XMM INSTRUMENT ON-BOARD SOFTWARE MAINTENANCE CONCEPT

Mr. N. Peccia
European Space Operations Centre (ESOC),
Robert-Bosch-Str. 5, 64293 Darmstadt, Germany

Mr. F. Giannini
European Space Technology Centre (ESTEC),
Keplerlaan 1, 2200 AG, Noordwijk, The Netherlands

ABSTRACT

Whilst the pre-launch responsibility for the production, validation and maintenance of instrument on-board software traditionally lies with the experimenter, the post-launch maintenance has been the subject of ad hoc arrangements with the responsibility shared to different extent between the experimenter, ESTEC and ESOC.

This paper summarizes the overall design and development of the instruments on-board software for the XMM satellite, and describes the concept adopted for the maintenance of such software post-launch.

The paper will also outline the on-board software maintenance and validation facilities and the expected advantages to be gained by the proposed strategy.

Conclusions with respect to adequacy of this approach will be presented as well as recommendations for future instrument on-board software developments.

Keywords: **On-Board Software, System testing, Software life cycle, Software maintenance**

1. INTRODUCTION

In the last decade, the complexity of space missions has increased significantly due to the more demanding requirements on mission efficiency and quality of mission products. Such requirements could only be satisfied by designing intelligence on board for increased autonomous operation of the spacecraft and the instruments in its orbit.

On-board software and autonomy however have a significant impact in the design of the ground facilities for the support of the mission. Although instrument on-board software is designed, developed and tested following strict quality assurance procedures, experience of past and current missions show that the capability of reprogramming instrument on-board software from the ground is an essential requirement throughout the instrument lifetime.

Certain events during the instrument lifetime can create the necessity to modify the flight software. The causes of change in the on-board software are manifold :

- Change of specification (e.g. errors, essentially numerical, in the specification of thresholds, calibration, delays, etc.)
- Non conformance of the software with the original specifications (e.g hidden

bugs not detected during testing on ground)

- On board hardware failure following which the instrument can only be recovered by reprogramming the on-board software. This event is the most likely reason for the necessity of maintenance activity because its unexpected nature renders it impractical to implement complete autonomy in the software with respect to such failures.

- Change in strategy in instrument operation (e.g. changes to improve capability or efficiency)

The complexity of instrument on-board software maintenance is directly related to the on-board software configuration.

Different approaches have been taken to instrument on-board software maintenance from mission to mission. The main variation has been the responsibility for actually making software corrections and implementing new on-board software requirements, which has in some cases been done by the experimenters and in other by ESTEC and/or ESOC.

The factors influencing the choice of a particular maintenance scheme are :

- availability of expertise

- availability of :

  a Software Development Environment (SDE) which contains CASE tools supporting on-board software lifecycle for development and change of the software, verification and validation, configuration management and documentation. The facilities also include cross-compilers, cross-debuggers and downloaders to compile, load and debug the instrument software from the host to the hardware target.

  a Software Validation Facility (SVF) for on ground testing and validation. The SVF provides facilities to emulate or simulate the hardware environment of the instrument on-board software. Facilities range from software simulators and emulators to replicas of instrument on-board hardware systems.

- duration of mission

2. XMM OVERVIEW

The X-Ray Multi-Mirror Mission (XMM) is a high throughput X-ray spectroscopy mission (photon energy range from 0.1 Kev to 10 Kev), which is the second cornerstone of the ESA long term scientific plan. The XMM is a facility type observatory open to the worldwide astronomical community. The scientific payload forms an integrated mutually complementary package optimised to fulfil the scientific aims of the mission and fully exploit the ESA supplied X-ray optics.

The XMM observatory will offer a major step forward in the field of X-ray astrophysics in the 21st Century. It is envisaged as a long duration facility class mission aimed at performing detailed imaging spectrophotometry of a wide variety of X-ray sources. The observatory will be placed in a 24 hour highly eccentric inclined orbit to allow uninterrupted observations up to 16 hours using the groundstation of Perth (Australia). The spacecraft consists of a service module which carries the payload module.

