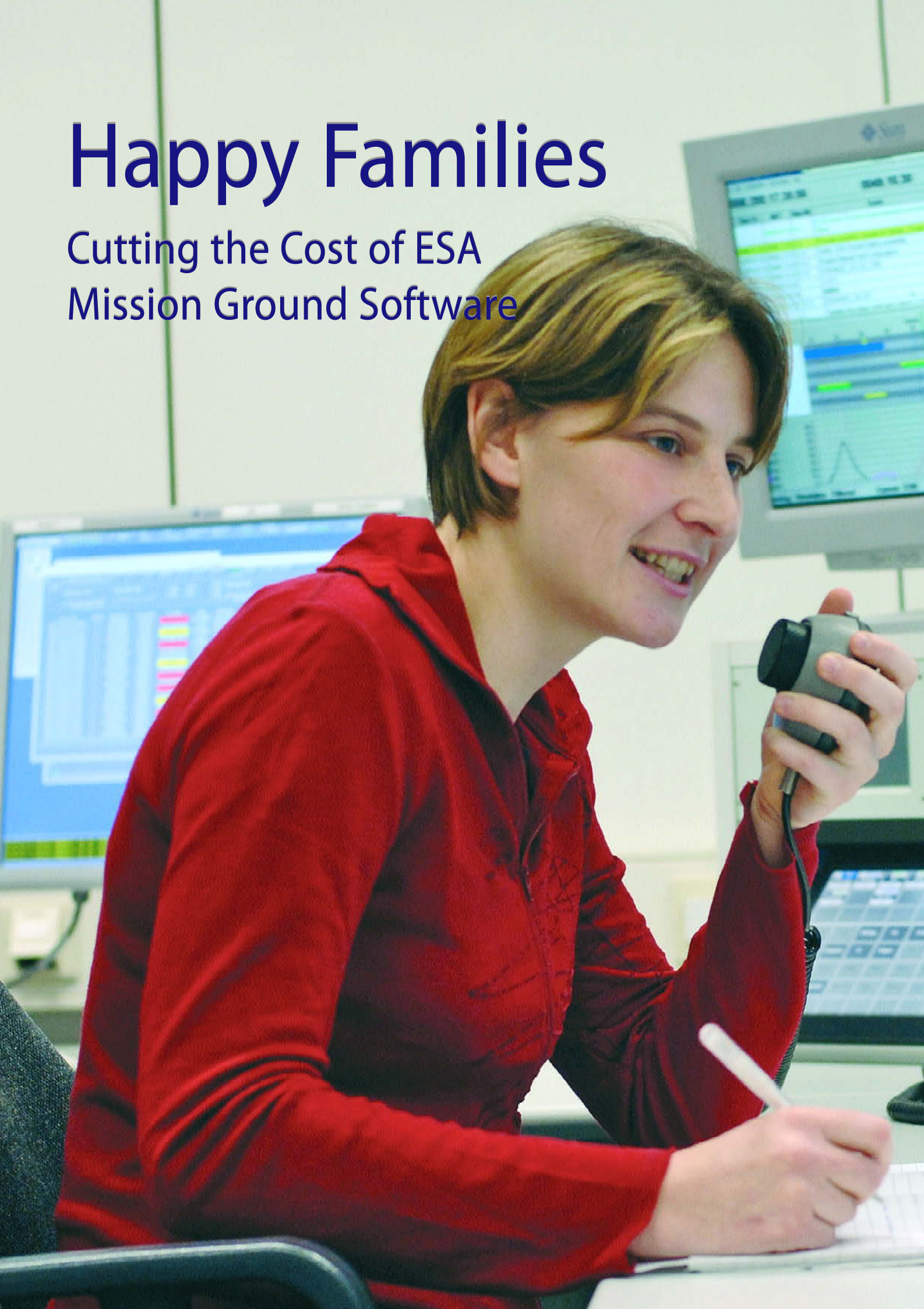
Happy Families

Cutting the Cost of ESA Mission Ground Software



Mario Merri, Alessandro Ercolani, Damiano Guerrucci, Vemund Reggestad & David Verrier
Ground System Engineering Department, Directorate of Operations and Infrastructure, ESOC, Darmstadt, Germany

Pier Paolo Emanuelli & Paolo Ferri
Mission Operations Department, Directorate of Operations and Infrastructure, ESOC, Darmstadt, Germany

In recent years, ESA has adopted a new approach to reduce cost and risk in the development and operation of ground software. The 'mission family' concept is the basis for cost-effective mission control systems for monitoring and controlling spacecraft, and operational simulators for testing and training. This concept is complemented by exploiting reusable software using a 'delta' approach. Since families of missions have lifetimes much longer than the individual projects, the challenges of evolving ground software and hardware platforms over ten or more years must be met.

