SESP 2010

Session:	Session: EGSE Architecture 1/2 (05)
Type:	Concurrent Session
Date:	Tuesday, September 28, 2010
Time:	14:00 - 15:30
Room:	Einstein
Chair:	
Co-chair:	
Remarks:	

Seq	Time	Title	Abs No
1	14:00	EGSE-Lite: A Light Weight EGSE/Simulation environment for LEON based Satellite Subsystem & Payload Equipment	

Cordero, F; Irvine, M; Mendes, J; Williams, A
VEGA Deutschland, GERMANY

The paper describes a configurable EGSE/Simulation environment, designed to support the development lifecycle of spacecraft equipment containing on board software - in our case a LEON3-based payload.

The "EGSE-Lite" environment has 3 main purposes:
- Software Development and Verification Facility for OBSW development and validation
- Simulation for performance and design feasibility assessment
- EGSE for hardware integration and testing of equipment components

The EGSE-Lite environment has already been successfully deployed to CESR, an Institute in France for Science Payload development and integration, providing an emulation of the LEON3 processor and models of the payload detector units and mass memory.

With a modular design and approach, EGSE-Lite can be modified into different configurations, from a full-virtual environment containing a processor emulator running OBSW and simulation models, to a complete hardware-in-the-loop configuration interfacing to integrated hardware units.

The modules currently included are as follows:

- Processor module containing LEON3FT and I/O peripherals - configurable as processor emulator (Gaisler TSIM) + Gaisler UT699 I/O module model or Aeroflex UT699 LEON3FT development board
- Mass memory simulation model - can be replaced by actual mass memory HW device
- Payload detector simulation models - can be replaced by actual detector HW units
- OBC interface simulation model

The EGSE-Lite also includes a central check out system for monitoring and control of the units under test. ESA's SCOS-2000 software is used for this element. The checkout system also includes an EGSE gateway enabling a direct interface with CCSDS TM/TC packets.

The key design characteristics of the EGSE-Lite are:
a) Simple coupling of interfaces between the various equipment modules: based on TCP/IP sockets and use of same communication

protocols for the exchangeable modules
b) Use of SCOS-2000 software as Central Check-Out System
c) Configurable to use model or real H/W versions of system components - Mass Memory, Payload Detectors.
d) Use of Eclipse and LEON Integrated Development Environment (LIDE) available from Gaisler Research.
e) Use of GR TSIM as virtual LEON emulator and GRMON + Aeroflex UT699 LEON3FT as development board
f) Use of a PCI-Spacewire interface card to connect the processor board with the equipment simulation models

The system has a "Star architecture" around a central "packet switch" which allows to quickly achieving any possible configuration, from virtual to HW-in-the-loop, using a simple XML file. Typical configurations include a mix of simulation models, with multiple instances and users, sharing unique EGSE hardware resources. The "packet switch" also provides a second level of 'packet sniffing & recording' in case this is required.

The design of the EGSE-Lite environment and the use of a sockets-based TM/TC interface would enable the system to be integrated into a larger satellite system if required.

The EGSE-Lite takes full advantage from COTS products (such as Gaisler TSIM), and ESA software products (such as SCOS-2000) to provide a light weight, scalable and cost-efficient solution, for integration and validation of satellite subsystems and payload equipment containing embedded software.

2 14:30 EGSE Architecture and the New Generation of Software Technologies

Poletti, M.[1]; Schito, E.[1]; Antonelli, D.[2]; Martinelli, E.[1]
[1]Thales Alenia Space Italia, ITALY;
[2]Intecs, ITALY

Introduction
This paper presents an EGSE architecture that takes benefit of the improvements made during the last years by the SW development industry in terms of UI (User Interface) and SW architecture and translates them in advantages for satellite AIT users. The reason is linked to a simple consideration: in the last 15 years the basic EGSE architecture never changed. Focus has been made on functions and tasks to allow TM/TC processing to cope ECSS standards and with specific modifications to answer to dedicated needs of the EGSE operators in a suitable and effective manner.
Despite all these efforts, the EGSE environment inside the main Prime contractors (at least for the ThalesAleniaSpace situation) remains heterogeneous and can be summarized as follow: 1. an "internal asset" EGSE, mainly used for commercial programs
2. a "third part" EGSE imposed by the programs
3. SCOS-2K, highly recommended for ESA programs
Moreover these EGSEs are used in parallel by different satellite AIT teams and, to complete the discussion, all these EGSEs are "closed" systems. They use their own data-base, custom SCOE interfaces, a proprietary test language and specific tools to interact with the test operators and different UIs. In general they are highly configurable but not "open". With this scenario, the cost of the AIT teams training inside the same company is very high. Several times the training activities must restart from scratch jumping from a program to another. And at the end, the usual test engineer sentence is "It is not exactly what we need".
The Embedded Generic Ground Server (EGGS)
For the reasons above TAS-I and INTECS Milano are studying a new

