Open Virtualization Framework for Testing Ground Systems

Nuno Duro
Evolve Space Solutions
Est. Paço do Lumiar, Lt 1
Lisboa, Portugal

nuno.duro@evolve.pt

Rui Santos
European Space Agency
Robert Bosch Strasse 5
Darmstadt, Germany

rui.santos@esa.int

João Lourenço, Hervé Paulino, João Martins
CITI – Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

{joao.lourenco, herve} @di.fct.unl.pt

citanul86@gmail.com

ABSTRACT

The recent developments in virtualization change completely the panorama of the Hardware/OS deployment. New bottlenecks arise in the deployment of application stacks, where IT industry will spend most of the time to assure automation. VIRTU tool aims at managing, configuring and testing distributed ground applications of space systems on a virtualized environment, based on open tools and cross virtualization support. This tool is a spin-off of previous activities performed by the European Space Operations Center (ESOC) and thus it covers the original needs from the ground data systems infrastructure division of the European Space Agency. VIRTU is a testing oriented solution. Its ability to group several virtual machines in an assembly provides the means to easily deploy a full testing infrastructure, including the client/server relationships

The possibility of making on-demand request of the testing infrastructure will provide some infrastructure optimizations, specially having in mind that ESA maintains Ground Control software of various missions, and each mission cam potentially have a different set of System baselines and last up to 15 years. The matrix array of supported system combinations is therefore enormous and any improvement on the process provides substantial benefits to ESA, by reducing the effort and schedule of each maintenance activity.

The ESOC's case study focuses on the development and validation activities of infrastructure or mission Ground Systems solutions. The Ground Systems solutions are typically composed of distributed systems that could take advantage of virtualized environments for testing purposes. Virtualization is used as way to optimize maintenance for tasks such as testing new releases and patches, test different system's configurations and replicate tests. The main benefits identified are related to deployment test environment and the possibility to have on-demand infrastructure.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; D.4.5 [**Reliability**]: Fault-tolerance; D.4.7 [**Organization and Design**]: Real-time systems and embedded systems; J.7 [**Computers in Other Systems**]: Command and control.

General Terms

Management, Performance, Reliability, Experimentation, Security, Standardization.

Keywords

Virtualization, Infrastructures, Applications, Management, Configuration, Deployment.

1. INTRODUCTION

ESOC is responsible for performing all operations regarding the satellite control for all ESA missions. This means managing a complex array of Systems along with their independent development lifecycles. In order to find a common approach (as far as possible) for ESA missions, a ground data system Infrastructure is currently in place to take care of all aspects related with the development and maintenance of most of these Systems.

The development and validation of the aforementioned ground data system infrastructure at ESOC clearly exposed the limitations of the traditional approach of having several distinct physical workstations available for executing the applications and many activities. The main observed issues were:

- The proliferation of software baselines and respective hardware resources;

- The low level of utilization of each individual available resource (e.g., CPU usage average per year);

- The fast paced OS market lifecycle compared with mission/infrastructure schedule and longevity requirements; and

- The duration of system test validation and deployment, due to environment and system complexity.

These are crosscutting issues, spanning areas as distinct as the aforementioned space systems, but also telecommunications and services, among others, and have, thus, become one of the major concerns of IT industries. A study conducted by ESOC has concluded that the market trend to address them is virtualization.

Virtualization techniques have been around for quite some time; however the IT massification only occurred recently with the advent of low cost multi-core processors, hardware virtualization support and a wider Operating Systems' support. Nonetheless, ESOC's study has demonstrated that the critical requirements for virtualizing its ground system-testing infrastructure, such as strong isolation between virtual machines, and high system availability and performance are covered/supported by the main virtualization vendors.

However, new bottlenecks have arisen in installation and configuration of application stacks, where it is foreseen that IT industry will spend most of time and money to achieve automatic machine deployment. How to improve the virtual machine duplication, configuration and deployment (along with lifecycle management) capabilities of the existing tools to meet the requirements remains a challenge.