The scientific instruments are:

- European Photon Imaging Camera (EPIC)

  The XMM x-ray telescope consists of three separate co-aligned mirror modules, for each of which EPIC will provide an imaging x-ray focal camera. Each of these cameras will be mounted at the focus of the respective mirror modules. Two different types of Charge Coupled Devices (CCDs) will be used: one type based on p-n and the other on MOS technology.

- Reflection Grating Spectrometer (RGS)

  RGS features two independent instrument chains placed behind two of the three mirror modules. Each chain incorporates an array of reflection gratings which pick off roughly half of the X-ray light and deflects it to a strip of CCD detectors offset from the telescope focal plane. The remaining light passes undeflected through the grating stack where it can be utilised by other instruments located in the focal plane.

- Optical Monitor (OM)

  OM is a UV / optical telescope with two chromatically split channels, the blue channel and the red one. Both beams are transmitted to the CCDs detectors.

The XMM spacecraft will be operated in a continuous interactive mode from a Mission Operations Control Centre (MOC). XMM Science operations will be conducted from a Science Operations Centre (SOC) in close interaction with the MOC.

Considering

- the long duration of the mission (10 years)
- each experimenter has his own SDE/SVF
- distributed processor architectures are present in the payload
- different languages are used on the processors

the following sections describe how a different set of instrument on-board software maintenance and validation facilities could be assembled, which would allow the SOC, given the appropriate expertise, to assume the responsibility for instrument on-board software maintenance in the majority of the cases.

Additionally the following items will be addressed

- Design choices
- Inclusion of hardware-in-the-loop
- "worst case" testability
- exception handling in the software

The current baseline is that the Instruments Software will be maintained in the SOC with the Instrument Software Subsystem (ISS); however during the early phases of the XMM mission support from the instrument development teams will be available (e.g. for validation).

3. XMM INSTRUMENT SOFTWARE MODULES

This section describes briefly the various on-board software modules in the Instruments,

mostly on different hardware units (instruments contain more than one processor with a maintainable software module) :

In the **EPIC** experiment SW is present in :

a) EPIC Mos Data Handling Unit
b) EPIC Control and Recognition Unit
c) EPIC Pn Data Handling Unit
d) EPIC Pn Event Analyzer Unit
e) EPIC Pn Analogue Electronic Unit

In the **RGS** experiment SW is present in the RGS Digital Electronic Unit running on the following processors:

f) Instrument Controller Processor
g) Data Pre-Processor

In the **OM** Software is present in:

h) Instrument Controller Unit
i) Data Processing Unit

In the following the Software Modules will be indicated by the above letters. These modules are of different size and complexity, and they can be classified in 2 categories:

- Running on what is traditionally identified as the Instrument Controller (a,c,f,h)

- Running on secondary processor (b,d,e,g,i)

Regardless of the names used for the Units, we will call IC the units interfacing with the Spacecraft On Board Data Handling System (OBDH).

The Software Modules run on different type of processor:

- MIL-STD-1750A (a,c,f,g,h)
- HARRIS 80C86 (b,d,e)
- MOTOROLA 56001 (i)

and different language are used:

- Ada (a,c,f,h)
- C (b,d,e,i)
- Assembler 1750A (g)

Assembler is also used on other software modules which are on PROM and are not modifiable (e.g. boot/loader code for f and h).

4. XMM INSTRUMENT DEVELOPMENT ENVIRONMENTS (SDE)

The XMM instruments are being developed by different experimenters across Europe (United Kingdom, France, Italy, Belgien, Germany and Netherlands) with some collaboration from USA (RGS and OM instruments). The consequence is the use of different host machines, target processors, languages and tools among the instruments.

Figure 1 summarizes the Software Development Environments which are being used to develop the various instrument on board software modules.

5. XMM INSTRUMENT ON-BOARD SOFTWARE MAINTENANCE APPROACH

The following assumption are made:

- Instrument simulators will be available and they will be based on Instrument Controller (ICU) processor emulator running the on-board SW.

