Timeliner: Automating Procedures on the ISS*

R.Brown[1], E.Braunstein[2], R.Brunet[3], R.Grace[3], T.Vu[3], D.Zimpfer[4], W.Dwyer[5]
Charles Stark Draper Laboratory, Inc.
555 Technology Square, MS31, Cambridge, MA 02139
phone: (617) 258-2366 email: rbrown@draper.com

Timeliner has been developed as a tool to automate procedural tasks. These tasks may be sequential tasks that would typically be performed by a human operator, or precisely ordered sequencing tasks that allow autonomous execution of a control process. The Timeliner system includes elements for compiling and executing sequences that are defined in the Timeliner language. The Timeliner language was specifically designed to allow easy definition of scripts that provide sequencing and control of complex systems. The execution environment provides real-time monitoring and control based on the commands and conditions defined in the Timeliner language. The Timeliner sequence control may be preprogrammed, compiled from Timeliner "scripts," or it may consist of real-time, interactive inputs from system operators.

Historically, Timeliner was created to emulate the timelines for onboard crew procedures followed by the crew of the Space Shuttle. It was used as a simulation driver in tests of the Space Shuttle system, mimicking crew actions in monitoring and controlling the spacecraft systems. This version of Timeliner has been in use since 1982. The Timeliner simulation system was extended for use in other applications, and was tailored to provide real-time sequence execution and support interactive control commands that might be entered by the systems engineer: for example, sequence start and stop.

In 1992, NASA selected Timeliner as the User Interface Language (UIL) for Space Station Freedom, and later for International Space Station (ISS) to be executed on the Space Station's real-time core command and control and payload control computers. Since that time, Timeliner has evolved as a modular, extensible system that allows scripts to be developed and executed in virtually any systems environment; Timeliner can be applied to control a variety of target systems and meet a wide range of mission objectives. It is currently in use on the Space Station for U.S and Japanese core and payload operations. In addition, an extended version of the real-time ISS Timeliner system is to be used throughout the U.S. by several academic institutions and NASA centers as a UNIX-based, general-purpose, procedure executor for ISS payload development, simulation and test environments

In general, the Timeliner system lowers the workload for mission or process control operations. In a mission environment, scripts can be used to automate spacecraft operations including autonomous or interactive vehicle control, performance of preflight and post-flight subsystem checkouts, or handling of failure detection and recovery. Timeliner may also be used for mission payload operations, such as stepping through pre-defined procedures of a scientific experiment.

For the International Space Station, Timeliner provides significant mission cost savings and productivity gains. Since Timeliner executes multiple automated sequences in parallel, multiple operations and experiments can occur simultaneously, utilizing the Space Station crew members for only the 'human required' activities (e.g. replacing filters, exchanging components, and observing visually successful or failed events). Also, since much of the sequencing is provided autonomously, the crew and ground operations training required is greatly reduced, providing a significant savings in mission cost. In addition, the use of the Timeliner provides significant improvements to mission success, reliability, and

---

* This work was supported by the NASA under Contract # NAS-9-01069

[1] Principal Member Technical Staff, C.S.Draper Laboratory

[2] Member Technical Staff, C.S.Draper Laboratory

[3] Senior Member Technical Staff, C.S.Draper Laboratory

[4] Program Manager, Space Transportation, C.S.Draper Laboratory

[5] Timeliner Technical Monitor, NASA/ JSC

safety since human sequencing errors are eliminated (e.g. steps executed out of order, duplicated, or omitted, incorrect command parameters, or improper evaluation of system data).

The Timeliner language was specifically designed to allow easy definition of scripts that provide sequencing and process control of complex systems. The Timeliner language is specifically intended as an operational language that produces preprogrammed sequences (Timeliner bundles) to be executed either in a totally autonomous mode or in an interactive control mode. The complete Timeliner language for ISS is specified in the NASA document SSP 30529, User Interface Language Specification.

The Timeliner script written by a mission planner may contain Timeliner "statements" that, when compiled and executed, cause actions to be taken not only on the basis of time, but also on the basis of other general conditions such as system events or complex dynamic conditions. Timeliner statements are grouped together to form a sequential set of actions called "sequences." Timeliner sequences, which logically execute in parallel, can be grouped together to form "bundles." Figure 1 illustrates the Timeliner script organization and its control paradigm.

