Right-Sizing Test Tools

A low cost alternative to full-scall system checkout equipment for Instrument & Payload Applications

SESP 2010

Andy Armitage

Terma BV
Schuttersveld 9,
2316 XG Leiden The Netherlands
Email: aba@terma.com

INTRODUCTION

In the last few years Terma, in partnership with Satellite Services, has developed a "small" monitoring and control system with equivalent or compatible features to those provided at system level AIT and spacecraft operations. At the start of 2010, this system was completed to the level where all monitoring and control features of SCOS2000 are now implemented.

The system has been used at payload- or instrument- level in several ESA missions including BepiColombo, Sentinel1, Sentinel2, EarthCare, Galileo FOC and SGEO. Initially there was a clear differentiation in features between the payload level and system level, however this gap is now largely closed.

MOTIVATIONS

Why was this kind of "small" system needed? Both technical and commercial motivations led to an inevitable conclusion. Terma had delivered several system level checkout systems for different ESA missions using a CCS based on SCOS2000, and for certain projects had also reused this system at instrument level. While these projects were in the end technically successful, and appreciated by their users, this experience highlighted several underlying problems.

Firstly, any system based on SCOS2000 is necessarily quite complex to develop, to configure and to use. While it is possible to streamline the delivery of SCOS2000 to the user, this reaches a schedule and budget minimum below which it is difficult to go. For example: when such a system is setup to monitor and command front end equipment using TM and TC packets, then this EGSE data must be characterized in a database distinct from the end user database. Since front ends and SCOE are not very standardized, this means that each project would have to develop this EGSE database and test it with the real front end before the project could be completed.

Secondly, for the end user, using TM and TC packets to command front ends is feasible but quite tricky and does not offer optimal performance. PUS packets may be appropriate for commanding a spacecraft in orbit but are not the most expressive or efficient way to command an electronic equipment that is only a few meters away. Furthermore some types of front end work better when there is a quick command and asynchronous reaction to events, simply because of the nature of the external interface. A TC and TM based approach is first limited by the maximum speed of the telecommand chain, which in SCOS2000 is either 2 packets per second (with checking) or 20 (with no checking). This is not sufficient for some modern front ends. Trying to configure a 1553 front end with a bus schedule and injecting asynchronous commands and polling TM as a result using TM and TC packets can be done, but one would prefer an easier way, and certainly with lower latency. Over time, as SpaceWire is becoming more adopted, the need for speed will only increase.

Thirdly, over time it was becoming clear that maintaining SCOS2000 on modern operating systems and hardware platforms was increasing in cost rather than decreasing over time. This was in direct conflict with the need to deliver such systems ever quicker, and with ever more constrained budgets. Budgets and schedules at instrument or payload level are even more constrained than at system level, and furthermore, since there is often a complex direct electronic interface to the system under test, the EGSE has to manage more complex rather than simpler interfaces.

It could be argued that SCOS2000 maintenance was being performed by ESOC and would therefore naturally keep up to date under ESOC investment. However the priorities of ESOC are mission operations, and the latest ESOC developments have been tended to be slightly disadvantageous for EGSE, and furthermore to increase the size and complexity of SCOS2000 rather than reduce it.

Faced with the probability of increased development costs, less than desirable performance, in an environment where the opposite trend is expected by the customer, Terma was faced with a difficult choice. To embark on the development of the new Monitoring & Control system is a daunting prospect since previous systems have cost of the order of 30MEURO to develop. To start from scratch using all new interfaces would be untenable since no end user would want to learn a completely new system, nor would database developers be prepared to invest in adaptations for a Terma-only product.

Any new system would therefore have to provide continuity and compatibility with existing systems for the end user. The question then asked was: is it possible and economical to develop a new system using the same test language and the same database interfaces as our current systems, but based on new technologies?

This question had a positive answer because:

1. the test language used at CCS level is freely available, is stable and familiar technology, easily adapted

2. the SCOS2000 MIB database interface, while not perfectly normalized, and not always perfectly designed, is nevertheless well known, relatively well described, and well understood inside Terma and other companies

3. during previous EGSE developments, the QT toolkit had been used successfully. This toolkit provides many benefits described below, and greatly reduces the need for turnkey development.