SW Module	Host Machine	Target Processor	Language	Tools
a , c	SUN	MA31750	Ada	TLD APSE, Debugger, Simulator, TEK V1750 Emulator
b, d, e	HP9000	80C86	C	HP AXLS C/80C86 Cross-compiler, Debugger, Emulator, Simulator
f, h, g	SUN Solaris 1.1	MA31750	Ada Assembler	Tartan Compiler / assembler, AdaScope debugger
i	SUN	56001	C	GCC,G56KCC OM Simulator

Figure 1; Instrument Software Development Environments

- All software modules need to be maintained

- Modification of the instrument on-board software cannot damage the instruments while the instrument is monitored from the ground.

- The Software delivered with the instrument flight model (FM) has been fully validated.

The purpose is to outline a coherent approach in the frame of a plan for the maintenance of the software on the various on-board processors and during the various relevant phases (development, commissioning and routine operations).

The trade-off between instrument on-board software maintenance at the XMM SOC on the one hand and maintenance via each experimenter on the other hand has been based on the following assessment criteria :

- Criticality of the on-board software, which covers an assessment of the impact an erroneous software modification might have on the performance of the instrument

- Software complexity versus availability of expertise, which addresses the degree of expertise needed for a specific software maintenance during the commissioning and the routine operations phases.

- Cost aspects which addresses

  - investment costs for hardware, software and documentation including installation at the SOC and training of personnel
  - operations costs at the SOC

6. DEVELOPMENT ENVIRONMENT

The set of activities involved in the maintenance of the XMM instrument flight software will be executed at the XMM Scientific Operations Centre (SOC).

In order to perform these activities the SOC will require a common SDE which will ease the maintenance activities and will limit the costs. The development environment for the Instrument Software will be composed of the total set of Software tools used by the developers of the Software modules.

All tools mentioned in the Figure 1 on section 4 will be available for modification of the instruments Software to ensure compatibility with the implemented instrument flight software.

Other tools will be used in the development of the Software (e.g. AdaNice HOOD tool for the Architectural design of the EPIC Data Handling Software), but the use of such tools is not considered necessary for the maintenance, due to the limited structural changes in the code during the maintenance phase.

The development environment will be hosted on the smallest set of computer needed to host all tools in a version equivalent to what used by the developers. At the moment a SUN SPARC and an HP9000 are needed to host all tools. In order to ensure that the compiled code produced by the SOC SDE is compatible with that flown during the mission it will be necessary to freeze the compilers version at the version delivered with the instrument Flight Model.

A configuration management tool should be added in order to keep track of changes. No configuration information prior to delivery will be used. Configuration management will be restarted with the Software as delivered for the launch.

Full documentation of the Software development will be available on paper as delivered by the developers. Electronic form of the documents might also be available, but no standard format has been mandated.

The following will be available:

- Source code of all instrument Software
- All "makefile" and any image generation procedure used by the developers

7. VALIDATION ENVIRONMENT

The validation environment will be different for the various type of processors used and the functionality of the Software module. The main driver of the proposed approach is the high investment and maintenance costs associated with a SVF based on an Engineering or spare Flight model.

7.1 INSTRUMENT CONTROLLER

For ICU software (modules a,c,f,h), the capability of the instrument simulators to run the Software will be exploited. This solution does not have the fidelity of the actual hardware, and therefore its adoption is associated with an element of risk. The level of risk is related to the degree to which the instrument on-board software is sensitive to the flight hardware performance (timing, I/O performance).

Other solution would be the "hardware-in-the-loop" design, based on commercially available VME cards and hosting the target processors. This approach was discarded due to higher costs because additional secondary processor hardware is needed.

The use of the instrument simulator as a validation tool has the advantage that it implicitly contains a realistic environment simulation and the means to easily vary this environment. The preparation for and conduct of validation tests is easier than for an EM (Engineering Model) or a spare FM (Flight Model) based system.

After the unit testing and software integration, the new executable image of the module will be first executed on the 1750 processor emulator for simple tests.

It will be then loaded on the instrument simulator, exercised by TC and stimulated by data files reproducing the Instrument data

flow. The data files used in these tests are available from unit tests or calibration tests.