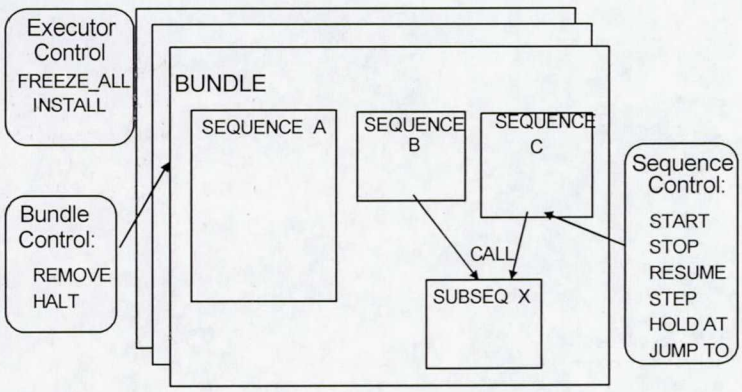

Figure 1: Timeliner Script Organization

Timeliner sequences contain instructions for monitoring, controlling, and reporting on the operations of the target system. The Timeliner language provides statements for organizing and performing the instructions. There are six general types of statements:

- Block declaration statements -- these organize the bundles, sequences, and subsequences of Timeliner (e.g. BUNDLE, SEQUENCE).
- Timing control statements -- these affect timing or flow of execution (e.g. EVERY, WAIT, CALL)
- Conditional control statements and their modifier clauses -- these allow for specific conditions that control execution based on general system values (e.g. WHEN, WHENEVER with modifiers BEFORE, WITHIN, OTHERWISE and IF-THEN-ELSE).
- Actions statements -- these are used to carry out actions affecting the target system and supports interaction with the mission operator (e.g. COMMAND, SET, MESSAGE, MESSAGE/PAUSE).
- Bundle/Sequence Control statements -- these are used to manage bundles and to control sequence execution (e.g. INSTALL, REMOVE, START, STOP, RESUME, STEP).
- Non-executable statements -- these are used for definitions of symbols and reserving of local storage.(DEFINE, DECLARE)

The Timeliner system allows the scripts to be defined, verified, and executed in a rapid, 'roll-in' fashion, independent of target system software builds. Figure 2 below illustrates the Timeliner system components.