Therefore Terma set out to begin development about 3 years ago on a completely new monitoring and control system for small scale users, with no complex distributed system features, and with the main goals of being lightweight, fast, portable between platforms, and easy to install.

After 3 years this system is now functionally completed. At the start of 2010, all features of the SCOS2000 telemetry and commanding models were implemented.

FRONT END INTERFACING

Discussions with Satellite Services BV (SSBV), a leading supplier of EGSE front ends began at an early stage. This company was a natural partner, since Terma had teamed with some success several times with SSBV, and their location was very close by.

Together with SSBV, an approach to front end interfacing was conceived that provides the single most important feature of this new system.

Rather than using TM and TC packets, an API approach would be used (Application Programming Interface). SSBV had already provided remote command & control features for their front ends, since these were already used in their own user interfaces. These API's were available as compilable header files.

The approach adopted was to take the SSBV API definition and convert into a QT C++ format that allows objects and their interfaces to be queried at runtime. The resulting compiled object when deployed as a shared library or DLL has the advantage of speed of a fully compiled program, but also has possibility of runtime introspection such as one would find in other technologies such as Java Beans or .NET. However using QT and TCL together one can also deliver best performance, and be completely portable between WIndows, LINUX and MacOS.

The new system therefore uses external system plugins, which are implemented as shared libraries in the specific technology of the operating system platform. The portability of these shared libraries is guaranteed by QT.

The new system is able to load a plugin that defines its external interfaces at runtime, and then to automatically query those interfaces. Since the chosen test language is interpreted, it is of course possible to add the front-end interfaces defined in the DLL directly to the test language, at runtime, and at very high speed. Adding the plugin functions is almost instantaneous, and a call to a front end API function is almost as fast as a direct function call in C++.

The introspection provided by QT also allows certain secondary but very important features:

1. Runtime syntax and type checking can be automatically implemented in a generic way

2. Generic type conversions can be performed, including binary data

3. User interfaces can be built in a generic way, that adapt to the features of the current plugin.

This new system was called **CMDVS** ("Commanding Monitoring Data-handling & Visualization System), which although not the most memorable acronym, reflects the continuity with SSBV's CMS product.

Today, several ESA missions are using CMDVS as part of their payload- or instrument-level EGSE.

FULLY FEATURED

In early versions of CMDVS, some advanced monitoring and control features were omitted. The development was approached in stages according to user requirements. However this development has now reached its completion, and all telemetry and command processing features are now implemented. In fact the opportunity has been taken to re-factor certain limitations of SCOS2000, for example the contents of variable packets are fully available to test sequences, calibrated and limit checked exactly as parameters from fixed packets. Variable length parameters are supported in both TM and TC direction. Telecommand header generation is completely database driven without even the assumption that CCSDS headers must be used.

The goal in this implementation was to find answers to the last remaining EGSE requirements from end users that SCOS2000 could not meet, and could not feasibly be adapted to meet. We do not list all such features in this paper but direct the reader to the web site (http://projects.nl.terma.com/CMDVS) where a they are completely summarised.

All code has been completely written for 64-bit compatibility. This does not simply mean that the system can be compiled and will run acceptably for 64 bit architectures. This means that the basic TM and TC processing works using 64-bit data including all 64-bit packet utilisation standard type codes/function codes, binary bit sequences and enumerated parameters.

The system includes a fully modernised user interface written using QT. Initially the system appears in a highly compact form with a standard layout, however the user can de-dock docking widgets throughout the user interface, position them where desired and then save the screen layout.

Almost all user interfaces where a short hand abbreviation is used provide a bubble help with the long-hand complete description. Almost all windows can be copied and pasted directly into other windows. For example a table of parameter values into a spreadsheet, raw packet telemetry in hex, or a set of telecommand verification stages. The intention is to integrate closely with the typical office tools encountered on modern PC desktops.

COMPATIBLE

The test language, monitoring and commanding behaviours, database and synoptic picture implementation are fully or very closely compatible with CCS based on SCOS2000. The goal is to allow test environments in which the end users have invested enormous effort to be able to be transferred to other systems. This compatibility also ensures familiarity between test engineers at different system levels, which reduces the need for system specific training.

