AEOLUS FLOORS THE MOVE
OF ESOC MISSION SPACECRAFT SIMULATORS TO LINUX

D. Guerrucci[1], V Reggestad[2]
ESA/ESOC, Robert Bosch Strae 5, 64293 Darmstadt, Germany

Filipe Metelo[3]
MakaluMedia GmbH, Robert-Bosch Straße 7, 64293 Darmstadt, Germany

ESOC (European Space Operations Centre) has been investing a big effort in developing and maintaining a software infrastructure layer covering the common functionalities of Mission Data Systems required by each mission. In the case of Spacecraft Simulators, the availability of reusable Ground Station Models, Generic Environment Models, Simulator Common Services and other infrastructure products today embraced under the name of SIMULUS has allowed for a big reduction in software development costs.

Recently, in response to wide strategic requirements, ESOC ported the SIMULUS and other infrastructure products (e.g. Mission Control System) from either Windows or Solaris operating systems to the Linux platform.

The Aeolus mission had to start the development process at the time ESOC was completing the porting of Simulator infrastructure from Windows to Linux and it had to decide either to remain with the Windows infrastructure or taking the opportunity of being the first mission to use the new Linux infrastructure, with its associated risks. The mission took a safe approach and the Simulator was developed on both platforms until a checkpoint where an accurate comparison analysis could determine the best way forward. The acquired experience was well documented at the benefit of future missions willing to repeat the process.

The design phase for the Aeolus Simulator concentrated first on the identification of the differences between the two platforms so that these could be isolated and abstracted from the operating system. A big role had the development of spacecraft subsystem models independent from the target platform, achieved through the definition of a set of interfaces allowing spacecraft models being deployed on both platforms with almost no modification.

Once the simulator on Linux platform achieved satisfactory results, the Aeolus Simulator was made only available on Linux. The approach chosen in the development marginally increased the overall costs, but provided the high benefit of reducing the risks to the mission and opening the door to applying the current strategic approach for future ESA missions in the Spacecraft Simulators' area. The ESOC infrastructure development got important feedback and able to achieve stability at an early development stage; other ESA mission, like Cryosat 2 and Hershel-Plank reaped the benefits of the Aeolus experience in the porting process. Also Cryosat 2 is today no longer supported in the Windows version and in addition to the expected benefits has achieved performance improvements on the new platform.

The Aeolus mission and the others that are following this strategy represent a valuable reference for any ESA mission currently willing to follow the trend dictated by the new technology but having already available a spacecraft model on a different platform. The strategy adopted also represents a term of reference for any future mission having to decide between new, promising technology and consolidated but obsolete technology.

NOMENCLATURE

ALADIN	=	Atmospheric Laser Doppler Instrument
CDMU	=	Central Data Management Unit
ESA	=	European Space Agency
ESOC	=	European Space Operations Centre
MCS	=	Mission Control System
MMI	=	Man Machine Interface
SMI	=	Simulation Model Interface
SMP2	=	Simulation Model Portability Standard version 2

[1] Data System Manager, OPS-GDA, damiano.guerrucci@esa.int tel +49 (0) 6151 903138, fax +49 (0) 6151 903105.
[2] Data System Manager, OPS-GDA, vemund.reggestad@esa.int tel +49 (0) 6151 902685.
[3] Software Engineer, OPS-GDA, filipe.metelo@esa.int tel +49 (0) 6151 902981.

SLE = Space Link Extension
TMTCS = Telemetry and Telecommand System
VNC = Virtual Network Computing