By anticipating these application deployment and automation issues the European Space Agency (ESA), in a joint consultancy with Evolve Space Solutions, has been developing for the last two years a custom solution to automate this process, since no product was identified and available at that time. Evolve has re-structured this solution into the VIRTU framework [12] (see Figure 1) in partnership with ESA, HP Labs, Universidade Nova de Lisboa and Universidade de Coimbra.
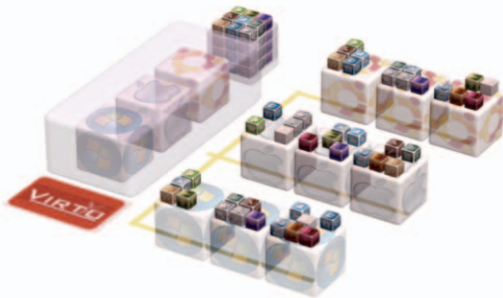

Figure 1. The VIRTU Framework

The framework focuses on ESA needs for testing and developing distributed software, including ground systems, considering that ESA maintains various missions during long periods (up to 15 years), and each mission requires several ground centralized and distributed applications, where sometimes a dedicated machine per application is needed with heterogeneous hardware/operating system. VIRTU now constitutes a viable alternative to ESA ground systems maintenance on this new age of computer availability and green IT.

The contribution of this paper is thus on the proposal of a framework for managing, configuring and deploying Virtual Machines instantiated which use a specified OS and a set of Application Stacks. This framework has a strong potential for application to different industry activities and, in this paper, we describe its ap-

plication to setting up the testing environments in the European Space Agency (Ground Systems).

The remaining of this paper is structured as follows: the following section presents the main features of the VIRTU framework; Section 3 describes how VIRTU can provide a virtualized infrastructure for the testing and developing applications in complex distributed environments, namely in the context of ESA's Ground Systems; Section 4 compares VIRTU with the current state-of-the-art in the management of application stacks on virtualized environments, and finally; Section 5 presents our conclusions and paths for future work.

2. THE VIRTU FRAMEWORK

The VIRTU framework aims at managing and configuring application stacks on a virtualized environment. It introduces an innovative approach for configuring and automatically deploying fully configured virtual machines with applications, independently of the virtualization provider.

VIRTU enables the handling of various software configurations, such as applications, operating systems and networks. It generalizes the configuration of customer applications based on open standards and tools, and supports cross virtualization environments, namely the VMware and Xen hypervisors. The actual configuration is based on open script programming, backed-up by a smart editor that automatically acquires and manages configurations.

Virtual machines in VIRTU are constructed by assembling Building Blocks (operating systems and applications) whose configuration is specified in special purpose files, named Publication Files (Figure 2).
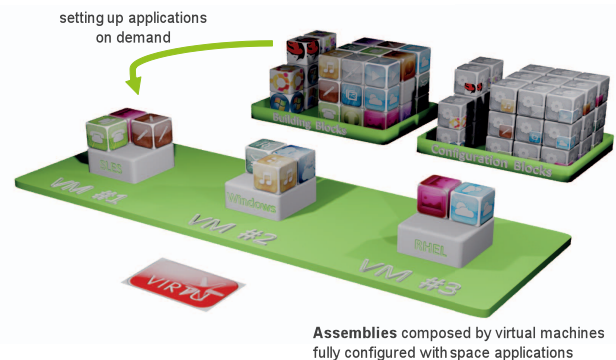

Assemblies composed by virtual machines fully configured with space applications

Figure 2. Configuring and managing applications

The tool provides three main functionalities to be used by final users and system administrators:

- Assembly Configuration – allows the specification of virtual machine configurations based on Building Blocks and configuration blocks (the Publication Files). An assembly contains one or more virtual machines depending on the aforementioned configuration blocks.

- Assembly Request and Deployment – provides on-demand requests of fully configured virtual machines.

- Assembly Usage – allows users to connect, share, and manage the lifecycle of deployed virtual machines.

2.1 Deploying Virtual Machines

This section provides a more in-depth description of the three VIRTU functionalities previously introduced, establishing a workflow for the deploying a virtual machine using VIRTU.