Some compatibility compromises have been made where a long-needed improvement was required. The approach taken in the case of a conflict between the need for improvement or the need for absolute compatibility was to improve. All known compatibility differences with SCOS2000 have been slavishly documented.

ADAPTABLE

One of the maintenance problems of SCOS2000 was that while the initial design had been fully designed to be mission generic and customisable for specific missions, over time, certain key parts of the code base had been approached in a short term way. This has led to a dire need for refactoring in some of the most critical parts of the system.

CMDVS has been approached from the assumption that budget for deep code changes for specific missions was not available and would never be available. Therefore mission adaptations have been designed from the start. One classic example is that many programs customise the use of source sequence counts. Outwardly this appears to be a trivial change since it is only a sequential number changing from one packet to the next. Unfortunately inside the long maintained code base for telecommand verification, this mission customisation in SCOS2000 leads to yet another modification of the same highly complex component. In CMDVS it is possible to implement and configure a different source sequence count strategy with no effect whatsoever on the generic telemetry and telecommand verification code.

Another example is that the telecommand verification stages are often very different for different missions. Many ESA missions do not use the 9 progress stages of a telecommand. CMDVS supports all the SCOS2000 stages, plus one more stage for telecommand finalisation (to overcome the limitation that no stage is available in the SCOS2000 model after receiving the last type 1 TM for completion). However in total up to 64 verification stages can be configured using a mask. Stages that are not configured in this mask are ignored, are not checked and do not appear in the user interface

PORTABLE

The system is delivered routinely on different varieties of WIndows and SUSE Linux. It compiles and runs on MacOS X, but is not available as an installable package.

The possibility to “Internationalise” the user interface using the features offered by QT have been maintained throughout. While today there are few cases where English is not used in ESA projects, this feature would be needed if it were ever needed to adapt to non-Roman text.

BENEFITS FOR BOTH DEVELOPER AND END-USER

Using a front-end plugin means that during the EGSE development, neither Terma nor SSBV are forced to devote any time to:

- defining an EGSE TMTC database
- verifying and documenting that EGSE database
- defining ICD documents

This approach significantly de-risks EGSE projects, and removes an element of the development that is always on the critical path. It therefore ensures cost reductions because the time spent on integration and project deployment is much more less.

Terma and SSBV have worked together to maximise the automation of plugin maintenance.

Some examples are:

1. When an interface is changed to the front end by SSBV, it is checked into configuration control, and after an overnight build, the next day the result is available to SSBV for installation. SSBV do not have to hold long and contentious discussions about minor API changes. This saves considerable time.
2. The API documentation is included as comments in the front end API. This means that using automated extraction and conversion tools, (Doxygen and others), the documentation is already available in the online help, in the next build. This avoids tedious maintenance to keep the documentation consistent with the implementation. It also allows certain tricks to be used in the user interface to call up documentation as bubble help when the user hovers his mouse over an API.
3. Consistency checking information is also extracted automatically, which ensures that the test language consistency checker is also able to check test sequences that use this API.

Maximal automation as in the above examples reduces the tedious and error prone human maintenance tasks performed in a project, and this significantly reduces the cost passed on to the customer.

From the end user point of view, this automated API approach means that the EGSE database does not have to be separately maintained, and he can concentrate on the database of his system under test. It also means that the end user does not have to lose time on merging and de-merging databases from the system and from the EGSE levels

POSSIBILITIES FOR SCOE AND INSTRUMENT EGSE

When used at instrument or payload level, the CMDVS is effectively an intelligent SCOE. There are numerous scenarios in new projects where the CMDVS based payload EGSE is itself a SCOE to the main system EGSE.

There is a further example from Galileo FOC SMS which is itself a CMDVS system, where a small thermal monitoring and cooling SCOE must be used by both payload and system level. It has access to local thermal devices such as a chiller and thermocouple scanners. It is deployed on a tiny low power PC (effectively a net book). In effect the SCOE is a sub-SCOE to the payload EGSE. In this case, a CMDVS installation is used with the same plugin as used in the CCS and Payload EGSE and the SMS, that supports protocol exchanges with external systems, for example to respond to commands and to generate SCOE telemetry. The “intelligence” of the SCOE (i.e. the specific application functionality) is implemented in TOPE/TCL test sequences which are highly adaptable, and allow the SCOE to provide an attractive user interface on a small touch screen. The system appears to the user as an embedded SCOE controller. This complete SCOE has been developed in a matter of a few months.