I. INTRODUCTION

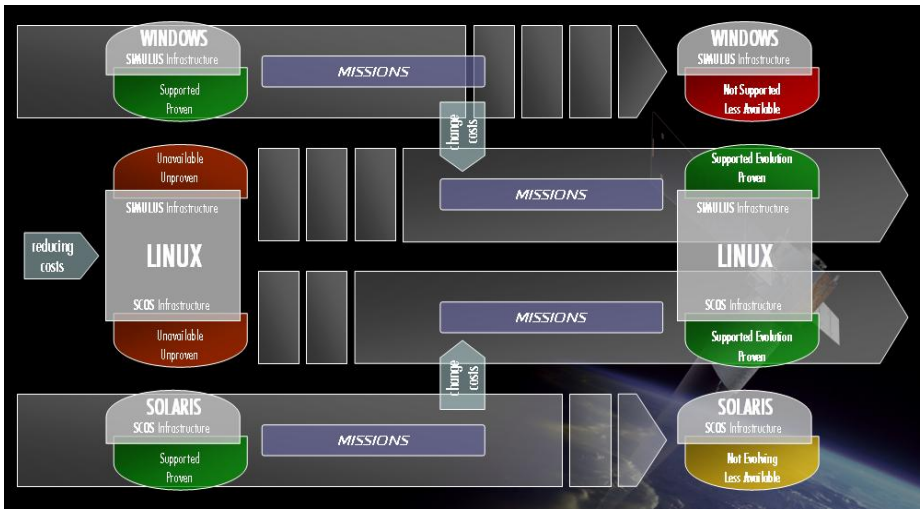
THIS paper describes a successful case where the European Space Agency accomplished the need of evolving it's current Mission Data Systems infrastructure, providing users with the latest technology, and reducing software development costs whilst still maintaining a high level of quality.

In particular, the AEOLUS Spacecraft Simulator in ESOC (European Space Operations Centre) is presented with the development strategy followed in the above context.

II. THE BIG PICTURE

A. The ESOC Transition

In the last decade, ESOC has acquired a very significant experience in the development and maintenance of software for Mission Data Systems serving a wide number of spacecraft. Each Software Data System (e.g. Mission Control Systems or Spacecraft Simulators) developed by the Agency is nowadays based on a software infrastructure layer covering all the common functionalities required by any mission, on top of which a number of specific requirements are implemented that serve only one or a few similar missions. The maintenance of one common infrastructure instead of many specific systems is motivated by the cost reductions resulting from supporting the same software for a wide number of missions running at the same time



In the context of a wide ESA strategy, ESOC has been the moving this layer of common infrastructure products from a variety of other operating systems to a common Linux Operating System. In this way, a common baseline for all Mission Data Systems can exist, leading to all sorts of benefits related to the synergies created by this common approach. The resulting homogenization of the platform across all systems has the clear advantage of reducing costs in several areas like software maintenance, hardware deployment, resource and knowledge sharing. The deployment of systems running on the same operating system provides a drastic reduction in the number of different platforms coexisting in the same environment, with clear benefits also to the maintenance of the entire computer centre.

The process is particularly critical because accumulated experience and evolution from previous systems shall not be lost. The best approach was found in porting the currently used models and systems (so keeping all of the development history) to the new operating system after their proper isolation from the operating system in general.

Concerning Mission Control Systems, this infrastructure layer is embraced under the name of SCOS wich contains functionalities such as Telemetry Processing, Telecommand Encoding Models and others. Following the broader tendency, these common infrastructure products are since some years migrated from Solaris to Linux, however supporting both platforms for the transition period.

Concerning Spacecraft Simulators, the group of common functionalities and generic models is embraced under a real-time satellite simulator infrastructure layer called SIMULUS including functionalities like Ground Station Models, Environment Models, Dynamic Models and Simulator Common Services. SIMSAT is the most important part of the SIMULUS product family covering the Runtime Environment for Simulator models. The generic models included in this software were specifically designed in compliance to the Simulation Model Interface (SMI) from the start and are nowadays (SIMSAT 4) already compliant to the Simulator Model Portability Standard version 2 (SMP2).

B. The AEOLUS Mission

The AEOLUS spacecraft is an atmospheric dynamics mission designed to provide global three-dimensional observations of wind profiles in clear air in the troposphere and lower stratosphere. Equipped with the Atmospheric Laser Doppler Instrument (ALADIN) payload, it will measure atmospheric wind profiles. Aeolus will be in a Sun synchronous, retrograde, near polar orbit such that ALADIN provide a near global coverage. This includes observation of winds at all levels in the lower atmosphere (primarily troposphere and stratosphere) and at the Earth's surface.