2.1.1 Assembly Configuration

The configuration of Assemblies is a task performed by the system administrator. It involves:

1. Defining of Building Blocks and their associated Publication Files and Installation Scripts (Publication Scripts), and;

2. Assembling of these blocks into ready to be deployed Virtual Machines, named Assemblies.

The initial step is to create the necessary Building Blocks composed of Operating Systems (Linux or Windows) and applications.

These blocks are then associated to Publication Files specifying the configuration variables (e.g., mode of operation of the application) and the Publication Scripts to install a given Building Block. The configuration process can be automated by using the Smart Application Reconfiguration Tool (SmART) for the automatic acquisition of new Building Block configurations (more on this in Section 2.2).

The second step consists on building the Assemblies by adding Virtual Machines and defining the Building Blocks to be deployed. A Virtual Machine is composed of multiple building blocks. The most common strategy when defining such building blocks is to have one for the OS and another for the specific application. This process is assisted by the UI depicted in Figure 3.
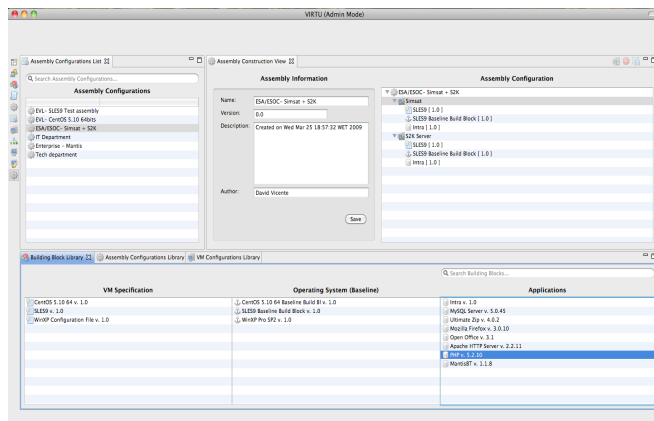

Figure 3. Assembly Configuration UI

2.1.2 Assembling Request and Deployment

The final user may request the deployment of instances of previously configured assemblies, by resorting to the UI depicted in Figure 4. The request is subject to administrator approval, and causes the target assembly to be instantiated and automatically deployed, according to a predefined order and rules.
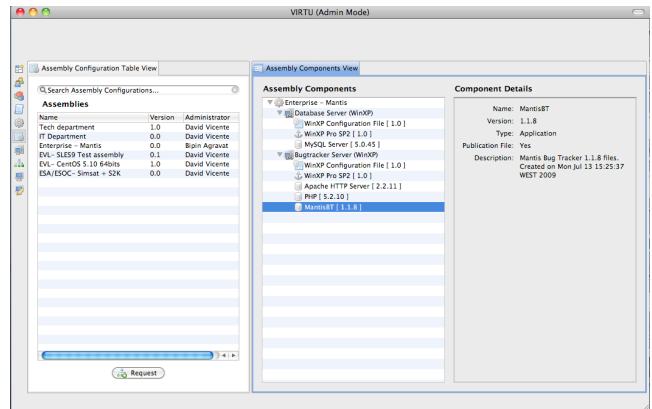

Figure 4. Assembly Request UI

2.1.3 Assembly Usage

The Assembly Usage functionality enables the final user to connect to Virtual Machines and run his applications remotely from the tool (see Figure 5) or using a third party remote connection (VNC third-party support). The users can also share instances with other users and manage the Virtual Machine's lifecycle.
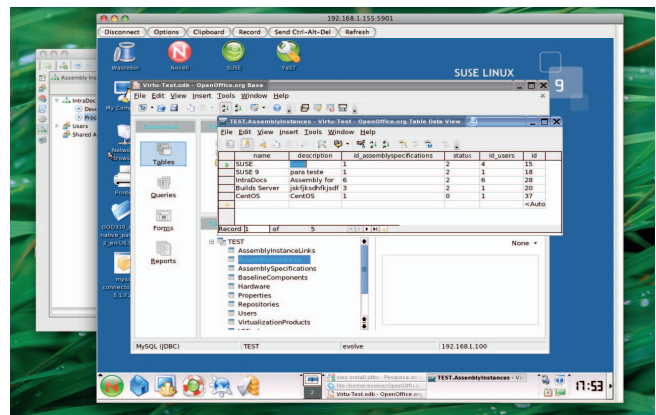

Figure 5. Assembly Instance Usage

In addition to the user interfaces, the VIRTU provides some external interfaces that can be used to ease the integration with a pre-established infrastructure.

