Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

The XMM Survey Science Centre

Clive G. Page

Dept. of Physics & Astronomy, Leicester University, UK

Abstract. The XMM Survey Science Centre will process all data from the XMM X-ray observatory to produce standard data products for observers and catalogues of serendipitous sources, all of which will be included in the XMM science archive. This paper introduces the XMM mission and outlines the plans for software development.

1. The XMM Mission

The X-ray Multi-mirror Mission (XMM) is the second cornerstone in ESA's Horizon 2000 programme. The observatory is equipped with a set of sensitive high-resolution X-ray instruments with simultaneous optical/UV monitoring. XMM is due for launch in late 1999 with consumables designed to last 10 years.

The X-ray telescope consists of three modules each having 58 nested thin mirrors to provide a total collecting area of $4500\,\mathrm{cm}^2$ and a field of view $30'$ across. Each module consists of an X-ray CCD camera imaging the 0.2–10 keV band with a resolution of $\approx 10''$ (FWHM); two modules are also fitted with an X-ray reflection grating spectrometer providing concurrent spectroscopy over the 0.35–2.5 keV band.

The Optical Monitor (OM) is a 30-cm Ritchey-Chretien telescope equipped with an image-intensified CCD detector, which should be able to detect stars of 24^m in a 1000 s exposure. More detailed descriptions of the XMM instruments are given in Lumb et. al. (1996) and the XMM Home Page.[1]

2. The XMM Survey Science Centre

XMM, besides providing excellent X-ray, optical, and UV data on the chosen target, will also detect large numbers (50–200) of other X-ray sources in each pointing. These serendipitous sources will constitute a deep sky survey which, over the course of the mission, will become a major resource offering potential for a wide range of astrophysical studies.

ESA has established the XMM Survey Science Centre (SSC) to ensure that all the scientific data are processed uniformly and systematically, and that the wealth of information from the serendipitous detections is properly catalogued and followed up for the benefit of the whole community.

[1] http://astro.estec.esa.nl/XMM/xmm.html

© Copyright 1997 Astronomical Society of the Pacific. All rights reserved.

The SSC Consortium, led by Mike Watson of Leicester University, consists of eight astronomical institutes in the UK, France, and Germany.[2]

3. Survey Science Centre Tasks

Before launch the SSC is helping to design and build the science data analysis software in conjunction with ESA's XMM Science Operations Centre (SOC) at ESTEC. A common set of software modules will be used in the pipeline processing system to produce the standard products, and in the interactive analysis system which will be made available to guest observers.

After launch, the SSC will process all XMM observations and slew datasets (perhaps 600 MB/day) to produce source lists, images, spectra, and time series. The quality, reliability, and completeness of these pipeline products are of high importance, and the SSC team members will inspect and validate the results. The SOC will distribute the validated products together with raw data and calibration files to observers, and will add them to the XMM science archive.

Another major responsibility of the SSC is that of finding identifications for the serendipitous detections in the X-ray images, and of supporting a programme of follow-up observations on data which have entered the public domain. This is designed to complement, rather than compete with, individual observer's own programmes, and the results will be publicly accessible through the XMM science archive at the SOC.

4. Software Development Issues

The high spatial and spectral resolution of the XMM instruments and the availability of simultaneous optical/UV data, present many challenges to the design of a data analysis system. In the pipeline a predefined sequence of tasks must execute with high reliability, with all operations carefully logged, while the interactive system needs a user-friendly GUI-style front-end. The pipeline will run on a Sparc/Solaris platform, but portability is of great importance since the interactive analysis software must run on a wide range of platforms.

The existing astronomical software environments (IRAF, MIDAS, Starlink/ADAM, AIPS) provide a wealth of applications, but most are designed to handle optical or radio datasets rather than the photon-event lists which X-ray telescopes produce. Each environment has also invented its own file formats, none of them sufficiently flexible or widely-used to be an attractive prospect. Indeed, no single environment has attracted unequivocal support from the X-ray community.

For this reason, XMM software will be not be dependent on any environment (except the operating system) and will handle data stored in FITS files, principally FITS binary tables. Interfacing by FITS simplifies many aspects of the design, and also allows many other astronomical packages to be used easily, including FTOOLS, XSPEC, and SAOimage. We have some concerns over the

---

[2]Department of Physics & Astronomy, Leicester; Mullard Space Science Laboratory, University College London; Observatoire de Strasbourg; Service d'Astrophysique (CEA/DSM/DAPNIA), Saclay; CESR, Toulouse; Max-Planck Institut für Extraterrestrische Physik, Garching; Astrophysikalisches Institüt Potsdam; and the Institute of Astronomy, Cambridge

I/O overheads of the FITS format, but with the use of buffers and disc caches where appropriate, we expect these to be tolerable. XMM software is likely to make use of mature and well-supported libraries such as FITSIO, PGPLOT, and SLALIB. New software will be developed, where appropriate, under the general guidance of the ESA Software Standard, PSS-05. Special attention is being paid to software quality management and the software testing programme.