As for every other ESA mission, the AEOLUS Mission required the development of a Spacecraft Simulator. In simple terms, this requires a system able to accept and execute telecommands sent to it, e.g. via the Mission Control System through Ground Models, and capable of generating return telemetry. At any point in time, the telemetry shall reflect the status of the spacecraft, to the level defined by the requirements. The most important model in the Simulator, the Central Data Management Unit (CDMU), must also be able to run the real Spacecraft Onboard Software in an emulated ERC32 processor.

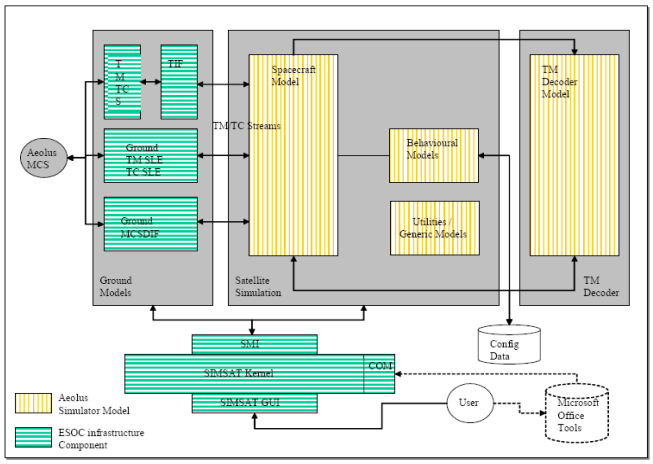


Figure 1. AEOLUS Simulator System Context.

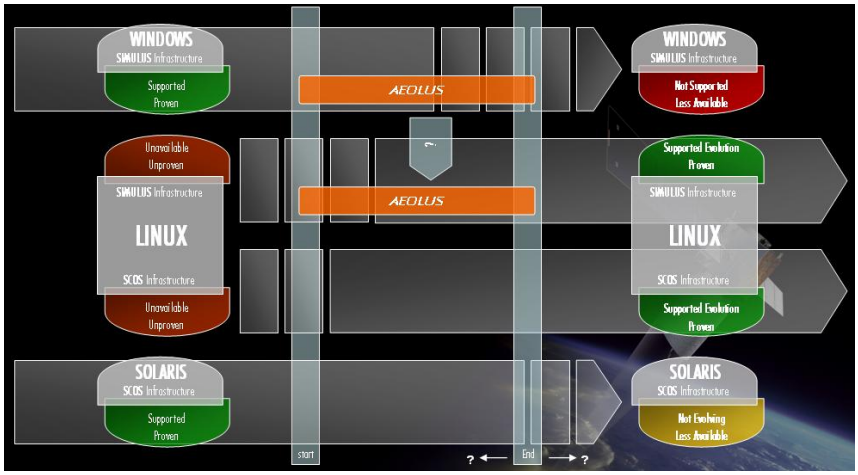
In line with ESA's development strategy, the simulator shall been developed on top of the SIMULUS infrastructure layer which already provides some of the basic and more common functionalities needed for a Spacecraft Simulator. As such, the Simulator uses SIMSAT as a real-time Kernel for the execution of Satellite Models and as graphical user interface to control and monitor a simulation. Generic models such as Ground Models (e.g. SLE and TMTCS models) are also provided to support the simulation of common Ground Station equipment used to connect to the Mission Control Systems. A number of other utilities are also provided as part of the simulation infrastructure SIMSAT, such as Telemetry and Command Modelling, TCP/IP servers for Telemetry frames and packets or breakpoint handlers.

A number of specific functionalities are developed on top of the infrastructure layer to implement Behavioural Models (such as Environment, Dynamics, Thermal, Electrical Network) and Telemetry Decoder Model to extract telemetry parameters from the generated telemetry packets.

III. THE CHOICE FOR AEOLUS

The AEOLUS Simulator development started at the time ESOC was in the process of completing the porting of simulator infrastructure SIMSAT from Windows to Linux. Because of this, the project was confronted with a very important decision between two available options, both valid in terms of requirements satisfaction:

- Remain with the current infrastructure running on the Windows platform, already proven on other missions but surely becoming obsolete at the time of flying;
- Become the first mission to use the new Linux platform, benefiting from the most modern and up to date system but also embracing its many associated risks due to lack of operational use.