The VIRTU tool mainly targets the use cases of testing and developing applications in complex distributed environment, in order to replicate the testing set-up environments.

2.2 Application Reconfiguration

As described in Section 2.1, application configuration in VIRTU is decoupled from assembling, allowing many-to-many relationships. Thus, a virtual machine is only configured when deployed. The Publication Files specifying the configurations of the assembled Building Blocks are retrieved from a pre-determined database and handled by a script, to perform the desired configurations before the system is on-line.

SmART [2], a Smart Application Reconfiguration Tool, is a framework for the automatic configuration of applications. Without compromising its generality, SmART targets virtualized IT infrastructures, configuring virtual machines and their applications. It implements an application configuration workflow, resorting to the similarities between configuration files (i.e., patterns like parameters, comments, blocks, etc.), to recognize the syntax and, to some extent, the semantic of a configuration file, and convert it into a generic (application independent) intermediate representation. Configuration transformation scripts may then be executed over this intermediate representation, with or without administrator support, to generate a customized configuration. This representation is then reconverted to reflect the alterations in the original application dependent syntax. SmART reduces the time required to (re)configure a set of applications, by automating time-consuming steps of the process, independently of the nature of the application to be configured.

2.2.1   Classes of Application Configurations Files
We performed a comprehensive analysis of the configuration files of well known and widely used open source applications, such as Apache, Eclipse, MySQL, PostgresSQL, GNUstep, and Mantis. We limited our study to text-based configurations files for now. As anticipated, the concrete syntax of configuration files tends to differ among applications, but effectively resorts to a limited number of concepts. In fact, only four distinct concepts where identified in all of the inspected files:

- Parameter assignment – set the value of an application configuration parameter;

- Block – group configuration settings;

- Comment – explain the purpose of one or more lines of the file; and

- Directive – denote commands or other directives, such as the inclusion of a file.

Our analysis also focused on the actual representation used by the applications to express these concepts, and although no standard exists, we observed that some formats, such as INI [3] and XML [4], have emerged as community standards. Consequently, we were able to classify all studied applications under the following three main categories:

- *INI-based:* The syntax follows the INI format or similar. Blocks are explicitly initialized, but are implicitly terminated by the beginning of the next block. Assignments are of the form parameter separator value, where value can be either a single value, a list of values, or even nothing. Block-nesting is not allowed.

- *XML-based:* This category encompasses syntaxes a little more permissive than pure XML, and addresses applications that store their configurations as XML variants. This category has a broader scope than the INI-based, since explicitly initialized and terminated blocks can nest other blocks.

- Block-based: This category addresses a wider scope of formats on which are delimited by symmetric symbols, such as { } and ( ).

Fro our analysis, we concluded that systematic and automatic application configuration can be achieved by transforming the application configuration files into a generic representation, detached from the application specifics. This representation can then be modified systemically, and once altered be converted back to the original syntax, reflecting the applied modifications.

2.2.2   The Application Reconfiguration Process
The reconfiguration process using SmART is divided in three stages, as illustrated in Figure 6:

1. Transform a given configuration file from its original representation to a generic (XML-based) format;

2. Modify the XML file to reflect the desired configuration; and

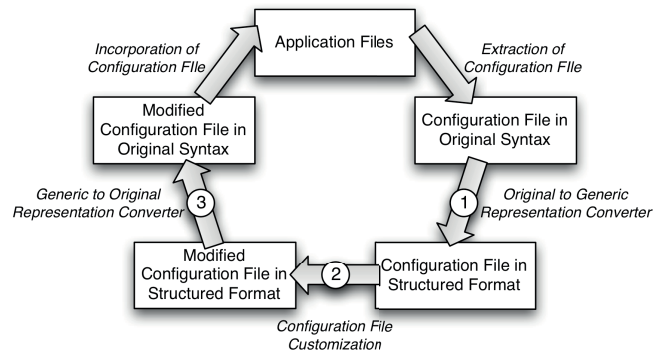3. Translate back the modified XML file to the original representation for the configuration file.

