based on a single configured key field.

Packet Translate Module
This module translates packets from one header type to another header type. This module currently supports CDH, CORTEX, Data Only (no header), IPDU, and NASCOM header. This module could be build with separate modules, but for throughput speed it was decided to create a separate module.

Process Manager
This module provides the capability of starting or stopping multiple SNApplication Framework applications.

Schedule Execution
This module provides a method of executing time synced ASCII messages and GSC commands based on an internal schedule passed in line by line. It will do conflict identification of overlapping schedule entries. New overlapping schedule entries will not be added.
Socket
This module provides a capability of creating a client or server TCP/IP socket.

Stream Monitor
This module allows multiple incoming streams of data to be parsed for different monitor points within the data. These monitor points can be used to determine a course of action. Any number of monitor points can be defined. The combined check section can be used to save the monitor point for analysis against a future incoming frame, checked against a value using string logic, report on the value or the result of a check or send a hard coded command to an external source.
Timer
This module provides the ability to send ASCII messages on a pre-defined cyclic basis.

Translator
This module allows a bit-by-bit translation of one data stream into another data stream.

VC Processor
This module provides the collection of statistics, gap checking and virtual channel de-multiplexer. It can handle native HDR and CORTEX frames as well as CCSDS. Even though this task could be done with many other modules, it was determined the efficiency gains doing this processing in one module was significant. It also simplified a number of configurations.

Many modules allow filtering and other processing based on “string logic” code. String Logic is a logic definition that allows a mathematical equation to be performed on the input data. There are three basic types of operators, unary <operator>(a), binary (a operator b), or logical. The operators with the above formats may be nested to any depth limited by computer memory. The following is an example of string logic: ((count % (({&output_rate} * 60) / {&rate})) eq 0). First substitute the globals output_rate and rate, then multiple output_rate by 60 and divide by rate. Take the modulo of count by the result. Then take that result and see if it equals 0 if so it is true otherwise it is false.