Reusable Software Components for Monitoring and Control of Telemetry Processing Systems

Jay Costenbader
Karen Thorn
Data Systems Technology Division, Code 520
Mission Operations and Data Systems Directorate
NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

NASA Goddard Space Flight Center (GSFC) has developed a set of functional telemetry processing components based upon Very Large Scale Integration (VLSI) and Application Specific Integrated Circuits (ASIC). These components provide a framework for the assembly of telemetry data ground systems for space projects such as the Earth Observing System (EOS) and the Small Explorer (SMEX) mission series. Implementation of the ground systems for such projects using a common set of functional components has obvious cost benefits in both systems development and maintenance. Given the existence of these components, the next logical step is to utilize a similar approach and create a set of reusable software components for the implementation of telemetry data system monitoring and control functions. This paper describes a generalized set of software components, called the Telemetry Processing Control Environment (TPCE), which has been developed to fulfil this need. This combination of hardware and software components enables the rapid development of flexible, cost-effective telemetry processing systems capable of meeting the performance requirements facing NASA in the coming decade.

Key Words: object-oriented, software, VLSI, CCSDS, telemetry processing

INTRODUCTION

Over the last ten years, the Data Systems Technology Division of the NASA Goddard Space Flight Center (GSFC) has developed a generalized set of functional telemetry processing components based upon Very Large Scale Integration (VLSI) and Application Specific Integrated Circuits (ASIC).[l] As a de facto demonstration of their generality, these components have been used, in various combinations, in

numerous mission ground systems. They also provide a framework for the assembly of telemetry data ground systems for future space projects such as the Earth Observing System (EOS) and the Small Explorer (SMEX) mission series.

Implementation of the ground systems for such projects based on a common set of functional components has obvious cost benefits in both development and maintenance. Given the existence of these VLSI subsystem components, a logical question is whether we can utilize a similar approach and create a set of reusable software components for the implementation of telemetry ground system monitoring and control functions. This combination of hardware and software components would enable the rapid, cost-effective development of flexible telemetry processing systems capable of meeting the performance requirements facing NASA in the coming decade. This paper describes a generalized set of software components, called the Telemetry Processing Control Environment (TPCE), which has been developed to fulfill this need.

GENERAL CAPABILITIES OF TPCE

The TPCE provides generic software components designed for controlling and monitoring the operation of the VLSI-based systems developed by NASA Goddard's Data Systems Technology Division. It controls the operation of these systems, provides mechanism allowing users to monitor their operation, and manages the distribution of telemetry data sets to user sites.

Operation of a telemetry processing system using the TPCE can be automated based on an activity schedule which can be automatically read from an external source (a file or network socket) or manually edited using a graphical user interface. Normally, TPCE will automatically initiate telemetry processing based upon activities identified in the schedule, the it also provides a display of the activity schedule and allows the user to add and delete telemetry processing session events and manually initiate telemetry processing. Control of the VLSI systems is accomplished through editable configuration sets, which include parameters used to specify the nature of telemetry processing. Thus, a single VLSI system can support various types of telemetry processing, simply by loading different configuration sets.

The TPCE system provides graphical displays that present the current status of telemetry processing. These displays update dynamically to present data quality information for on-going telemetry processing sessions, present historical, summary data quality and accounting information for processed sessions, and present the status of VLSI subsystems. TPCE also automatically generates reports summarizing the quality and accounting information for completed telemetry processing sessions.

Telemetry data sets generated by the VLSI systems as the result of processing are automatically collected by TPCE and distributed to specific user sites using TCP/IP and FTP. TPCE provides for queuing of data set distributions and re-transmission of individual data sets upon user request. TPCE also automatically generates reports summarizing the status of data distribution to user sites.

To allow for tailoring the operation of the system without modifying or writing software, TPCE provides a large set of system Preferences. To maintain an history of processing activities, TPCE provides an on-line, viewable log of system events. This event log can also be annotated on-line by the user.

ARCHITECTURE

The TPCE runs on Hewlett-Packard graphics workstations under the UNIX operating system and is composed of a set of UNIX operating system processes that interact to control and monitor operation of the VLSI-based systems. TPCE provides a windowed user interface based upon the X Window System and compliant with the Open Software Foundation's Motif user interface style. The entire TPCE system follows an object-oriented design and is implemented in the C++ object-oriented programming language. This object-oriented implementation provides two key characteristics which help to enhance the reusability of individual TPCE components. First, each TPCE object is designed to perform a single, specific purpose. This reduces the complexity of individual code components and also makes it possible to combine components in new, unforeseen ways. Second, the interface to any individual component is well-defined, but its internal implementation is hidden. Thus, the implementation of a component can change, to improve its performance, for example, without affecting other components.