Introduction

Operating spacecraft is a demanding job, requiring a complex ground segment of stations, communications networks and computer centres hosting large software systems. It is an unforgiving undertaking – a satellite is launched only once and there may be no second chance if there are ground segment failures or operational errors that degrade the mission or even result in its loss.

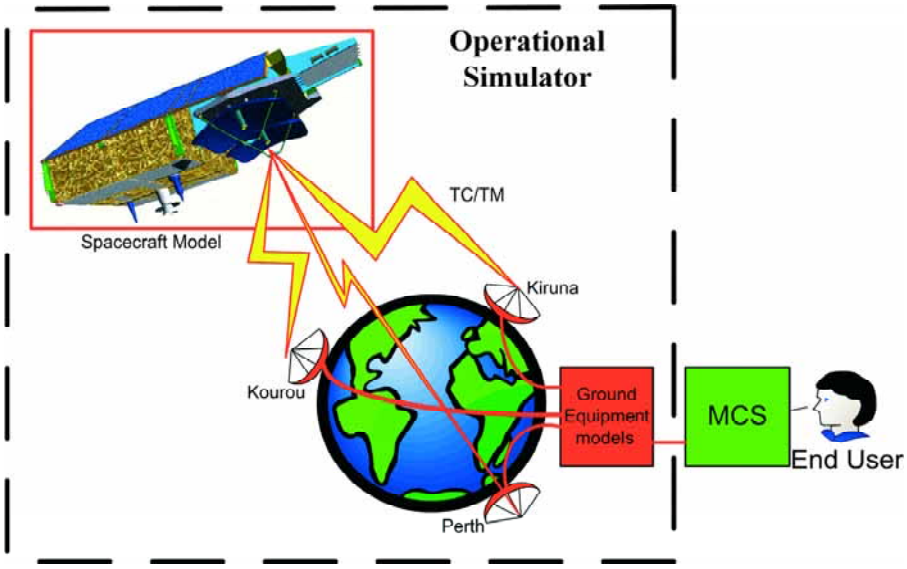
For more than 35 years, the European Space Operations Centre (ESOC) in Darmstadt (D) has accumulated extensive experience in the development, operations and maintenance of the ground software systems. Demanding levels of availability, reliability and maintainability imply high-performance technology and extensive validation, which drives up costs. Despite this, ESOC has succeeded in drastically reducing the costs of ground software systems for space missions. This has been achieved largely through reusing software that was designed and developed for reuse from the outset.

The first such reusable software system (now called 'infrastructure software') was the Multi-Satellite Support System (MSSS), developed in the 1970s and used for over 20 years. In the mid-1980s the first generation of the Spacecraft Control and Operations System (SCOS) was developed; it was succeeded by SCOS-2000 in the late 1990s.

In parallel in the simulation domain, infrastructure software included the General Purpose Software Simulator Package and then the first generation of SIMSAT. Both were based on the latest technology at the time. SIMSAT was then migrated to PC/Windows and converted to the more modern C++ language. More recently, it was made available under Linux, now the operating system of choice for simulators and MCS.

This infrastructure has thus continuously adapted to current technology, culminating in the ESA Ground Operations Software (EGOS), of which SIMSAT and SCOS-2000 are cornerstones. EGOS is an extensive suite of applications, middleware and lower level components covering all major software needs in the ground segment, including mission control software, software simulator infrastructure and ground station 'back-end' software. Naturally, this software has in effect encoded into it much of ESOC's accumulated operations expertise and knowledge.

In recent years, software reuse has been taken a step further, based on the observation that for related missions – 'mission families' – the common characteristics between the family members can be exploited for further cost reductions.

