BENEFITS OF MODULAR ARCHITECTURE FOR EGSE SYSTEMS AND REMOTE TESTING SUPPORT

Massimiliano Mazza [(1)], Nieves Salor Moral [(1)], Simone Dionisi [(2)]

[(1)]Vitrociset Belgium, permanent establishment in the Netherlands
'sGravendijkseweg, 53 - 2201CZ Noordwijk (NL), +31(0)71 3649770
Email: n.salor_moral@vitrocisetbelgium.com
m.mazza@vitrocisetbelgium.com

[(2)]Vitrociset Belgium, permanent establishment in Germany
Lise-Meitner-Strasse 10, 64293 Darmstadt (Germany), +49(0)6151 9573415
Email: s.dionisi@vitrocisetbelgium.com

ABSTRACT

Check Out solutions vary in size and complexity depending on the scope of the unit under test. EGSE industry traditionally addressed different scopes with heterogeneous ad hoc approaches.

Vitrociset comes from a long history of cooperation with ESTEC in proposing the advantages of the standardisation at all levels and is in the process of transferring this heritage and the approach to the commercial market proposing standard check out systems coping with all exigencies, compliant with relevant standards and providing evident advantages in the life-cycle of the integration, testing and verification campaigns from subsystems to spacecraft.

Thanks to the modular and scalable philosophy adopted in the re-definition of the ASE family of applications the possibility will be provided to assemble spacecraft check-out systems simply connecting subsystem EGSEs (ASE based) to an ASE based platform EGSE allowing:

- an immediate and painless transfer and reuse of subsystem's testing know-how together with its procedures and its database.
- the guarantee to reuse equipment developed for subsystem validation at higher level
- a significant reduction of the test definition phases
- a significant improvement in the knowledge transfer process between different test teams.

Due to this innovative approach and the many user requirements driven features, ASE based solutions aim at becoming a commonly accepted and used layer to the most of the check-out projects implementation.

INTRODUCTION

Vitrociset is in the space systems automation market and standardisation since more than 15 years with projects spinning out of the ASE (Automated Schedule Executor) line of prototypes.

Since the beginning the cooperation with ESTEC and ESOC produced successive versions of the ASE concept tailored either for EGSE or for Mission Automation needs, under the more and more consolidated assumption that Testing & Validation and Operations are homogeneous activities.

The deployment of an instance of ASE family (TPE) as front end of the VEGA-EGSE Check-out system confronted for the first time the tool with the need for modularity. The VEGA-EGSE is a complex system made of three identical and inter-changeable chains (dedicated to the testing and verification of three launcher stages) with the possibility to be merged into a super-chain for the testing of the whole assembled launcher. A strict requirement was the possibility to drive all EGSE components by means of a unique centralised GUI also because user console could be 4 Km away from the SCOEs during firing tests.

While implementing this project Vitrociset matured the necessity of supporting this kind of need in a structured and embedded way, aiming at providing, in the end, real plug-and-play modularity in Check-out infrastructures.

DEVELOPMENT OF A MODULAR CONCEPT

A Modular Test Environment

Vitrociset is already capable of importing branches of Space System Model in order to incorporate different but compatible test environments into a master test plan. In the current configuration the biggest challenge is indeed to ensure the compatibility, what we currently do in post processing by means of a Model Integrity Tool (MIT).

This approach is not only the solution for huge infrastructures like the VEGA EGSE. The same kind of challenges are currently exploited in all average complexity space programs. For this reason the aim of our current development is to implement the capability to assemble complex check-out infrastructures in a plug and play fashion from sub units already behaving like check-out systems on a lower scale.

This solution is especially effective when dealing with the check-out at system (space-craft) level. Generally the spacecraft test equipment, during system integration, may need inheriting equipment from subsystems (or payloads) EGSE together with the know-how on their testing and operations. This migration phase is often painful and extremely expensive in term of man-hours for data porting and validation of interfaces. It normally introduces delays and unforeseen costs.

Since the latest releases, ASE products handle as coherent compound of information what we call a test environment (or an operational environment, depending from the context) built by a set of:

- Database tables (ODB, OBSW, SSM, configuration data)
- Procedural scripts (70-32)
- Infrastructural scripts (non 70-32, mostly shell/batch scripts)

- Settings files

This approach ensures that a test environment, as long as undivided and developed in a controlled and coherent way, is by all means executable on a defined target configuration. With this pre-requisite importing a test-environment of a subsystem during integration will imply having access to all validated test experience and procedures within the master test environment of the integrated system with minimal efforts.