TPCE is designed as a multi-process system which can be distributed over multiple UNIX workstations. A number of simultaneously-executing operating system processes cooperate and communicate to control and monitor operation of telemetry processing systems. The TPCE may operate on a single host workstation or in a multi-host mode, wherein any specific process may be allocated to one of a number of host workstations. In a particular system being built by NASA Goddard for the SMEX Fast Auroral Snapshot Explorer (FAST) mission, two workstations are configured with TPCE to cooperatively support operation of the system. One workstation supports control and monitoring, the second supports distribution of data sets to user sites. Additionally, all user displays from the second workstation appear on the first's monitor. This allows the user to monitor the operation of a multi-host system from a single display. [2]

TPCE processes are classified into two categories based on the temporal nature of their operation: static or transient. Static processes form the core of the system and execute continuously, throughout the operation of the system. Transient processes are initiated and terminated upon user request. For example, most processes which drive status and monitoring displays are transient, initiated and terminated upon user request.

The internal structure of almost every TPCE process is virtually the same: a process contains a series of C++ class objects which are initialized by the process and then invoked through system-event function callbacks. When some type of system event occurs - a user input, system wall clock time-out, inter-process communication, or file input - a system event is generated, causing a specific, predefined member function of one of the process' classes to be called. This event callback structure is supported by a set of C++ classes which provide general callback capabilities instantiated for specific types of system events. Examples of these C++ classes include WrEvent, which provides callbacks based upon user input to TPCE displays, AlarmClock, which provides callbacks based on wall clock timers, IPCServer, which provides inter-process communication callbacks, and DataServer, which provides status and monitoring data to registering objects through callbacks.

SOFTWARE DESIGN

Static processes form the core of the TPCE system and execute continuously, throughout the operation of the system. Figure 1 presents the complete set of static processes.

The Controller process manages initiation and termination of all TPCE processes and monitors process execution status. All static and transient processes, as defined in a user-editable table, are started by the Controller.

The Executor directs the activities of the TPCE system by controlling telemetry processing sessions. This process can be tailored to do any combination of the following:

- set up a recorder for recording an incoming telemetry stream or to play back a recorded telemetry stream,

- command the VLSI system to set up for telemetry processing sessions,

- collect summary telemetry quality and accounting data and generate quality and accounting reports.