We foresee the need for a database management system at Leicester to manage the data flow through the pipeline, and there are similar requirements at Strasbourg and MSSL. We are investigating object-oriented and object-relational DBMS to see if these can handle XMM scientific data, as well as the management tasks. Commercial software licences will not, however, be needed by users of the interactive system.

The main processing pipeline is likely to contain at least 30 tasks and operate on up to 1000 raw data files each day. The recovery from failures in such a system can be very labour-intensive when conventional scripts and batch jobs are used. We are therefore investigating the use of an event-driven scheduler, such as the OPUS system used at STScI (Rose et al. 1996).

5. Programming Languages: C, C++, and FORTRAN

The SSC will use FORTRAN (which, all earlier forms being obsolete, now means FORTRAN 90) as the primary language for new scientific software. Since this goes against the current fashion, it may be appropriate to explain our reasons.

Object-oriented (OO) programming, it is often claimed, reduces development and maintenance costs particularly by promoting code re-use. The experiences reported in the 1995 ADASS meeting of those using OO programming in astronomy are, however, not wholly encouraging. See, for example, Glendenning (1996) on the AIPS++ project. Clearly, the up-front costs of switching to the OO paradigm can be substantial, especially in training the software development staff. Although C++ is the most widely used OO language, there are many who consider it to be by no means the best.

In my view, apart from the inheritance of methods, which is essential for OO programming, C++ has only two significant advantages over FORTRAN: a built-in exception-handling mechanism, and templates (which make it easier to create generic functions than in FORTRAN). For scientific programming, however, FORTRAN clearly has many substantial advantages over C++ (and indeed C):

1. FORTRAN arrays are first-class objects: this allows whole-array expressions and assignments, a compact array-section notation, and intrinsic functions which operate element-wise. C and C++ arrays turn into pointers when passed to a procedure, which is especially awkward when multi-dimensional arrays are involved.

2. FORTRAN has a better set of built-in data types, for example `logical` type distinct from `integer`, true character-string type (not just arrays of single `char`), and has identical facilities in single and double precision.

3. FORTRAN has better encapsulation of data and procedures in its modules than C++ has in classes, because there is no artificial split between code and header files. FORTRAN's `use` statement also provides better name-space management.

4. Dynamic memory can be allocated and accessed without using pointers in FORTRAN, where pointers are only needed to handle complex dynamic structures like trees and linked-lists, and are entirely type-safe.

5. FORTRAN allows user-defined operators, whereas in C++ one can only overload existing operators.

6. FORTRAN allows generic names to be used for procedures in a predictable and efficient manner since all names are resolved at compile-time.

7. FORTRAN programs are more portable than those in C++, because an ISO Standard exists and it carefully defines many features left as *implementation dependent* in the C++ standard, which is so far only a draft.

8. FORTRAN 90 provides many more features which allow errors to be detected at compile-time, for example argument `intent`, and explicit interfaces to procedures.

9. FORTRAN has a simpler syntax which makes it easier to write, and far fewer syntactic pitfalls than in C++, which it inherited from C.

10. FORTRAN programs usually execute faster because of built-in array features and powerful intrinsic functions, and because C/C++ pointers inhibit optimisation within loops. FORTRAN 90 also has better support for parallel or multi-processor architectures, a feature of increasing practical importance.

It is true that some of these deficiencies can be remedied by writing (or purchasing) a suitable class library, for example, one to handle arrays. But compilers are unlikely to generate code which is as efficient as if arrays had been built-in to the language. And if one makes use of higher-level class libraries from commercial or free-ware sources, they are unlikely to be based on the same array class, which can easily lead to code duplication, or other problems.

In choosing FORTRAN, we also took into account the need to make use of many existing libraries coded in FORTRAN 77, which are easier to access from FORTRAN 90 than from C++, and on the fact that good FORTRAN 90 compilers are now readily available for all major platforms, including PCs.

References

Glendenning, B. E. 1996, in ASP Conf. Ser., Vol. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes (San Francisco: ASP), 271

Lumb, D., et. al. 1996,[3] X-ray Multi-mirror Mission—an overview, Proc. 1996 SPIE Conference

Rose, J., Choo, T. H., & Rose, M. A. 1996, in ASP Conf. Ser., Vol. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes (San Francisco: ASP), 311

[3]ftp://astro.estec.esa.nl/pub/sciproj/xmm_mission.ps.gz