The porting of the infrastructure layers to Linux (first 32 and later 64 bits) was not yet fully completed and presented high risk of unavailability at the time that the first delivery of the simulator was required. Even if it was available, other risks were present due to the infrastructure software not being stable enough for operational usage.

On the other hand, the adoption of the current available infrastructure (Windows based) presented a very stable and well known way forward for the mission, although this would compromise the spirit of technical evolution and, more importantly, it would mean the need for specifically supporting an obsolete software over the future years of mission lifetime. This could be critical because the foreseen end of the mission can always be delayed, leaving it more and more inside this period where the Windows infrastructure would no longer be supported.

For these reasons, and because there was no clear advantage yet on following one of the available paths, instead of a firm decision ahead, the mission took a safer approach by developing the AEOLUS Spacecraft Simulator on both platforms (based on both SIMSAT Windows and Linux). This strategy would be pursued until a certain checkpoint was reached (in this case the first delivery of the software) where comparison analysis could determine with more accuracy the best way forward.

A. Parallel Development and Configuration

The above strategy affected considerably the usual design approach. Not only was it required to port all the existing models, configurations, procedures and working methodology (all inherited from previous Windows simulators) to the new target platform, it was also necessary to continue developing the simulator such that any new components would be able to run on the previous platform as well as the new one.

Therefore, the first activity aimed to identify the differences between the two platforms that would affect any aspect of the development. Those indications were required further ahead to make sure that the development process and models produced were always compatible with both platforms and any decision taken regarding the platform to be used should have a minimum impact on either the development cost or time.

The need of keeping both platforms “active and synchronized” required changes in the areas of source code, development environment, emulator, scripting, data displays, configuration control and operational considerations.

1. Source Code Changes

A number of modifications to the source code were also required due to the different infrastructures used. This was because the Windows Visual C++ and the Linux gcc compilers require the use of different syntax in several cases. Some examples are the friend class, the type long int or the unsigned integer 64 bits and also loop iterator definition and case sensitive include files. The effect on the code was not always straightforward and, in some cases, this required the implementation of specific macros able to embed the differences.

Most of the code in the simulator is common between Windows and Linux. This was to be expected since most interactions made by model code are with the SIMSAT infrastructure and not with the underlying Operating System. Unfortunately, not all infrastructure services are identical on the two platforms (as a result of the infrastructure porting), so this is one source of code differences between the Windows and Linux simulator models.

Some features not handled by the infrastructure originate other more significant changes. For example, a library called AQUA2 inherited from previous missions was used for importing the Spacecraft Database in XML format. This library was very Windows specific as it was using the Windows XML file reading functionality. On Linux the third party tool Xerces had to be used to read the XML files.

A categorization of all the required modifications has been done and, considering the large amount of source code present in the simulator, the low number of modifications required is impressive. This is the advantage of having masked the operating system behind a kernel of infrastructure functionality, such that porting efforts can be centralized and minimized at the higher level, almost independent from the specific missions.

Area	Development Infrastructure	Operating System	Operating System Infrastructure	SIMSAT	*Total*
Simulator	26	26	1	12	**65**
Telemetry Decoder	2	4	0	3	**9**
AQUA2	3	15	0	0	**18**
Total	**31**	**45**	**1**	**15**	**92**

Table 1. Distribution of Changes in the Source Code(*Measured in modules modified)*

2. Development Environment Changes

The biggest technical difference in the development environments used is the different organization of the source code. The Windows environment is organized in projects and the Linux is based on workspaces with makefiles.

A systematic approach was taken initially to move each individual project into a workspace and creating a relevant makefile to build it. The process required a number of iterations making sure each time that a change in the Linux

environment could either be applied to the Windows one, or a specific code had to be prepared. In all of the cases a retest in both environments was needed.

Once each makefile was developed and each project built, the same process was tested on both 32 bit and 64 bit Linux platforms. A few differences were found, due to the small differences of the two Linux operating systems, however solved using links against slightly different libraries where necessary. This highlights that once the porting to 32 bit Linux was complete, the porting to 64 bit Linux would be straightforward. At the end of the whole process a full set of makefiles were maintained in a way such that any time a new file was created in the Windows project, it was also added to the corresponding makefile for Linux.