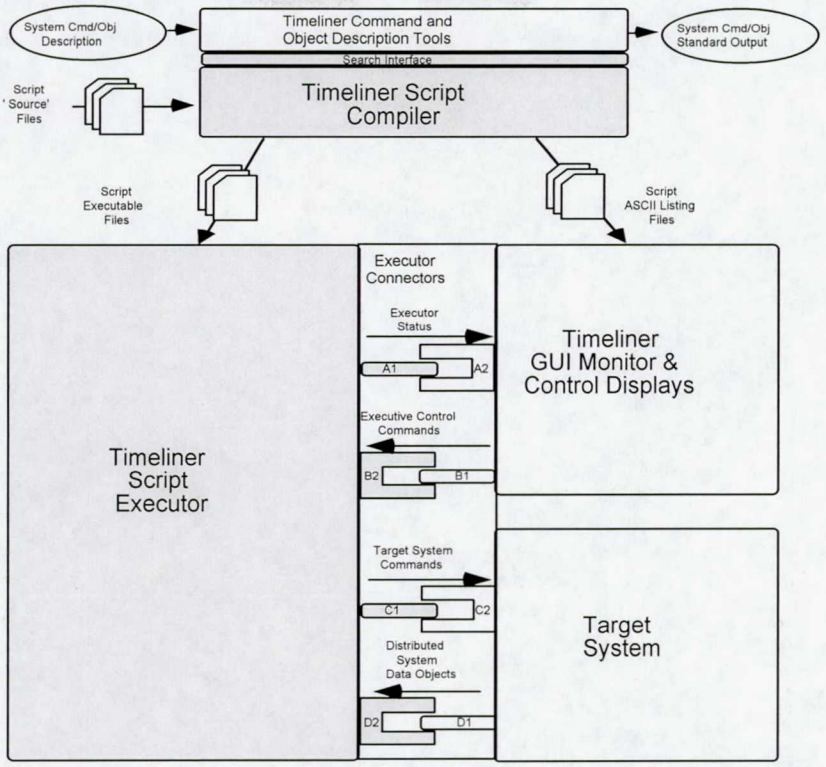

Figure 2: Timeliner System Components

The Timeliner Compiler system architecture provides several key capabilities. The script definition (bundle) is provided to the Compiler as a simple ASCII text file, and therefore can be produced in any text editor, word processor, or planning system. The Compiler also supports independent definition of system data object and command formats, types and names definitions (names to be used in the script). In this way, the system definition database may be developed, maintained, and provided for script compilation independent of the Timeliner Compiler environment software build, and allows script algorithmic development in abstraction from detailed system data formats. In addition the Compiler provides 'cross-platform' compilation that allows the Compiler and Executor to be platform independent.

The Timeliner Executor system architecture also provides several key and powerful features. As indicated earlier, a compiled bundle may be 'rolled-in' or INSTALLed, executed, and REMOVEd independent of execution environment software build. In addition, the Executor supports execution and independent control of multiple bundles, in parallel, which may itself contain multiple sequences that execute in parallel (either in synchronous or asynchronous fashion) and also can be independently controlled. For the ISS Timeliner, the Executor software also provides Command and Data Handling (C&DH) functions necessary for integrating into the ISS system architecture (e.g. bundle memory management, sequence execution time management, input and output command and message queuing, multiple machine format system data conversion).

The Executor supports several unique features to monitor and control executing scripts. The Executor in concert with the operator displays provides the ability to precisely track, view, annotate, and interactively control an executing Timeliner procedure. Through the Timeliner displays, the operator can monitor the status of an executing procedure, and receive messages from the executing script. The message capability provides a key link between the autonomous and manual portions of a crew operation. For example, automated commanding (e.g. TURN ON, SELECT GAS, IGNITE, TURN OFF) and automated system monitoring (e.g. WHEN POWER IS ON, WHENEVER PRESSURE > ACCEPTABLE RANGE, IF DIAGNOSTIC_STATUS = FAILED) can be sequentially integrated with a crew MESSAGE

to perform a manual operation (e.g. MESSAGE/PAUSE "Please replace gas filter 1, then Resume sequence"). Figure 3 shows the main Sequence Control display provided for the ISS crew.

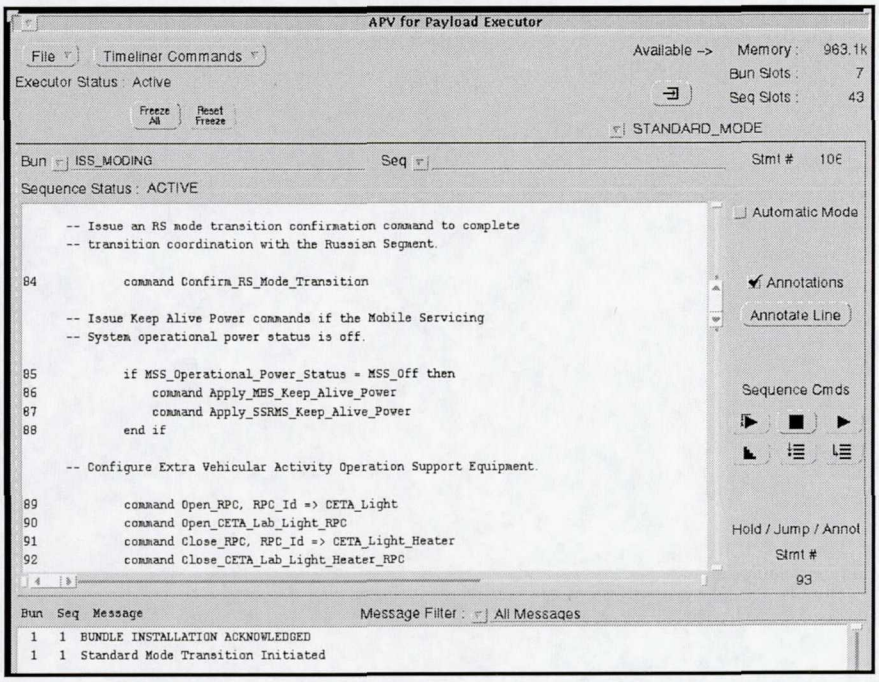

Figure 3: Sequence Control Display

The Timeliner software architecture supports the application of Timeliner to many script development and execution environments. The goal is to allow Timeliner to be easily integrated with diverse implementations of the interfacing systems while preserving the common language and control features. The Timeliner Adapter/Kernel Software Architecture achieves this goal.