EGSE Kernel based on SOA paradigm (ReST implementation) with independent data presentation & interaction layers. This new application is able to fulfill the standard satellite data processing needs, but will publish test results in multiple formats to allow any resource (human or machine) to have readable data representations according to their specific needs. A simple example will clarify this concept: during the early stage of satellite integration the EGSE MMI will present the raw telemetry data and the basic parameter windows to verify a correct behavior after electrical integration. During the thermal vacuum campaign an incremental test report is what is required. Of course in both cases the data server archive can be used to extract any data requested and to present the result in the most appropriate format.

Software Technologies and Tools

As previously said, during the last years EGSE has taken advantages from the PC HW performances growth, but the same cannot be said about software technologies evolution.

SOA (Service Oriented Architecture)

The benefits deriving from Service Oriented Architectures are widely known: flexibility, ease of change (maintainability), extensibility, reuse, etc.. Nevertheless in some cases traditional SOA implementations (SOAP, WS-*, ...) aren't simple enough. For this reason the API of our new DB is based on ReST (Representational State Transfer) which moves the SOA concept a step further, just like OOP did with structured programming, proposing an healthy convention and forcing the programmer to distinguish between the subject (resource in ReST terminology) of the operation and the operation itself in a cleaner way. Also allows the server to "speak" many languages as they are needed by the client (HTML, PDF, XML, Binary, etc.) enabling contents fruition by different kinds of clients (GUI, CLI, Web Browsers, ...).

MVC

This SOA approach allows us to rethink the application structure in terms of MVC. MVC is a widely known pattern for graphic user interfaces, it helps separation of concerns giving to each element its own place. With MVC we typically have a data backend (Model) which owns the business logic, the graphical front end (View) which is responsible of displaying informations to the user and to gather input, and an orchestrator between the two (Controller) which routes inputs and outputs to and fro models and views.

Non Relational DataBases

Finally, as data beck-end we're evaluating the use of non-relational databases (aka document based) which will help in unclutternig the process of database preparation, and would allow us to define a simple (yet complete) DSL (Domain Specific Language) for data structure, not bound to a specific technology.

3 15:00 Future Proofing EGSE - Is Virtualization the Answer?

Patrick, R.M.[1]; Armitage, A[2]; Leadbeater, K.[2]; Polverini, A.[2]
[1]Terma A/S, DENMARK;
[2]Terma B.V., NETHERLANDS

This paper looks at how some newer developments in the information technology sector can be applied to EGSE/CCS systems. The aim is to improve the lifetime of systems, reduce the operations costs and improve the flexibility of Central Checkout Systems. This would apply particularly when addressing the needs of long term programs (Galileo), production lines (also Galileo) or when a CCS is considered part of an infrastructure to support many missions over a long period.

The technologies in question include:

- use of virtualization - both in the development and deployment of the systems
- use of blade servers to provide a very high availability and efficiently

packaged computing resource
- use of thin clients to ease the system management overhead of these systems

These technologies were first investigated by Terma when addressing the needs of a large production line i.e. Galileo FOC AIT/AIV. However, the feasibility studies and prototypes developed by Terma were so successful that they have also been deployed on systems to support other missions as well.

This paper will look at the three technologies identified above, describing the benefits of applying them in the CCS environment: Some examples include

- Virtualization - allows old systems to be run on new hardware, thus allowing users to benefit from the latest hardware without the need for software upgrades to achieve it. It also allows developers to rationalize dramatically the infrastructure they use for developing and maintaining systems, thus keeping maintenance costs down.
- Blade Servers - coupled with the correct disc drives and network topology, this increases the availability of systems, with many hot redundant features that are available at modest cost. They also can be deployed in "dense" configurations - which is important where space can be ata premium - such as in crowded clean rooms
- Thin clients - these replace traditional workstations as the MMI to the system. The system management overhead of these is far less, thus simplifying their deployment.

From this, we have identified a number of different deployment alternatives that ensure that the CCS is "right-sized" for its designated use. The operational software (SCOS-2000 et al) can run unmodified on a great range of computer hardware and this feature is exploited here.

The use of SCOS-2000 also helps with the compatibility thoughout major parts of the lifecycle of the mission as already proven on previous missions.

Having been working with these technologies now for 2 years, including deploying systems for operational use, there were of course some problem areas, and it is worth to highlight these as well.

We conclude that this approach is a major step forward for both developers and users alike. We also identify other space applications - namely Mission Control Systems - where we believe the benefits are even greater.