The key to the implementation is the Space System Model. Infrastructures designed around the concept of SSM are conceived to deal with the information from an abstraction layer. Procedures execute activities defined in the SSM handling information accessible through SSM reporting data and/or events.

Activities themselves are defined in several ways depending on the implementation driver (TC, bus commands, OS commands, other PLUTO procedures, SCOE directives…) but appear to the procedural environment as the same kind of object (indeed "the activity"), therefore are managed in the same way

```
…
Initiate and confirm Send1553msg of 1553SCOE of EGSE with
      RT := 8,
      Msg := 0x0;
Initiate and confirm launchBrowser of OpSys of Infrastructure with
      Browser := "firefox",
      Page :="$HOME/mimics/1553spy.html";
Wait for P010101 = 0x01;
Initiate and confirm OpenAND of S2K of EGSE with
      AND_ID := "Power_N";
…
```

Figure 1: Sample PLUTO code

So in our definition test environments are self-consistent entities encapsulated in an undivided structure and accessible through a hierarchical representation.

Therefore the act of importing a subsystem test environment simply become adding a branch, related to the selected subsystem, to the master environment's tree, having access to validated subsystem command sequences as activities from the system.

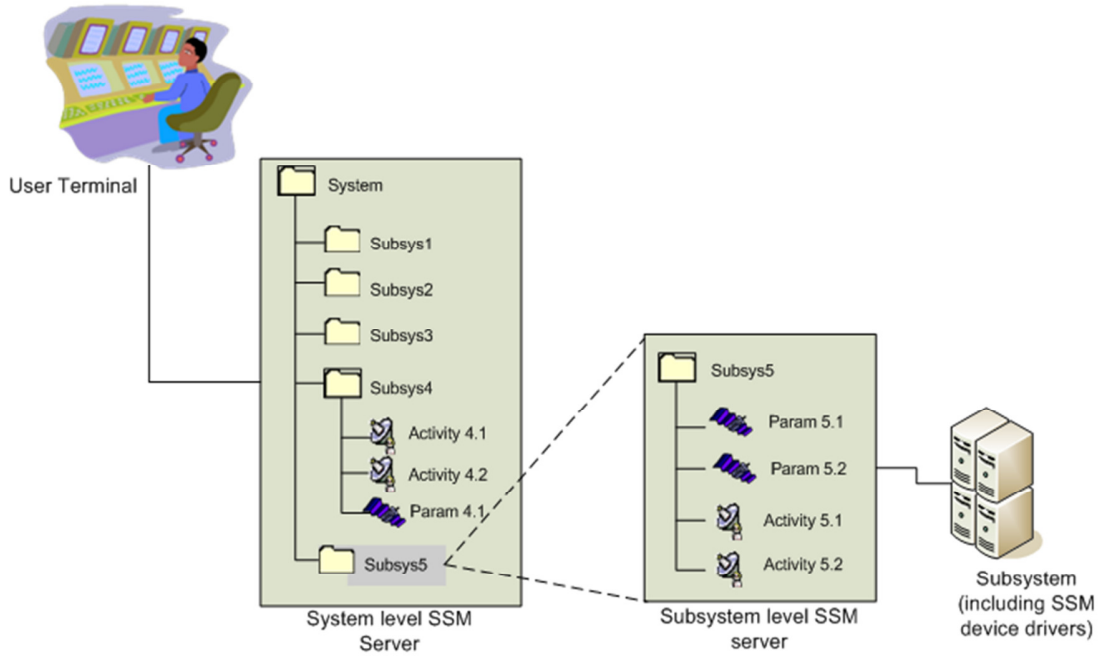

Figure 2: Distributed SSM architecture

It should be stressed that the effectiveness of the method can only be maximised by the rigour and the structured approach in the definition of subsystem procedures as macro activities to be called from a master test procedure at system level.

The approach may seem at a first glance restrictive. In our view it is not. The only two constraints to be respected to achieve the full functionality are:

- All subsystems are configured, modelled and operated by means of a compatible Space System Model data base
- A subsystem's real-time SSM is able to accept remote directives for activity execution and requests for data notification granting the same functionality as from their local interface.