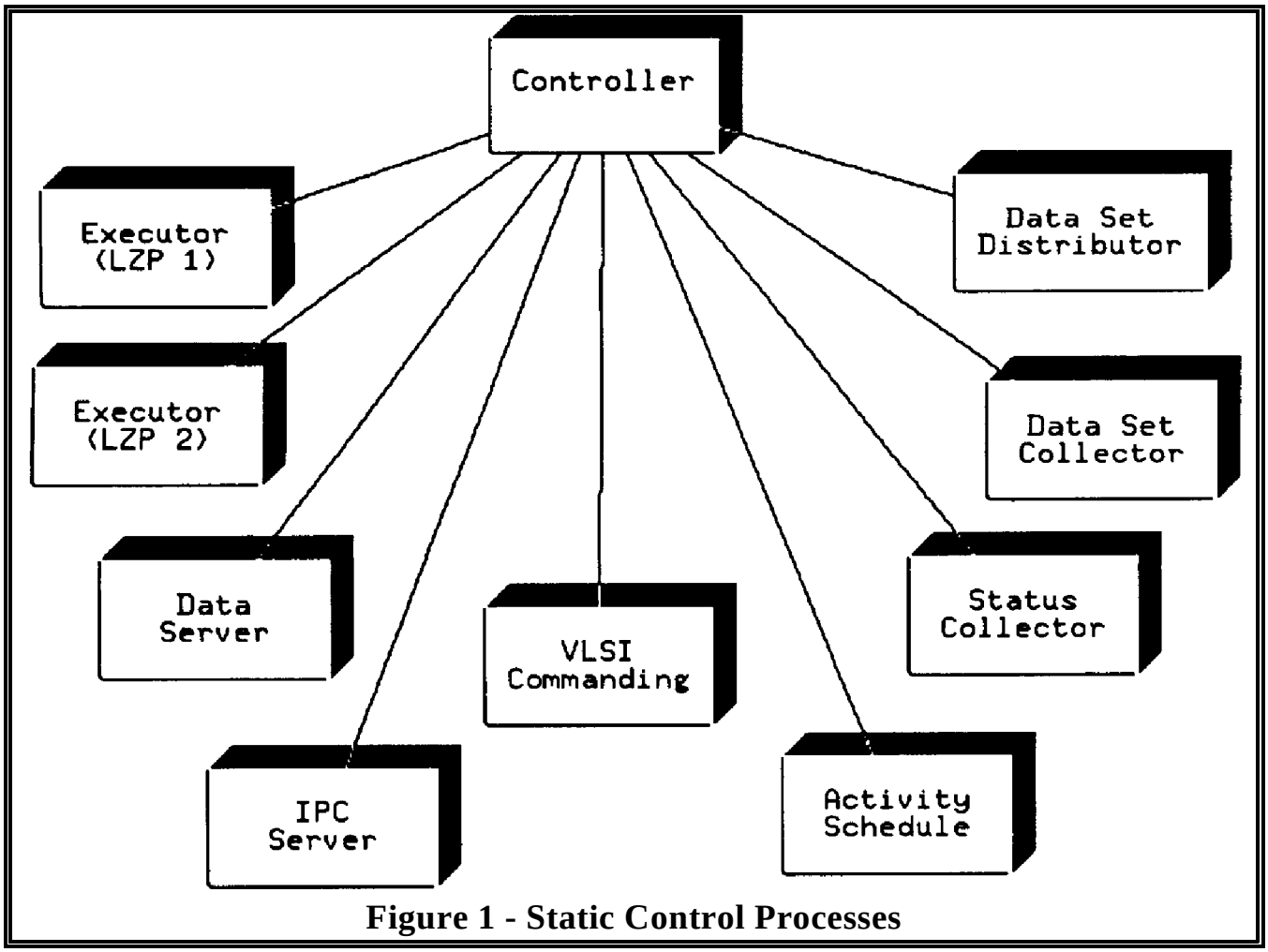

Figure 1 - Static Control Processes

The DataServer provides a central repository for system status and telemetry data quality and accounting information which may be required by multiple processes, potentially executing on different machines. A process may register for specific data parameters and have those parameters automatically delivered at regular intervals via the callback mechanism discussed earlier. The DataServer is driven from a standard, user-definable database of status parameters. The StatusCollector collects system and processing status which is then placed into the DataServer for distribution to TPCE client processes. The IPCServer provides a repository for inter-process communication messages similar to the DataServer, allowing TPCE processes to communicate with one another without knowledge of specific IPC mechanisms or even of each other. The IPCServer acts as a multicast server, broadcasting IPC messages to all processes requesting it, without the originator's knowledge of the recipients.

The ActivitySchedule process maintains the activity schedule, ingests scheduling information from an external file into the schedule, displays the activity schedule to the user, and allows the user to edit the schedule.

The DataSetCollector collects processed telemetry data sets and stores these for later distribution to user sites. The DataSetDistributor distributes data sets to user destinations, as specified in a user-editable file. All processes which wish to interact with VLSI systems must do so via the VLSICommanding process. The process manages a connection to the VLSI systems through which all commands are transmitted and responses received.

Transient processes are initiated and terminated upon user request. For example, most processes which drive status and monitoring displays are transient. The ConfigurationSetEditor provides a user interface for the manipulation of configuration sets, which contain parameters that define the exact nature of telemetry data processing. The user is provided with a graphical user interface through which he can edit the parameters associated with any configuration set and create and delete configuration sets. The PreferenceEditor provides a user interface for the manipulation of preferences, which are parameters which define operational characteristics of the TPCE. Preferences allow the user to define how the TPCE is to operate.

The following list identifies transient processes that drive graphical user interfaces which present information used to monitor the processing activities of the TPCE and VLSI systems. They are initiated and terminated upon user request and update dynamically at a user-definable rate. Unless so indicated, all processes gather status information from the DataServer.

- LZPStatusEditor displays summary processing status information
- BlockStatusEditor displays Nascom block level processing status
- FrameProcessingStatusEditor displays transfer frame processing status
- PacketProcessingStatusEditor displays packet level processing status
- PacketReassemblyStatusEditor displays packet reassembly status
- RealtimePacketEditor process can display the contents of telemetry packets to the user. It extracts these packets from the telemetry stream as they are being processed by the Processing Subsystem.
- SessionQAEditor process displays quality and accounting information for a particular telemetry processing session

- DataSetAnalyzerEditor process displays the contents of processed telemetry data sets, rejected packets files and rejected frames files for analysis.