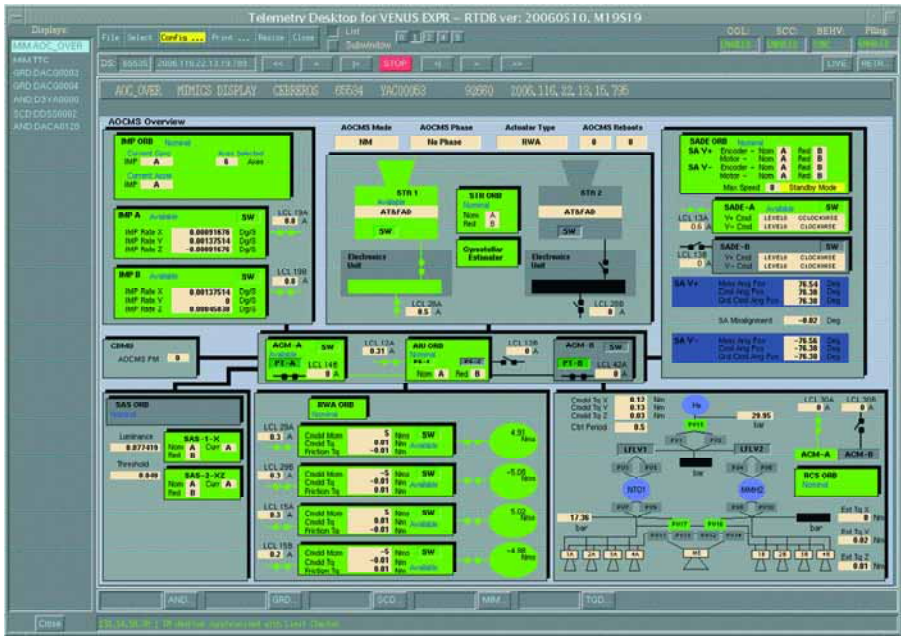


The Mission Control System (MCS) and Operational Simulator. TC: telecommand. TM: telemetry

The Mission Control System (MCS)

The MCS is a software system that enables the operators on the ground to interact with the satellite. It receives, interprets, analyses and archives telemetry, the data downlinked from the spacecraft used for monitoring its health and receiving its mission products. The MCS also generates, verifies and uplinks commands, transmitting instructions to the satellite to control all its operations.

Via the MCS, the Flight Control Team can assess the health of a satellite throughout its mission and command it to achieve the mission goals. The MCS is



A typical Mission Control System display

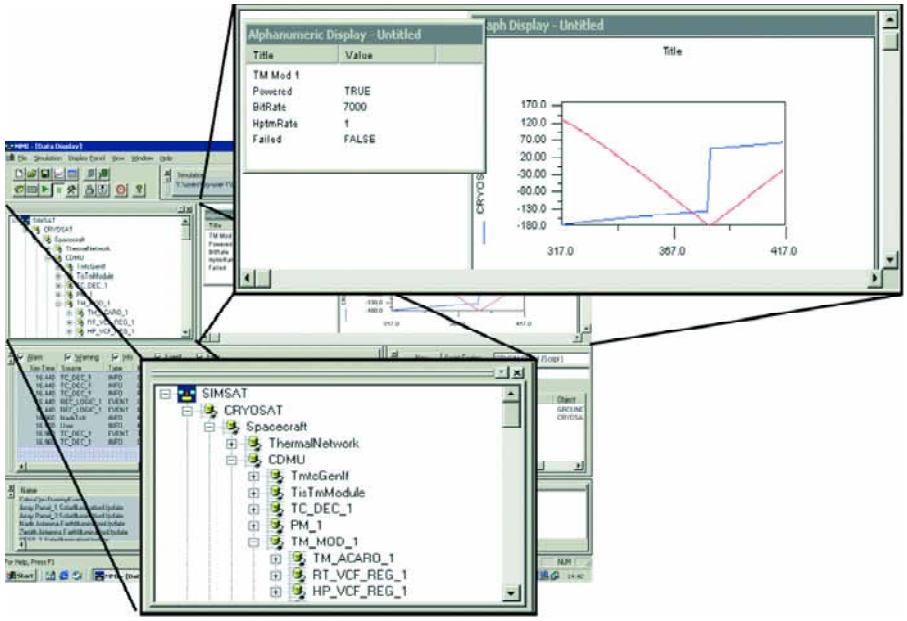
a critical system that needs to go through a comprehensive verification and validation process. This is not limited to the software itself, but extends to the operational data that are used to configure the MCS. It includes, for example, the database describing the spacecraft's telemetry and command data.