The Timeliner Kernel provides all features of the Timeliner system that are independent of host machine or target system interfaces. The Kernel serves as the core element for both the Compiler and Executor, and is insulated from external interface dependencies by the Compiler and Executor Adapter. The Kernel implements the Timeliner language, performs all sequence execution, and control and maintains the state of the Timeliner bundles and sequences. In addition, the Timeliner Kernel provides common utilities that adapters may utilize for parsing and evaluation of external data references.

The Timeliner Adapter, on the other hand, provides all the features of the Timeliner system that are dependent on host machine or target system interfaces. It provides the main link to the target system to be monitored and controlled, including reading and writing system variables, and issuing commands. The Adapter also provides bundle execution status, sequence execution errors, and messages (as defined in the MESSAGE statement) to the display or telemetry systems. Additionally, the Adapter handles incoming control commands such as sequence Start and sequence Stop and the bundle commands Install and Remove. Finally, the Adapter provides access to operating system features such as tasking, semaphores, time, and memory management.

Figure 4 illustrates the Adapter-Kernel architecture in the mission planning and bundle development environment. This figure shows the interfaces between the Kernel and Adapter ground tools and how the Adapter provides connectivity to ground-based external systems, including target description database and file systems.

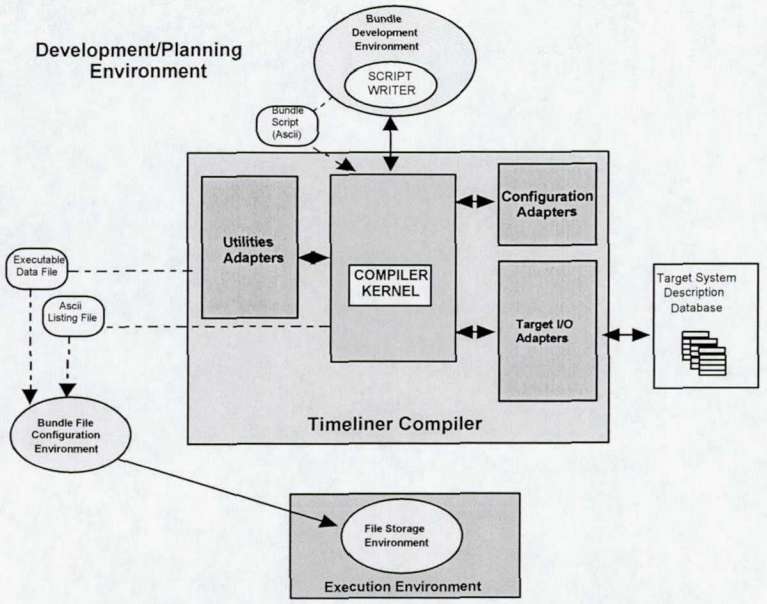

Figure 4: Adapter-Kernel Architecture in the Planning/Development Environment.