Figure 6. Application Reconfiguration Process using SmART

The stages are independent, requiring only the use of the same intermediate representation. Such design enhances flexibility, for instance: file modification can be performed manually (e.g., a graphic tool that displays all the editable settings) or automatically (e.g., a script); several executions of stage 2 can be performed over a single output of stage 1, and; stage 3 can convert a file back to this original representation with no extra information besides the one included in its input file.

2.2.3   Integration of SmART in VIRTU
SmART integrates with VIRTU at two levels. The *Original to Generic Syntax Converter* is used by the administrator to transpose one or more configuration files into a building block publication file, in order to define the block's default configuration and user editable parameters. The *Generic to Original Syntax Converter* is used in the virtual machine's on-boot configuration process. It converts the building block's publication file back into its original format, so that the configuration script may carry its work.

Dynamic reconfigurations pf running virtual machines can also be achieved by having a daemon on every running virtual machine listening for reconfiguration requests. This process is equivalent to the on-boot configuration process, with the exception that the target application may have to be restarted before the new configuration can be applied.

2.3   VIRTU License, Usage and Availability
VIRTU is distributed under a dual-licensing model, open-source (LGPL v3) and commercial, known as Quid-Pro-Quo business model. The open-source package will be free-to-use and Evolve's

main goal is to develop a widespread user community and enable its adoption as a standard for virtual machine application stack deployment. The expectation is that this community will provide in return free testing, free enhancements and most importantly free marketing. While maintaining the solution in open source assures a wider dissemination and acceptance, private source will assure the competitive advantages where customers wishing to use commercially, alter or bundle the VIRTU product must purchase it through a commercial license. The final business solution follows a balance between open source and closed source software.

3. A CASE STUDY: ESA

The ESA case study targets the use cases of testing and developing applications in complex distributed environment. The possibility to automatically make multiple deployments is quite interesting for maintenance of Generic System Infrastructure testing and Mission testing. The Virtualization tools are integrated in the general ESA maintenance process as described in figure 7.
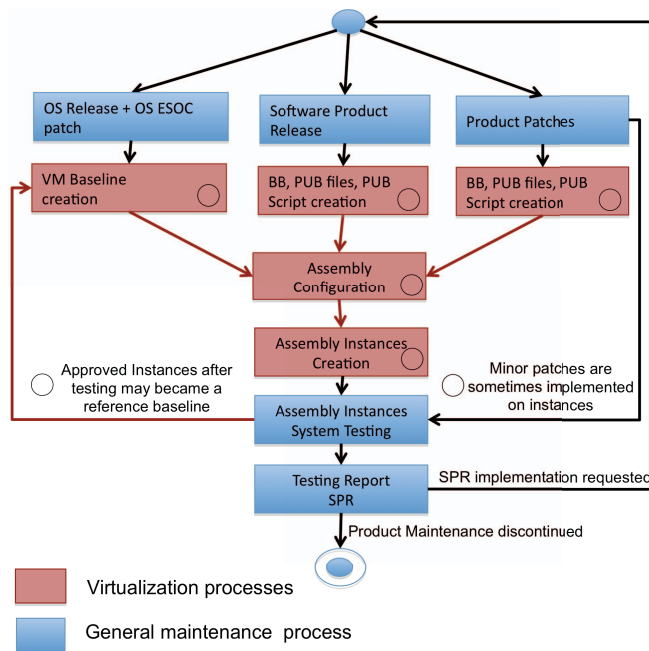

Figure 7 - ESA Maintenance Process with VIRTU

The process described starts with the release of software products (OS, Product, Patches) in binary format (installer and application). Virtualization is used then to setup the environment and testing is performed on the virtual machines instance requested by the developers until the product is discontinued.