In addition to these functions, the MCS usually provides support for mission planning, data analysis and distribution, telemetry and command database management and onboard software management.

The Operational Simulator

The Operational Simulator is a software system that simulates the satellite, the ground stations and, to a certain extent, the space environment. An Operational Simulator is developed for almost every mission operated by ESOC. It is used to:

- support testing of the ground systems, including the MCS;
- support validation of operational data, including the operational database and the operational procedures;
- prepare the System Validation Test campaign, which contributes to the operational validation of the operations system that ground systems, procedures and personnel are able to operate the system successfully. This includes test and validation of the interfaces between ESOC and the satellite, confirmation of the correct functioning of the MCS and Flight Dynamics System and the validity of the operational procedures.
- support training of the Flight Control Team to make sure that everyone has the skills to ensure mission success under nominal and contingency situations. When something unexpected happens on the satellite or on the ground, it is extremely important to have a well-trained team of experts covering all areas and able to take timely decisions. Simulated failures in the space and ground segments test the team's responses to contingencies.



A typical Operational Simulator desktop, with graphs, displays and models

The ESOC Infrastructure Software

Most of ESOC's activities involve science and Earth observation missions. Such satellites are complex and every mission is different. At first sight, this would seem to prevent the reuse of tools and functions between missions, but software reuse has been achieved using various techniques, such as:

- changing configuration via setting different parameters in tables or databases;
- using appropriate software engineering technologies.

Two parts of EGOS are particularly relevant here:

- SCOS-2000 is the basis for mission-specific MCSs and covers most of the functions required for telemetry reception and processing, telecommand uplink and verification, data archiving, display and retrieval, and data distribution;
- the Simulus toolset is the basis for mission-specific Operational Simulators. It covers the SIMSAT simulator kernel and the Generic Models, including onboard processor emulators, orbit prediction and propagation (including environment perturbations), selected onboard subsystem models (such as thermal, electrical, telemetry/commanding) and ground system models.

Mission Families

A family is a set of satellite missions with a high degree of commonality in the spacecraft platform and/or in the operational profile. The families currently in use at ESOC are:

- interplanetary, in orbits not bound to Earth;
- Earth observation, in near-Earth orbits;
- observatory, with long visibility periods, high data rates and observation schedules proposed by users;
- navigation, typically a constellation of satellites in medium Earth orbits operated simultaneously.

There are requirements common to all mission families and these are generally supported by the infrastructure software. On the other hand, there are also requirements specific to one mission family only, and for which the

supporting software has to be specifically developed. Further software reuse can be achieved by exploiting this 'intra-family' commonality.

Two mission families are considered further here: interplanetary and Earth observation, although significant levels of reuse have also occurred in others.