On top of the above technical issue, there was another important step: the experience of developers had to be extended as people expert in coding in the Windows environment had the need of learning and, in some cases, finding new tools able to serve their needs in the new Linux environment. A particular difference is that the Linux development environment is much more command line based than the Windows one, mostly based on graphical interfaces.

3. ERC32 Emulator changes

At the starting of the AEOLUS Spacecraft Simulator development, for Windows there was commercially available the TSIM[4] emulator of the ERC32 processor. However, the same was available only for the Linux 32 bits platform. ESOC has developed an emulator for the same ERC32 processor that could already run either in Windows or Linux (32 and 64 bits) at that same point in time.

The Windows version of the simulator is a 32 bit application running on a 32 bit platform. When using the ESOC emulator on this platform real-time performance cannot be achieved, since the ESOC emulator was designed as a 64 bit application. The only way to achieve real-time performance on Windows is to use the TSIM emulator.

The Linux version of the AEOLUS Simulator can run on both 32 bit and 64 bit platforms. As in Windows, it is not possible to achieve real-time performance with the ESOC emulator on the 32 bit platform, but instead, it is possible to achieve it on the 64 bit Linux platform.

The decision was therefore to continue using the TSIM in Windows 32 bit platform and move to the ESOC Emulator for the Linux 64 bit platform. The overall existing emulator development and runtime configurations had to be modified to include the ESOC Emulator. Those configurations would have to be maintained for both platforms until the checkpoint was reached.

4. Scripting Changes

The SIMSAT infrastructure supports a script language to initialize and setup the simulator status as well as control it when a simulation is already running. The script engine allows the user to bring the simulator to the required status either by sending single user command on the commander window or executing complex functions including several commands, loops and verifications in an automatic way. The same scripts can also be used to restore or save the simulator status into breakpoint files.

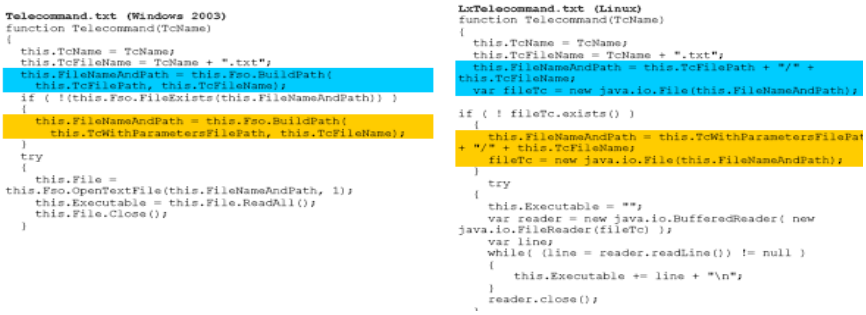


Figure 2. Changes in Script – File System Call

On the Windows version, all these scripts were written in JScript, Microsoft's own version of JavaScript. JScript is almost equivalent to JavaScript so it was always assumed that the scripts from the Windows 2003 version of the AEOLUS Spacecraft Simulator could be reused on the Linux version. Examples of differences were in the usage of GetObject function in JScript that does not exist in JavaScript or a different way to access array elements.

Some modifications done in the SIMSAT infrastructure porting led to differences in the script functions, and those changes indirectly affect the simulator, whenever those changed functions are used. In this case, wrapper classes and aliases have been defined to mask the differences in calling a function in SIMSAT Linux or Windows and they are all collected in a single configuration file for easy control.

Only in rare cases, the operating system change was reflected in script changes, like the access to the file system to get the list of files, used in the script to get the list of spacecraft commands defined and available in the system.

5. Data Displays Changes

[4] TSIM ERC32 Emulator, Gaisler Research, Första Långgatan 19, Göteborg, Sweden, www.gaisler.com

The SIMSAT infrastructure provides data displaying services in the form of Alphanumeric and Graphical Displays (ANDs). These are fully configurable by users to display relevant information like value, description and other characteristics for sets of parameters coming from the simulator model.