3.1 Infrastructure Initial Use Case

The predecessor version of VIRTU (GoVI project [5]) was started in 2006 and deployed in 2009 at ESOC for use within the ground systems infrastructure division. The GOVI connector provides a centralized and common access to Virtual machines depending on user privileges (one user only has access to his own Virtual Machines or the ones shared with him by other users). It also allows the users to perform Virtual machine/Assembly requests, which will be later authorized and deployed by the GOVI administrator. The GOVI key functionalities have been kept and improved in VIRTU.

On this first phase, the typical scenarios considered were the deployment and testing of new systems and patches to existing systems on a virtualized environment composed by one or more virtual machines (partially implementation of step 1 and 3 of Figure 7). This accelerates and improves the software acceptance process by ensuring a clean environment baseline is used for each activity.

Another important scenario is the use and deployment of common Virtual Assemblies, following an OS ESOC specific baseline, for each Project in order to harmonize the use of Virtualization (partially implementation of step 2 of Figure 7).

In both scenarios, the portability provided by the creation of reference assemblies allows for a much wider range of end-to-end System test cases (where a "black box" Virtual assembly can be used without a huge amount of effort).

It needs to be pointed out that in this first stage the process of configuring new assemblies is still manual and requires a relatively high level of knowledge on the system to be installed. However once the Assembly is available the process of creating multiple instances of this reference assembly is automated (step 6 of Figure 7). The next logical step is therefore to improve the tool in order to perform the System deployment and configuration (which will lead in the end to a new reference assembly) based on a limited number of input parameters.

3.2 Infrastructure Use Case

The extension provided with VIRTU is a consolidation of its predecessor, GOVI project, providing better customization and flexibility. It is more focused on the request and deployment of multiple (possibly distributed) systems and testing multiple configurations on the generic ground systems infrastructure.

The ESA Ground System products are organized in blocks to be installed on different testing assemblies as presented in Figure 8. Some of these blocks have to be manually created and costumed for the first time (step 1, 2 and 3 of Figure 7). However once available they can be re-used by different Assemblies and even different instances of the same Assembly (step 4 of Figure 7). This allows for a much faster and uniform system test integration and deployment. It also ensure that a pre validated block (application) is not going to be misused introducing another level of complexity and origin of problems.
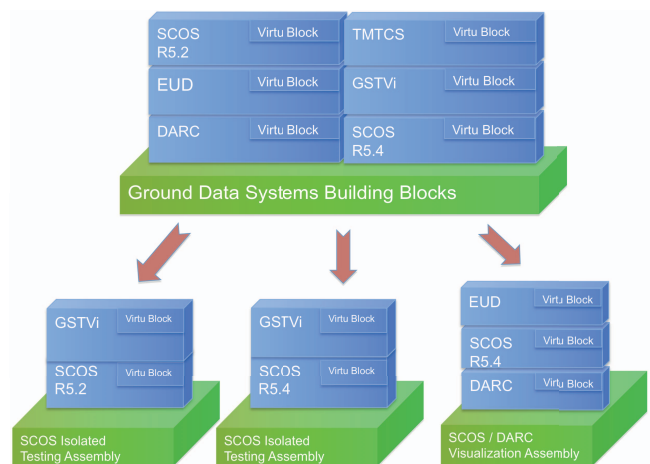

Figure 8. VIRTU Use Case evolution in OPS-GI

The figure presents several VIRTU blocks configured with various GS products, namely SCOS R5.2 and 5.4, Egos User Desktop, Packet Archive (DARC), GSTVi and Telemetry and Telecommand System (TMTCS). Together with these blocks are also configured Off-The-Shelf third-parties (e.g., My SQL) represented in the figure as VIRTU Block.

All these blocks have to be created and configured for the first time following a manual process or assisted by the SmART tool. However once available they can be re-used by different Assemblies and even different instances of the same Assembly. This allows for a much faster and uniform system test integration and deployment. It also ensure that a pre validated block (application) is not going to be misused introducing another level of complexity and origin of problems.

The possibility of making on-demand request of the testing infrastructure (step 5 of Figure 7) will provide some infrastructure optimizations, specially having in mind that ESA maintains ground control software of various missions potentially on different System baselines for long periods. The matrix array of supported system combinations is therefore enormous and any improvement on the process provides substantial benefits to ESA (by reducing the effort and schedule of each maintenance activity).