- DistributionStatusEditor process displays the status of telemetry data set distributions

There are also a variety of processes which display the availability and performance status of VLSI system elements.

The TPCE processes are the largest, reusable components of the system. But there are a large number of C++ class components which comprise these processes which themselves are reusable. This component-level generality provides a framework for combining components to support additional capabilities not currently implemented in the existing TPCE system. Virtually all these components are C++ classes that provide very descriptive interface definitions, hiding details of the internal implementation. TPCE components include:

- tape recording, archival, play back, and contents management, as well as tape drive device management

- telemetry data quality and accounting data collection, storage, retrieval, display, reporting, and archival

- user editing, creation, deletion, and management of libraries of system configuration parameters

- operational event logging, archival, user annotation and display, and synchronous, acknowledged user messaging

- schedule ingest, display, storage, and editing

- batch, user initiated telemetry processing

- system preference parameters, to dynamically change characteristics of system operations

- various time types, clocks, and alarms

- graphical user interface (GUI) components for the X Window System and OSF Motif, and for gathering user input, time, and inter-process communication events

- generalized, distributed operating system process initiation, termination, and management
- UNIX system services such as file and directory management, FTP, E-mail, pipes, sockets, datagrams, and shared memory
- container objects such as ordered and unordered lists, dictionaries, and linked lists

SCENARIO

While each of the tasks in the TPCE system consists of a relatively independent set of related functions, most of these tasks need to communicate with one or more other tasks to exchange data or synchronize activities. The scenario below briefly describes the TPCE interprocess communication that takes place during a particular phase of processing: that of the automatic set up, monitoring, and termination of real time telemetry processing sessions. We assume that a real time telemetry processing session activity is described by an entry in the activity schedule.

The Controller task examines the activity schedule to determine when the next real time session is scheduled to begin. The task then sets an alarm to wake it when the time arrives for that session to be executed. When the alarm goes off, the Controller task extracts the session information from the activity schedule and sends it to the appropriate Executor task for the specified VLSI system.

Based on the characteristics of the session, the Executor task commands the VLSI system through each step of the data processing process. Since more than one task may need to command a given VLSI system, and each system can only accept a single command connection, the Executor does not communicate with the VLSI system directly. Instead it sends commands to the VLSICommanding task which then passes them on to the VLSI system. Command responses are in turn passed back to the appropriate commanding task.

Each VLSI system also generates status through a separate electronic connection. This status information is received by the StatusCollector process which simply makes the data available to other processes by storing it directly into the DataServer. Arbitrary processes throughout the system may then receive this information asynchronously. A large number of the processes that receive this information are graphical status windows. These processes display windows for the user in which processing and data quality status are displayed. In many cases, this information is simply received from the Data Server and displayed on the screen. Some of the status is also received by the

Executor processes in order to determine the state of processing. Session status is in turn made available to the ActivitySchedule process, so that the a currently executing session may be protected from inadvertent modifications by the user.[2,4]

CONCLUSION

The TPCE system provides a framework for the implementation of telemetry ground systems for future NASA projects. Implementing these systems based on NASA Goddard's VLSI-based hardware systems and the TPCE software components has obvious benefits in both development and maintenance. The availability and flexibility of these hardware and software components will enable the rapid, cost-effective development of telemetry processing systems capable of meeting the performance requirements facing NASA in the coming decade.

REFERENCES

1. Bennet, T., "Microelectronics Systems Branch Application Specific Integrated Circuits (ASIC)", 521-SPEC-002, NASA GSFC Microelectronics Systems Branch (Code 521), Greenbelt, MD, June 1992.

2. -, "FAST PPS Detailed Design Document, Volume 3", 521-DSGN-002, NASA GSFC Data Systems Technology Division (Code 520), Greenbelt, MD, March 1993.

3. Bennet, T., "Next Generation Functional Components for Space Telemetry Data Processing", NASA GSFC Microelectronics Systems Branch (Code 521), Greenbelt, MD, 1992.

4. Thorn, K., "FAST LZP Operations Concept Document", 520-OCD-001, NASA GSFC Data Systems Technology Division (Code 520), Greenbelt, MD, 1992.