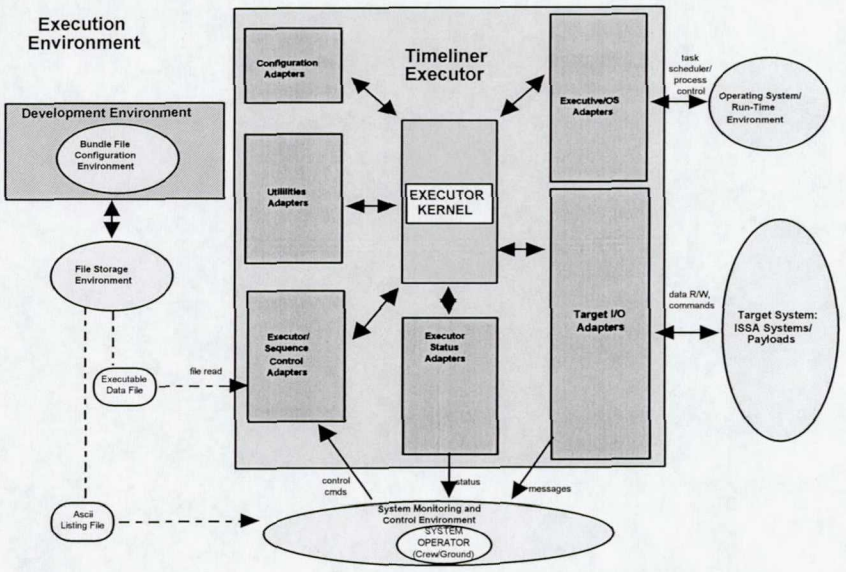

Figure 5: Adapter-Kernel Architecture in the Execution Environment.

Figure 5 shows the interfaces between the Kernel and Adapter and how the Adapter provides connectivity to the external systems, including the target system to be monitored and controlled, the host operating system, the display and control systems, and run-time file storage systems. This includes interfacing with the host operating system, supporting required tasking and control architectures, and providing required semaphore and memory management functions. This allows for a multi-bundle execution environment in which many bundles can run simultaneously and can be installed and removed independently.

The Adapter/Kernel architecture was defined to facilitate the application of the Timeliner tool to diverse systems and isolate the dependency on any particular feature of the data management systems or run-time environment. It preserves the constancy of the language features by encapsulating the functionality into the Kernel. This achieves the goal of maximum interoperability when applied (through the Adapter) to separate environments.

At the implementation level, the Adapter/Kernel interface is defined by a set of Ada 'package specifications'. These interfaces are said to be 'owned' by the Kernel. In this way, many diverse Adapters can be interfaced with a common Kernel and achieve common functionality and interoperability. Use of a common Kernel maximizes code reuse in application development, thus reducing the total software development and test costs to customize the Timeliner system. This partitioning facilitates a modular development of Adapter functions, allowing separate development of very complex procedures to accommodate complex external interfaces or very simple procedures and functions to support simple interfaces.

The Timeliner system is used in several product configurations described below:

Timeliner Kernel -- This Timeliner configuration is the language 'core' software. It is the core building block for various Timeliner 'adaptations' (via Timeliner Adapters). It is currently provided to NEC of Japan via NASA's COSMIC database for integration with the Japanese Experiment Module (JEM) Adapter developed by NEC. The re-use of the Kernel software, also used in the ISS Timeliner configuration, provides extensive interoperability between U.S. and Japanese operations interface.

ISS Timeliner -- This Timeliner configuration combines the Timeliner Kernel with an ISS MDM Command and Data Handling (C&DH) specific Adapter. The ISS Adapter adds all critical features for crew interactive monitoring and control, concurrent multiple procedure execution, real-time memory management, and machine type data conversion. It was developed to formally defined software interfaces for both the procedure development and execution system environments. The Ada source code delivered to the ISS Mission Build Facility is a single Timeliner product containing two unique development and execution environments – one for the C&C MDM-JSC/MOD and one for the Payload MDM-MSFC.

Timeliner For Solaris -- This Timeliner configuration combines the ISS Timeliner Compiler and Executor build for a UNIX environment with a GUI Compiler interface, a basic system object and command database definition capability, interactive GUI Executor interface, and generic target system command and monitoring interface via standard UNIX socket protocols. This configuration is provided to Boeing-Huntsville to be incorporated into a standard ISS payload development and test facility (Suitcase Test Environment For Payloads - STEP) to be provided to several key universities and NASA centers for payload development.

Timeliner Toolsuite -- This Timeliner configuration combines the ISS Timeliner Compiler and Executor with a GUI interface to the Timeliner system to provide an integrated script development and test environment. The Toolsuite runs stand-alone in a Windows NT desktop or Linux environment. It includes both a functional and a real-time test capability. This configuration is provided to ISS Flight Controllers and Payload Officers for development and checkout of Timeliner scripts on their desktops.