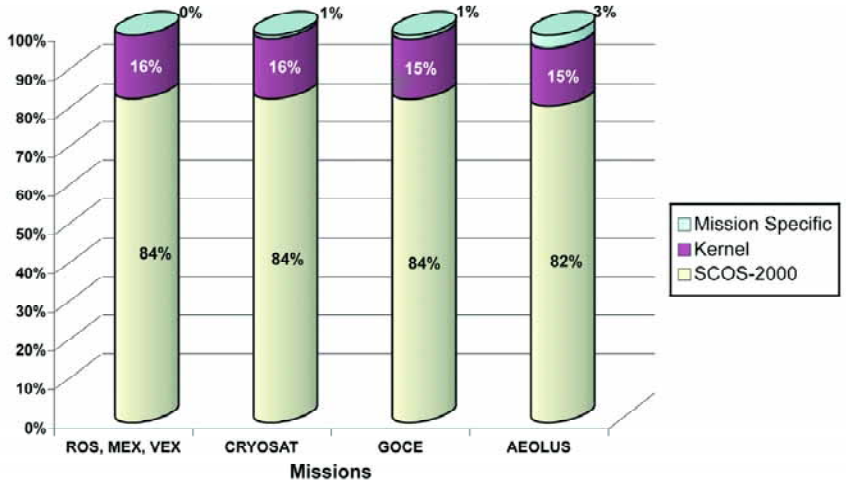
The interplanetary mission family

The interplanetary mission family comprises Rosetta (launched in March 2004 towards Comet Churyumov-Gerasimenko), Mars Express (launched in June 2003 and now orbiting Mars), and Venus Express (launched in November 2005 and now orbiting Venus). Two more missions are under preparation to join this family: BepiColombo (launch in 2013 to Mercury) and Solar Orbiter (launch in 2015 towards the Sun).

Missions that are launched on Earth-escape trajectories have operational characteristics significantly different from Earth-orbiting missions. Many of these directly affect the ground software systems. For example, the long radio signal travel delays must be taken into account in telecommand verification and telemetry time-stamping. Also, special commanding protocols (such as File Transfer) are needed to deal with the delayed space-ground interactions. The MCS has to cope with two or more parallel data streams, one for real-time telemetry and one or more transporting the telemetry recorded onboard during the long non-coverage period.

The unifying feature of these ESA interplanetary missions is their common spacecraft platform, leading to:

– the same types of processors and major software functions;
– similar orbit and attitude control sensors and actuators;
– similar interfaces to the Solid-State Mass Memories;
– reuse of payloads;
– similar satellite autonomy functions;
– identical operator interaction with the MCS.



The apportioned requirements for Mission Control Systems. ROS, MEX and VEX represent Rosetta, Mars Express and Venus Express, respectively

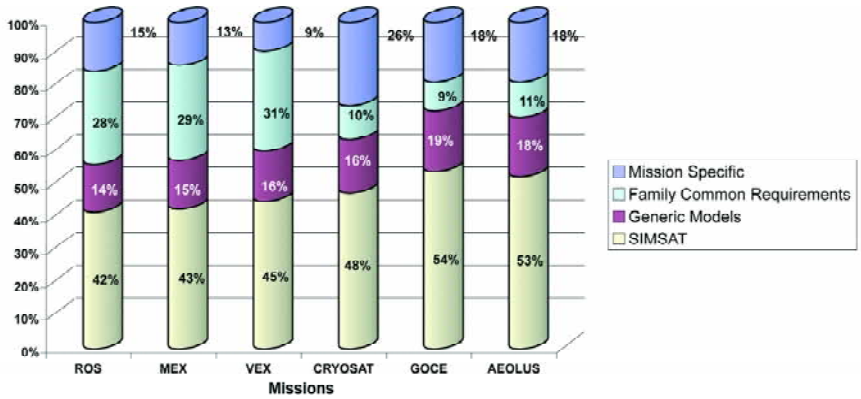
The Earth observation mission family

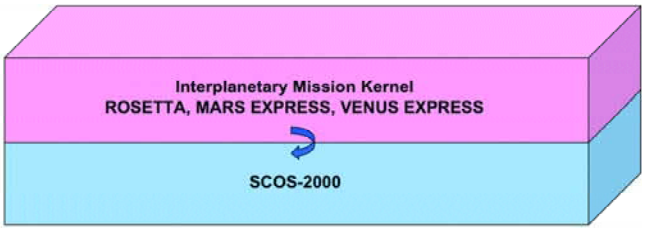
The Earth observation mission family consists today of CryoSat, GOCE and Aeolus. CryoSat was the first Earth Explorer Opportunity mission, which unfortunately suffered a launch failure in October 2005. GOCE is the first Earth Explorer Core Mission, with launch foreseen at the end of 2007. Aeolus is the second Core Mission, planned for launch in 2008. Future missions include CryoSat-2 (rebuild of CryoSat), Swarm, EarthCARE and the Global Monitoring for Environment and Security (GMES) initiative, comprising the Sentinel mission series.

Typically, these missions fly at low altitudes over the Earth (250–800 km), often in polar orbits, and their operational profiles provide frequent (about 15 times daily) and short passes (about 10 minutes) over a single ground station.