This will allow to "partially" validate the Software before up-linking into the instrument. The settings of the instrument simulator will allow to simulate Hardware failure in the instrument. This strategy for test and integration causes some difficulties on real time embedded software, as follows :

- Emulators have limited facilities for exercising in real time and simultaneously monitor the embedded software
- It is often impossible to reproduce a test 100%
- It is very difficult to create a "worst case" test
- It is difficult to exercise exception handling in the software

The proposed approach is also based on the criticality of the Instrument Controller on-board software. Error in new images or software patches causes no damage to other instruments. A power on reset will bring the software back to the PROM reference. Full ground validation is not required because of criticality. The proposed partial validation is necessary due to the complexity of some software modules.

7.2 SECONDARY PROCESSORS

Modification to all other software modules will only be tested on the host computers and on the processor emulators (b,d,e,g,i) with data files generated by simulation software or collected during instrument testing. The impact of software modifications on the secondary processors is considered negligible. Additionally the XMM instrument simulator can not be used for validation because it does not emulate the secondary processors.

Anyhow, the relative simplicity of these Software modules makes them easier to test except for timing behaviour. Furthermore, the modification to these modules are more likely to be needed during the early phases of the operations, because of the possible difference of the actual data from the expected ones.

During the early phases of the operation (6 months), the instrument developer teams will modify the software using the Instrument Software Subsystem at the SOC; validation of the Software modules will be complemented by the possible use of the test equipment available at the experimenter premises.

After this period, if the experimenter facilities are not available, the Software will only be tested on the host computers; the last tests before declaring the Software operational, will be executed on the flying instrument using when possible the period of the orbit below 40,000 Km.

It is, however necessary, that it be demonstrated that changes to the instrument software do not adversely affect the performance of the system as a whole, either functionally or in the consumption of resources. This can be done by analysis (in the absence of a validation facility), or by demonstration.

The responsibility for demonstrating that any changes to instrument software will not adversely affect the system will lie with the XMM SOC during the routine operations phase.

8. CONCLUSIONS

In the XMM SOC project the instrument software maintenance problem is tackled by setting up a centralised software maintenance facility, the ISS (Instrument Software subsystem), which will take over the on-board software maintenance of all instruments when the commissioning phase is over.

In order to fulfil the requirements and responsibilities for such facility the XMM SOC requires :

- a common Software Development Environment (SDE) compatible with that used by the experimenters to produce the flight software and maintainable for the mission duration. This common SDE will ease the maintenance task and will limit the costs.
- delivery of the software module source code, the standards applied during development and testing, the test harnesses, test procedures and test results.
- a Software Validation Facility ,also maintainable throughout the mission, which consist of :
  - the instrument simulator running the instrument controller emulator
  - simple emulators of the secondary processors for testing and validation on the host computer

The final decision whether or not to implement an instrument software change resides with the XMM Project Scientist.

The proposed solution is based on the analyses of the criticality of the XMM instrument on-board Software as regards instrument performance, on the availability of expertise at the SOC during the various phases of the mission as well as on a cost estimation.

9. RECOMMENDATIONS

For missions with long lifetime, as XMM, ESA should take over post launch instrument on-board software maintenance. This will be more cost effective, since it involves only a marginal expansion of existing teams. It will also result in a better and more responsive service, will simplify the operational interfaces and will help continuity of expertise in the SOCs.

Standardisation of Software Development Environments should be managed / mandated early in the project in order to reduce the cost of the maintenance environment without penalising the instrument developers.

10. REFERENCE DOCUMENTS

1. Science Operations Centre Implementation Requirement Document (SIRD), PX-RS-0392
2. SOC Implementation Plan (SIP), XMM-SOC-PL-0100
3. Software Project Management Plan for EPIC Experiment, TL 10002 EPIC-LAB-PL-001 Issue 1, April 1994
4. XMM-OM Instrument Control Unit Software Development Environment, XMM-OM/MSSL/SP/0025.01 25/5/94
5. XMM-OM Instrument Control Unit Software Verification & Validation Plan, XMM-OM/MSSL/SP/0026.01 25/5/94
6. Digital Processing Unit Electronic Ground Support Equipment and Software Development Environment XMM-OM/PENN/SP/0005.1 24/5/94
7. DPU Software Validation Plan, XMM-OM/PEN/SP/0006.draft 3/5/94
8. Huygens On-Board Software Maintenance, DOPS-SMD-HUY-OBS-001, Issue 1, 17.02.94