The user is not in fact aware that the underlying system is actually a powerful TMTC monitoring and control system!

POSSIBILITIES FOR SUBSYSTEM SIMULATION

When working with direct hardware data acquisitions it nevertheless still possible to work partially with a TM and TC database. Although the packet TM and TC layout may not yet be known, the data that must be measured and the direct command interfaces to the system under test are already well known.

This allows the CMDVS user to write test sequences for, and create synoptic pictures for parameters whose allocation to packets are not yet known. It means that already at an early stage the database material needed for system level operations are being used and tested, and means that more test products can be transferred to the system level.

When the subsystem telemetry packet layouts become known (are defined in a database) it becomes possible with CMDVS for the instrument EGSE to generate those packets using the parameter measurements it acquired directly from

the front end. In a limited way, it is possible to communicate with the instrument or payload using TM and TC packets via its EGSE as though it were already mounted on the spacecraft.

BENEFITS FOR SYSTEM LEVEL EGSE

An extremely fruitful result of this development has been the realisation that some of the technologies used in CMDVS can be reused at CCS level. Even though CMDVS is implemented on a Windows platform, and the CCS is implemented using LINUX, the same approach is fully portable.

The dynamic plugin approach is now routinely used in CCS, and the external interface manager of the SCOS2000 based system is now implemented as a generic process, and a configurable project specific plugin. This plugin can be queried, and now, from any test sequence started in the CCS, the API presented by the plugin is automatically loaded. This means that for any project, special interface functions can be added that allow additional features to the standard TM and TC exchange. To give examples most programs require to be able to send string formatted commands to their front ends. Using the CMDVS approach, there is no need to encapsulate these commands in TC or TM packets, but the messages can be created by a direct API call.

Another benefit of this approach is that the same plugin can be used at instrument or payload level and at system level. This gives much greater confidence that the protocol implementation is understood in an identical way on both sides of the EGSE network link.

This new approach has also helped to overcome some of the doubts about future maintenance of SCOS2000 based EGSE. The approach is extremely efficient and economical to develop, and when both CMDVS and SCOS2000 based CCS are used together, at system and subsystem level, there is compatibility of:

- database
- test language
- synoptic pictures
- protocol implementation

FUTURE POTENTIAL

For the future it could be considered that this plugin approach is not necessarily unique to SSBV front ends. Some effort has been spent in past ESA studies to standardise front-end interfacing. This is a difficult task because no front-end supplier is especially interested to invest in compatibility with their competitors.

One insight that could be drawn from this experience is that it is perhaps not necessary to standardise front end interfaces, only to standardise the format in which they define their API. While different front end supplier companies may use C++ or TurboPascal header files, there is currently very little uniformity between their actual functional interfaces. To force these suppliers to change for the sake of standardisation at their own cost is likely to be a slow process. One way to side-step this problem could be merely to ask the front-end suppliers to express their API in a standard format that can be computer processed into a test language plugin. Example formats could be CORBA IDL or an XML format such as WSDL. Building a language plugin from this definition is an approach that could be generalised without favouring one supplier or another but nevertheless ensuring that different front ends and SCOE from different suppliers can nevertheless be used in the same mission without excessive work.

This approach would also not help to remove a barrier to standardisation of procedure languages, since the possibility would exist for any procedure language developer to build their language extensions in a generic way that can be adapted for any front end supplier.

Applying some of the CMDVS technologies to SCOS2000 EGSE has ensured a potential development road map for several years to come. While SCOS2000 will no doubt reach the end of its software life-cycle in the coming years, there is no longer any need to consider this a forced retirement. This experience has opened up choices for how to approach the development of future systems. On the one hand, this mixed approach has addressed and solved many of the concerns about future performance and maintenance of SCOS2000 for EGSE purposes.. Alternatively, the original SCOS2000 code base could be phased out, providing a SCOS2000-compatible system entirely based on new technology. Terma welcomes input from ESA and the primes as to which is their preferred path