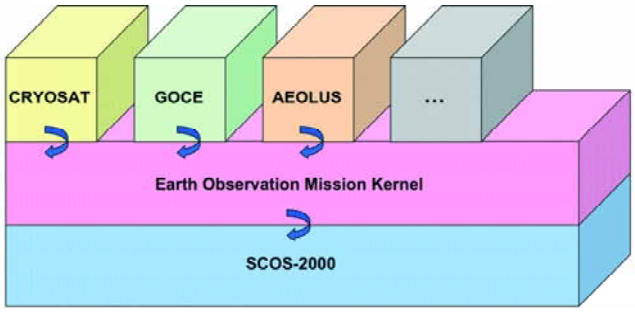
Despite the fact that the missions in this family have different spacecraft platforms and technologies, they all have very similar mission profiles and consequently very similar operational concepts. However, differences in satellite design can result in significantly different functions in the ground software. For example, onboard time is managed differently for the missions in the family. CryoSat uses the French payload DORIS to synchronise its time via a number of ground-generated microwave beacons, GOCE uses a simple onboard counter kept in synchronisation by the ground, and

The apportioned requirements for Operational Simulators

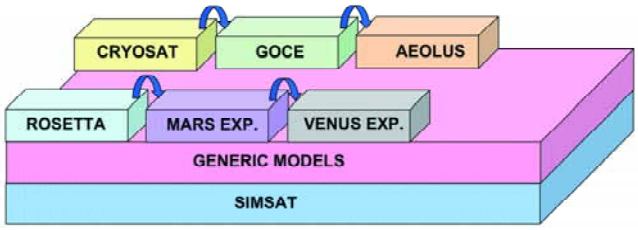




Typical software layering for reuse for Mission Control Systems in the interplanetary mission family. Arrow indicates possible path for requirements migration



Typical software layering for reuse for Mission Control Systems in the Earth observation mission family. Arrows indicate possible path for requirements migration



Typical software layering for reuse for Operational Simulators in the interplanetary and the Earth observation mission families. Arrows indicate possible path for requirements migration

Aeolus carries a Global Positioning System (GPS) receiver in addition to its counter.

The Delta Approach

The 'delta' approach isolates layers of software components that could be used as building blocks for mission-specific ground software (see illustrations). Three layers are identified:

- the lowest is the most generic and is reused across mission families. Typically, this is the infrastructure software;
- the middle increases the level of specialisation and is reused within a mission family;
- the upper is mission-specific and groups only those characteristics that are specific to a single mission.

Not all of these layers are needed every time, but a fundamental characteristic of this approach is that each layer is defined only in terms of the differences ('deltas') with respect to the layer below.

Requirements engineering

During requirements engineering, a 'delta requirements document' is produced that covers exclusively the mission-specific requirements. Since the majority of the requirements come from the lower layers, the document is slimmer and simpler. Experts can focus on what is unique in the mission, taking standard functionality for granted.

Clearly, a prerequisite for realising the full benefit of the delta approach is that the authors of the requirements document have a good knowledge of the functions of the reused software.

An important benefit in isolating the kernel requirements from the lower and upper layers is that it makes it easier to follow the evolution of the infrastructure software.

In fact, requirements common to missions in the same family will move down from the mission-specific context to the family kernel layer, while requirements generic enough to serve any type of mission can be moved down to the infrastructure layer. There is a similar process for Operational Simulators. The process of assigning or moving requirements between software layers is 'configuration-controlled': a control board ensures that the requirements are generic and stable.

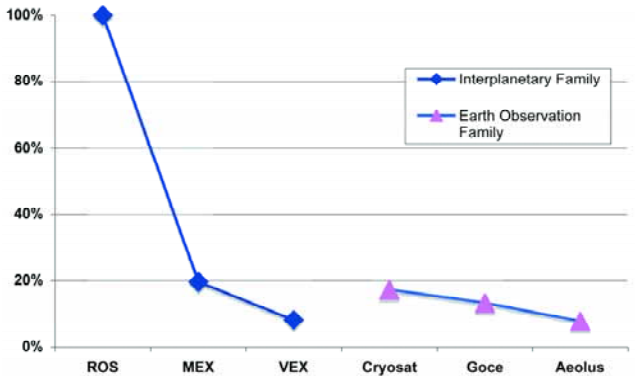
The rest of the lifecycle

The same basic idea applies to the rest of the lifecycle: concentrate on the differences and reuse as much as possible of what is already available.