The advantages obtained in return are by far bigger:

- re-use of the subsystem check-out infrastructure as peripheral EGSE of the master check-out with minimal effort
- migration of the subsystem's state of the art database as-is provided by the supplier with minimal effort
- Acquisition and incorporation of the know-how and experience achieved by the suppliers during the verification and validation of their subsystem with minimized impact.

The architecture will allow either importing the subsystem's SSM permanently and in a static way into the master SSM either simply connecting one to the other so that the master SSM will have a sync mirror of the subsystem's status.

A Modular Architecture

Also in the software design our architecture is devoted to scalability and modularity, and follows the implementation guidelines of the SOA.

In order to achieve scalability, maintainability and robustness against future extensions, modifications or technology needs, the ASE-5 system is designed following a decoupled and distributed approach. SOA has been chosen as the architecture to carry it out by using self-contained components able to interact not only between them but also with external systems (e.g. provide or request information) via standardized services. These services will be described and published in a catalogue and, if wanted, may be replaced by proprietary implementations. This approach allows great flexibility to the system as it will be completely technology independent. Another advantage is the unique way to communicate components as local or remote communications are dealt the same way via the service discovery or routing, which will be configured once in the ASE-5 system.

The core of the architecture relies in the Space System Model, as a Runtime Server instantiated and initialised from a static SSM, populated to model the complete world we plan to deal with.

The main classes of components for the architecture are:

- Client applications, including executors, displays, services…
- The SSM server(s)
- The device drivers, encapsulated in a System Element software wrapper

In this way the only communication interface between the components to be implemented is SSM-to-SSM introducing enormous simplification and providing every possible means for optimising the communication performances.

The RT-SSM server, differently from previous implementations, is an independent module executing its tasks at the highest priority. These tasks are mainly the initiation of activities (i.e. instantiation of the actions foreseen on the drivers implementing the activity), the collection and distribution of information (reporting data, activity execution status) and collection, generation and distribution of events.

The SSM is instructed on how and when to initiate an activity by means of a directive that can be received on a dedicated interface (activity initiation interface) and exposes on the data notification area data collected from the drivers on a notifications interface.

The system is completed by services interacting with the SSM on these interfaces. Communications are vehiculated by a naming server allowing intrinsically the possibility for distributed systems.

The advantages of such a modular architecture go beyond the achievement of properly designed software. The development of the capability of SSM-to-SSM interaction in real-time provides users with a next-level spectrum of deployment configurations not known presently in this kind of operational environment.

What we refer to is actually a geographically distributed operations/testing capability with intrinsically reduced setup effort achieved by dynamic SSM-to-SSM mirroring over remote connections.

Such capability could be exploited for instance in preliminary functional tests previous to subsystems shipping to integration facilities, or the definition and execution of spacecraft operations making use of multiple ground stations where centralised control is needed.

Obviously this kind of setup may be just targeted for low performance applications, at least with the actual level of technology.

CONCLUSIONS AND FURTHER WORK

The new architecture for the ASE system has been designed trying to prevent being influenced by previous work but benefiting from all past experiences and acquired know-how, the main aim being to achieve the maximum flexibility to cope with different needs and operate in different fields.

User requirements from the Spacecraft AIT and Operations have been considered and targeted for satisfaction in a way to prevent excessive specialisation. This has been achieved by delegating context specific tasks to pluggable modules.

The Space System Model, due to the absolute generality in its use has been maintained as the core of the architecture without any risk of over specialisation. The compliance to ECSS-E-70-31 is maintained.

Communication protocols within the architecture are standardised to achieve the maximum possible optimisation and performances, and specialised communication (e.g. to the specific device drivers) have been localised on the devices themselves (possibly moving their responsibility to the device provider).

ASE-5 inherits from its predecessors the vocation to both AIT and Ops applications integration. This vocation is exploited as never before due to the matured experience and to the new wave of interest towards the search for synergies and commonalities between the two environments. For this reason ASE-5 could be very well considered an automation component candidate to be selected among the available options of the Common core suite.

The development of ASE-5 is on-going and its first release is due in 2013, nevertheless many of the interesting features will work already as extensions to the current products.

ASE-5 will be as well scalable, meaning that light versions can be assembled to operate simpler environments (SCOE interface, payload operations GS automation). Obviously sub-systems developed with an ASE-5 light interface will inherently provide full compatibility for integration into main ASE-5 driven systems.