The SIMSAT Windows loaded each Data Display following a single ASCII configuration file in which the hierarchy pathname of each data item was specified in a predefined format (e.g. using specific delimiter like "!").

The new Linux infrastructure improved allowing each Data Display File to have one or more display files associated. These files are all written in XML and data items are specified using a new SIMSAT hierarchy pathname (e.g. with a "/" delimiter). These differences meant that the existing Windows data displays had to be ported to Linux, however keeping updates in line and always coherent in the two simulator versions.

The simulator is also equipped with a Conversion Tool able to auto-generate all Telemetry Data Displays from the Satellite Database. Due to the above changes, the Conversion Tool was also modified to generate data displays in both SIMSAT Windows and Linux formats.

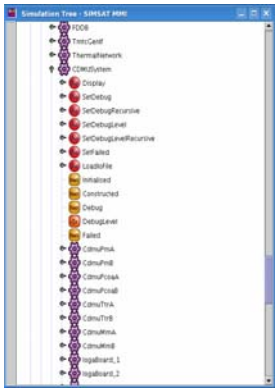


Figure 2. Simulator Object Tree

6. Configuration Changes and Control

Several infrastructure (SIMSAT) configuration files are required to setup the simulator architecture, i.e. the components to be loaded in the simulation. Components are not only the AEOLUS Simulator but also the Ground Models or the Telemetry Decoder models. Depending on the goal of the simulation, a particular component might be needed or not.

In addition, there are Configuration Files on SIMSAT Windows that involve specifying which script files to load at simulator initialization and those needed porting to equivalent SIMSAT Linux Architecture Files, where the structure was modified and XML format was used.

At least during the period where the two platform coexisted, the two sets of Architecture Files and Configuration Script Files had to coexist and always be coherently synchronized, which represented a tight configuration control necessity.

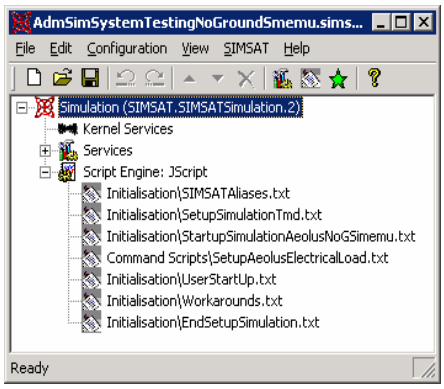


Figure 3. SIMSAT Architecture File

7. Operational Considerations

The way to operate simulators in ESOC is to have the Simulator Host Machine sitting in a central equipped computer room and access it remotely through the simulator interface from a number of service desktop computers located in different operations rooms. So far that the simulator and the remote machine were based on the Windows operating system, a direct remote connection was possible through the supported Windows remote terminal application.

The adoption of the Linux platform in parallel with the Windows one opened the need of remote connections from the same set of remote desktop computers to the Linux platform machines. An immediate solution able to cross the platforms is using X11 connections and a number of well-known applications like CYGWIN or VNC. This way of connecting to the simulator revealed a number of potential problems that did not exist when the same platform was used directly:

- Graphical performance degradation: a strong degradation of overall MMI performances when the user performed a simple graph display creation (even without data) on the MMI.
- Incorrect colour mapping: colours were sometimes not mapped correctly on the remote displays. This is relevant as each colour is mapped to a message criticality in the simulator log;
- Log display misbehaviour: in some cases, the simulator log display stopped displaying messages when changing filtering options. The problem did not occur when working directly on the simulator host;
- Firewall compliance: some remote terminal applications require more than one session open through the firewall. Any of those sessions is subject to firewall timeout breaking the connection. In other cases (see CORBA) ports are allocated dynamically and this is not acceptable as it cannot be controlled by the firewall.

	graph performance degradation	*incorrect colour mapping*	*log display misbehaviour*	*firewall compliance*
CYGWIN	Present	Not present	Not present	Not compliant
EXCEED	Present	Present	Present	Not compliant
VNC	Not present	Not present	Not present	Compliant
LOCAL MMI	Not present	Not present	Not present	Not compliant
UNIXEXPORT	Present	Not present	Not present	Compliant

Table 2. Remote Display Mechanism Comparison

As the result of a punctual analysis on some available software VNC was pointed as the solution. This also demonstrated to be the best solution in terms of local and network performances as using an optimized transfer of data. The cross of firewall is possible through the setup of a single port.