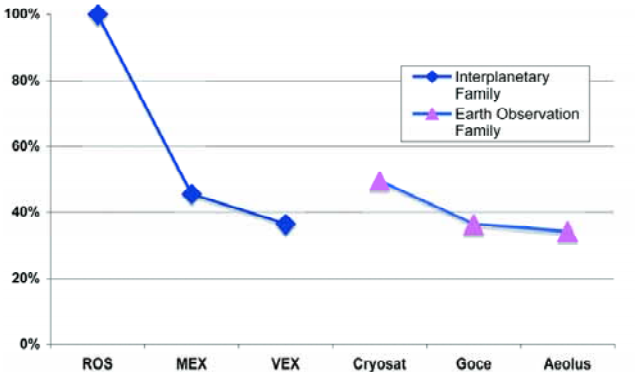
Specific to architectural design is the reuse of interface definitions along with the software at either side of these interfaces. For example, the interfaces between the MCS and Operational Simulator are based on generically defined file formats. For the Operational Simulator, ESA has defined a standard for the interaction between models and the simulation infrastructure. This Simulator Model Portability (SMP) standard encourages platform and infrastructure independence and permit the reuse of models both throughout the mission lifecycle and also from one mission to another.

During the implementation process, the software development team makes use of the fact that the infrastructure software and the mission family kernel are self-standing components to ensure maximum reuse. This also helps open industrial competition and extends the pool of contractors familiar with the software. This wide reuse creates a virtuous circle in which the software progressively improves as it is employed in more and more missions. In fact, fixes to software bugs found in one mission-specific system may be fixed in the other systems in the family at minimal effort.

In the maintenance phase, combined maintenance reduces costs. 'Combined maintenance' means that a single team maintains several data systems, such as



The development costs of the Mission Control Systems for the two mission families relative to the most expensive development



The development costs of the Operational Simulators for the two mission families relative to the most expensive development

all the mission control systems in a family. Overheads and maintenance manpower per mission are reduced: costs are shared among the missions and the engineers are given more interesting work through involvement in several missions. The delta approach also ensures that the amount of mission-specific software needing to be maintained is reduced.

The delta approach in practice

This approach achieves the best results when it is followed from the outset of a development. The later the start, the less effective it is.

For the MCS of interplanetary missions, a two-layer delta approach was used for all missions in the family, thus capitalising on the common spacecraft platform. Practically, this implied a single system that could be used for each of the three missions by simple reconfiguration. The effort usually required for one mission covered all three.

For the MCS of the Earth observation family, a three-layer approach was adopted to cope better with the spacecraft platform differences. Common functions in the family are separated in the middle layer. This is the Earth Observation Mission Kernel, lying between the mission-specific software (unique to each mission) and the infrastructure software. It is a self-standing piece of software.

For Operational Simulators, the tailoring is very similar for the two families. Heavy reuse is made of the infrastructure software (SIMSAT and Generic Models) and commonalities among spacecraft are exploited within each family by reuse of models from previous developments in the family. For interplanetary missions, near-identical spacecraft platforms means that the simulators are also almost identical. The teething problems of developing the simulator, such as immature or late onboard software and problems with early versions of spacecraft documentation, were mainly borne by the first mission in the family, Rosetta.

For simulators there are other aspects of reuse. Emulation of the onboard processors means that the actual onboard software can be reused, leading to iterative reuse from space to ground segments. The software Model Portability standard allows reuse of models (developed by the spacecraft manufacturer, for example) in the Operational Simulator. In fact, if common subsystems are used in different spacecraft, models developed in compliance with SMP can be used in any simulator, whether developed for spacecraft checkout by the spacecraft manufacturer or for operations training by the operations supplier.

These examples show how reuse is beginning to happen at the higher level of the space system lifecycle. A similar trend can be seen in mission control systems and operations preparation tools. For example, common tools for spacecraft checkout and operations preparation are promoting seamless transition from spacecraft manufacture to operations. This was exploited recently by Herschel and Planck.

The arrows in the illustrations show the typical direction of migration of functionality. Once stable, the more generic functions may migrate to the lower software layer (not shown for the Operational Simulator for simplicity).