The use case also takes into account the scenario where there is no building block or Assembly for a particular system (very common at the initial stage of the development). In this case, a "raw" Assembly containing only the required OS block and third party software is deployed. The system is then deployed, configured, sometimes patched (step 7 of Figure 7) and validated. Once completed, a new building block is created and used as a reference for any future activities (step 6 of Figure 7).

Another use case is the deployment of products in multiple Virtual Machines assuring the setup communications between them. A common example is depicted in figure 8 deploying the SCOS Testing Assembly products in two virtual machines that needs to communicate between each other: one with the Mission Control System (SCOS-2000) and the second with the mission spacecraft model integrated in GSTVi (SIMSAT and Simulation Models).
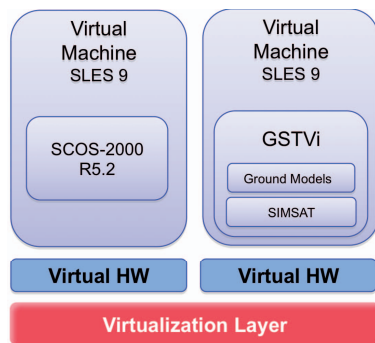

Figure 9. Typical GS testing scenario

In fact, this end-to-end testing may results from two Assembly Instances previously approved and promoted as reference baselines (step 6 of Figure 7).

Another benefit of the blocks approach can be seen on future study/prototype activities where different block combinations can be tested without much effort. The typical example would be to

deploy a SCOS-2000 block into a different baseline OS and evaluate the potential impacts of a future migration activity.

3.3 Mission Use Case

VIRTU can be used to support testing on the Ground System's Missions (this means the "blocks" concept described before is still valid for the mission validation approach). Conceptually there are not many differences between the Infrastructure Use case and the Mission Use case. The various product versions are also organized in blocks as depicted in Figure 10.
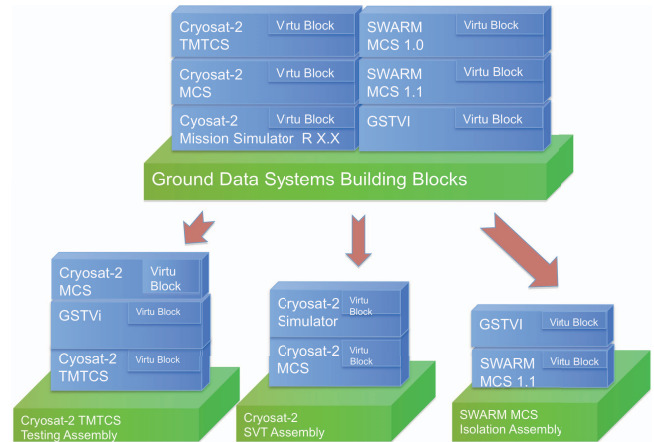

Figure 10. VIRTU Use Case on OPS-GD

The figure illustrates two mission scenarios (Cryosat-2 and Swarm):

- VIRTU Blocks could be configured with several product versions (e.g., Swarm MCS 1.0 and Swarm MCS 1.1);

- Testing results comparison between product versions baselines may be performed.

Each VIRTU Block can be used on several assemblies, where each assembly has its own testing purpose, for instance:

- Cryosat-2 TMTCS Testing Assembly is used for validations of MCS and TMTCS using GSTVi;

- Cryosat-2 SVT Assembly is used with MCS and the simulator for System Validation Testing.

In both infrastructure and mission maintenance, virtualization is used as way to optimize maintenance of ESA ground systems infrastructure for tasks such as testing new releases and patches, test different system's configurations and replicate tests. The main benefits identified are related to test scenarios deployment and the possibility to have on-demand infrastructure.

4. RELATED WORK

Being VIRTU a virtualization framework that provides the systemic management and configuration of application stacks on virtualized environments, this section addresses related work in both the virtualization and the systemic configuration fields.