B. Decision Point

As agreed, the development started preparing a first delivery running on both Operating Systems. At that time a clear decision had to be taken mainly based on the evaluation of the system performances, but covering a number of areas presented below in more details.

The constant synchronization of the two systems was no longer acceptable. Parallel development on both platforms requires a large amount of effort so this would increase enormously if both platforms were required to be coherent and up to date through the rest of the project, particularly as the simulator grew in complexity. The decision still could not be a definite one but, clearly, a rollback later on would imply a high cost. To decide which Operating System platform to select, the two were compared in the aspects of functionality, system tests; and performance.

The comparison activity was done by the developer first and results were presented together with the first delivery of the simulator. The user community then prepared an independent comparison test in the same areas. The reports from both parties were collected and a final coordinated decision taken. The output of the reports confirmed the ability of the Linux platform not only to satisfy the need of the user community but also to be in line with the current ESOC software evolution strategy.

1. Functionality

There are no important differences in functionality between the Windows 2003 and Linux versions of the AEOLUS specific simulator code. Any AEOLUS specific object, attribute or service available on one platform is also available on the other with exactly the same features (name, type, interface, etc).

The only differences in functionality are resulting from the different SIMSAT infrastructure, however those either related to a modified way of achieving the same result or functionality added/ removed. The developer's report demonstrated that depending on the case the best platform to select was not always the same.

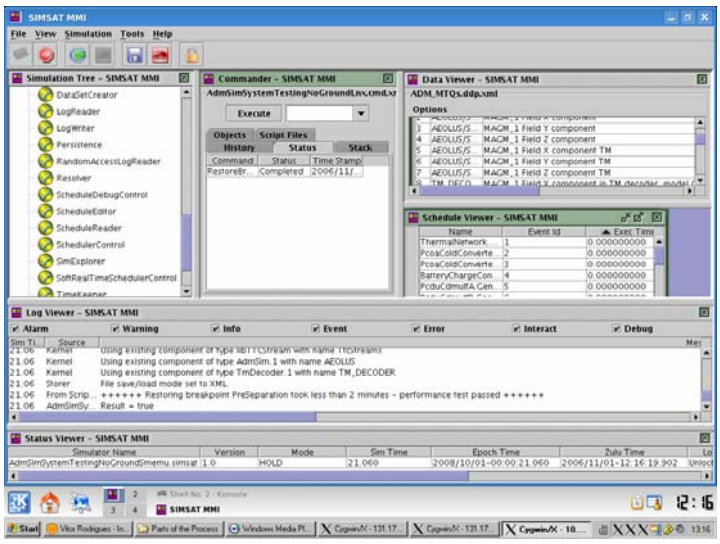


Figure 5. Linux MMI

On the other hand, the user community did its analysis, taking into account the preliminary output from developers. The differences were correlated with the operational usage needs of the simulator in order to apply a weight factor to the issues noted and being able to correctly decide which platform was globally responding better.

At the end of the process, it was clear that both simulators were responding to the actual need of the user community. The development of the new SIMSAT infrastructure did not change much the MMI appearance and the user interface so that any users able to operate the previous system (SIMSAT Windows version), would not require more than a familiarization session in order to operate the new system (SIMSAT Linux version). For those reasons, it remained clear that the new infrastructure was the solution to be adopted, as this kept the same functionality and had the advantage of still being continuously evolved and maintained.

2. System Tests

Essentially the functionality of the AEOLUS Spacecraft Simulator on the two platforms was the same. However, there were implementation differences that have led to differences in the system test results on the two platforms. In addition, the underlying SIMSAT infrastructure had a completely different history, being very mature on Windows and the Linux version never used before operationally.

The result of this comparison also demonstrated that the Linux platform was presenting some anomalies and memory leaks, expectable, however, in such a young software and overall not gravely affecting the operation of the simulator. All these problems were later fixed, in fact, within the warranty period of the infrastructure development contract, as expected.