Benefits for the infrastructure software

Evolution of the infrastructure software to cope with the needs for new generic features is a relatively slow process because it requires consensus across mission families. The mission family

kernel layer allows missions to build up initial operational experience on new features that are of general interest and could be introduced into the infrastructure software at a later stage. Thus the mission family kernel is like an operational laboratory where requirements are refined, experimented with and modified until they are mature and stable. At this point, the new generic features may be imported into the infrastructure software. A successful mission family kernel will thus be reduced to nothing as this importing process proceeds!

The infrastructure software must also evolve to take account of new versions of the operating system and commercial-off-the-shelf software, as well as changes resulting from fault correction during maintenance and, last but not least, major functional upgrades. Migration of ground software to new versions can be a major effort, involving extensive testing and validation by developers and the Flight Control Team. For missions in their operational phase, the effort and operational risks are often considered to be too high, so missions in their routine phase are usually reluctant to take new infrastructure versions. However, the mission family approach can help to remove this barrier. For example, when Venus Express was preparing for launch, its MCS was based on the most recent SCOS-2000 (at that time, version 3.1), whilst the already-flying Rosetta and Mars Express continued to use the previous version. Once the Venus Express project completed operational validation of the new system, migration of the Rosetta system to the new interplanetary kernel was straightforward, taking a few months. Following this, migrating the Mars Express MCS was even quicker, lasting only a few weeks.

Cost, schedule, quality and risk

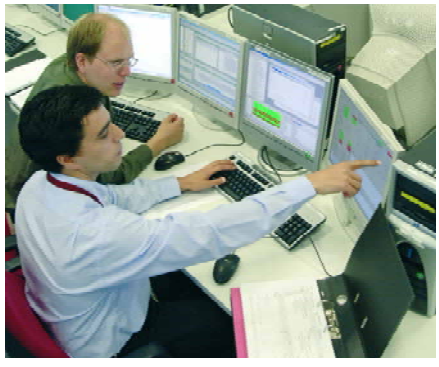
The delta approach achieves cost reduction by having:

- less system specification, design and development effort;
- less validation effort;
- shortened project elapsed time;
- higher software quality;
- lower risk because of the validation by earlier missions in the family.

The illustrations show that the development costs progressively decrease and that the earlier missions have to pay the price of an earlier (less rich and stable) infrastructure software and of being the first implementers of the mission family kernel. Clearly, one has also to factor into the interpretation of these figures the complexity of the spacecraft, which is certainly higher for the interplanetary family. Additional cost savings can be achieved during maintenance.

Conclusions

There are benefits in taking the delta approach to building mission control systems and simulators, and splitting the mission ground software into layers for progressive reuse. Software is systematically reused, focusing specifications, design and development on the differences. The new software may then itself be reused in a new layer, the Mission Family Kernel. This delta approach encourages new missions to make their new features or improvements generic. Migration to new versions of the infrastructure software can be done by one mission (say, a new family member), and the other missions in the family may then benefit from it with minimal risk and effort.



ESOC simulators have also benefited from this approach. With a common simulation framework, standardisation of external interfaces, internal interfaces and components, a high-level of reuse has been achieved. Simulator costs, however, remain sensitive to the differences between the spacecraft they simulate. By the same token, they are reduced as standardisation of space and ground segments subsystems, interfaces and processes consolidate.

ESOC has drawn its proven infrastructures for MCS and Operational Simulators together to provide the EGOS open, flexible infrastructure that stimulates reuse during all project phases and from one project to another. This provides a virtuous circle of improving quality and increasing reuse, which results in lower costs and improved service for the end users.

Last but not least, it is worth stressing that the whole field of ground segment engineering and operations benefits from the family mission approach. This allows combination or cooperation between teams for the individual families in all specialist areas, ranging from ground stations, mission data systems and flight dynamics to ground segment operations and spacecraft operations. On the operations side, similarities between missions and between the systems on the ground used to operate them save on training and operations costs, leading to fewer operations staff per mission, as well as facilitating mobility of staff between missions. In short, knowledge reuse within the various engineering teams and the infrastructure itself is maximised, with considerable benefits in cost and risk reduction.