Regarding virtualization, VIRTU constitutes a step forward from the current proposals, such as VMWare [6], Xen [7], VirtualBox

[8], and so on. In fact, VIRTU is not a virtualization provider. Instead it provides for configuring and automatically deploying fully configured virtual machines, independently of the virtualization provider. Currently, VIRTU supports VMWare and Xen images, but, with the general adoption of the Open Virtualization Format (OVF) standard [9], compatibility will no longer be an issue.

Moreover, VIRTU is a testing oriented solution. Its ability to group several virtual machines in an assembly provides the means to easily deploy a full testing infrastructure, including the client/server relationships.

Regarding application configuration and deployment, to the best of our knowledge, VIRTU's SmART is the first to exploit the similarities among configuration files to allow for automatic, vendor-independent and on-the-fly application reconfiguration. Thin Crust [10] and SmartFrog [11] are the projects that more closely related to our work, but there is no effort on automating the configuration process.

Thin Crust is an open-source set of tools and meta-data for the creation of virtual appliances. It features three key components: Appliance Operating System (AOS), Appliance Creation Tool (ACT) and Appliance Configuration Engine (ACE). The AOS is a minimal OS built from a Fedora Kickstart file, which can be cut down to just the required packages to run an appliance. SmartFrog is a framework for the creation of configuration-based systems. Its objective is to make the design, deployment and management of distributed component-based systems simpler and more robust. It defines a language to describe component configurations and a runtime environment to activate and manage those components.

5. CONCLUSIONS

We described VIRTU, a framework for configuration, deployment and management of the life cycle of Virtual Machines. VIRTU has a wide range of potential uses in industry, but has been driven mainly by the needs of the Ground Systems sector of the European Space Agency, who have contributed to the requirements analysis and validation of the framework.

By resorting to virtualization, VIRTU makes possible the on-demand request of the testing infrastructure and allows infrastructure optimizations when dealing with a large number of combinations of hardware, operating system and application stacks. It also substantially improves the deployment of end-to-end test facilities and demonstration scenarios. We describe two use cases in ESA, where the matrix array of supported system combinations is enormous.

VIRTU process can bring great advantages when deploying the systems several times like the Ground System testing performed by ESA

- Manual deployments are effort intensive;

- Preparation of the deployment process in VIRTU involves a certain level of customization (creating the scripts) but is of the same order of magnitude as performing the setup manually

VIRTU deployment is a clear advantage when you need to repeat the same deployment tasks on a higher level of magnitude. Naturally the first VIRTU setups take a bit more configuration effort but become an advantage in terms of effort and reliability when various setup tests need to be performed.

Possible extensions on testing may allow to fully configuring specific blocks for performing regression testing the Ground Systems. GUI Testing framework block is typical example that can be use to automate regression system testing.

The product may also be used on other customers where the use cases are also driven by the need of reducing costs on system deployment and configuration.

ACKNOWLEDGMENTS

6. REFERENCES
[1] Technical Note 6: Final Report - Consolidation of Ground Segment Using Virtual Machines, Technical Report UC-ESA/2007.10.05/1 2007.10.05, 2007

[2] Paulino H., Martins J.A., Lourenço J., and Duro N., SmART: An Application Reconfiguration Framework, In Proceedings of Complex Systems Design & Management (CSDM) 2010, Lecture Notes in Computational Science and Engineering, Springer-Verlag, to appear

[3] Cloanto, Cloanto implementation of INI file format http://www.cloanto/specs/ini/, 2009

[4] W3C, Extensible Markup Language (XML). http://www.w3.org/XML/

[5] Use of Virtualisation Techniques for Ground Data Systems, SpaceOps, 2008

[6] VMware, VMware Virtualization Software for Desktops, Servers & Virtual Machines for a Private Cloud, http://www.vmware.com/

[7] Xen, Xen Hypervisor, http://www.xen.org/

[8] Oracle, VirtualBox, http://www.virtualbox.org/

[9] Distributed Management Task Force, Inc., Open Virtualization Format Specification (version 1.0.0), DSP0243, 2009

[10] Thin Crust: Thin crust main page. http://www.thincrust.net/

[11] Goldsack P. et al: The SmartFrog configuration management framework. SIGOPS Oper. Syst. Rev. 43(1), 16–25, 2009

[12] VIRTU, http://www.evolve.pt/virtu