3. Performance

As the absolute comparison was not possible, it was agreed that a comparison of the best performance reachable on each platform and version was still achieving the goal of finding out which system would accommodate better the needs of the mission. For this reason, it was agreed to compare the best performing Windows baseline (i.e. using the TSIM emulator) with the best performing Linux baseline (i.e. using the ESOC ERC32 emulator on 64 bits). Both were tested on the best hardware available at the time.

It was decided to compare the performance of the two platforms using the different configurations for what concerning hardware and software infrastructure, but the same on board software and spacecraft database were used.

A rigorous test plan was prepared and the script engine of the simulator was used to allow an automatic test easily repeatable. These same scenarios were also later reused at each delivery to constantly monitor and detect any change in the performance over the development time.

The result of the tests between the different platforms revealed very similar performances. A few anomalies were found on the Linux platform, which were already mentioned: memory leaks and bad performances in the filtering of the Log display. Although those problems were considered important, they could not be considered as discrimination point as they could be attributed to the still very young software and a solution was expected in later software deliveries.

Therefore, and as was also pointed out before, the disadvantages were foreseen to be solved and the great advantage of continuous evolution and maintenance for the Linux platform made this the choice.

IV. CONCLUSION

The greatest benefit of such an approach was, in reality, the possibility of postponing the decision to move to the new Linux platform at a later time when a feasibility test would be available on the real system. The good value of the strategy was the presence of a fallback solution to run the same Simulator Models also on the Windows platform infrastructure.

The AEOLUS simulator was the first one in ESOC that moved to the new SIMSAT infrastructure running on the Linux platform. The risk taken by the mission in using the first version of this infrastructure software instead of the consolidated version on the Windows platform has been well mitigated by having the first delivery on both platforms.

The additional cost due to the need of identifying first differences between the two platforms and later developing the models isolated from the target operating system is well balanced by the benefits of having the Spacecraft Simulator based on the new and commonly maintained infrastructure.

Even when the decision was taken to go ahead only with the new infrastructure, any disruption to the final users other than a fresher interface (but still similar to the Windows one) was occurring.

In a wider Esoc perspective, the previous version of the Mission Data Systems (either simulator or mission control system) infrastructure running on several different hardware and Operating Systems configurations was reaching a point where accommodation of modern and more demanding requirements would have been almost impossible or very expensive to achieve. The approach taken by ESOC infrastructure to move first the same functionality to the new platform (SIMSAT in the case of Spacecraft Simulators) and, once proven the concept, to enlarge the scope of this to the mission Simulators has been proven successful in the AEOLUS case.

The example of AEOLUS is today a reference not only for missions starting the development, for which using the SIMSAT Linux infrastructure is almost natural, but also for missions that are still today based on SIMSAT Windows based and intend to move their current simulator to the new platform.

In fact, at the time of the writing of this paper, several other missions at ESOC are receiving the first Linux deliveries of their operational simulators and are building on the experience of AEOLUS to be able to do the transition in a smoother and faster way. The additional efforts required to the AEOLUS mission and developer company are paying off today with the gained experience in the new environment and with bringing them in line with wider strategy of ESA.

In conclusion, it is true that the approach increased the overall costs, but certainly provided the high benefit of being able to follow a wider strategy for future ESA missions. Finally, the AEOLUS development strategy can be referenced as a success case for any future mission having to decide between new and very promising technology and systems or consolidated but predictably obsolete technology.

REFERENCES

[1] D. Rothwell, AEOLUS simulator Linux Port Technical Note, AE-TN-ESC-FS-4011, *Science Systems*, Issue 1, September 2006.

[2] D. Innorta, D. Guerrucci, P. Bargellini, AEOLUS Simulator: WINDOWS vs LINUX Assessment Test, AE-SIM-TN-1001-OPS-ONF, *ESOC/ESA*, Issue 1.1, December 2006.

[3] F. Metelo, D.Guerrucci, P. Bargellini, AEOLUS Simulator: Performance Assessment Test Plan & Report, AE-RP-ESC-FS-4015, *ESOC/ESA*, Issue 1.2, November 2007.