Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR TP 96573

Development procedures of the on-board attitude control software for the Sax satellite

G.J. Dekker

~

DOCUMENT CONTROL SHEET

	ORIGINATOR'S REF. **NLR TP 96573 U**		SECURITY CLASS. **Unclassified**

ORIGINATOR
National Aerospace Laboratory NLR, Amsterdam, The Netherlands

TITLE
Development procedures of the on-board attitude control software for the SAX satellite

PRESENTED AT
The 27th BIAS '96 Conference, Milan, Italy, 26-28 November, 1996.

AUTHORS **G.J. Dekker**	DATE **960919**	pp **21**	ref **11**

DESCRIPTORS

SAX satellite	**Satellite attitude control**
Applications programs (computers)	**Software reliability**
International cooperation	**Software engineering**
Italian space program	**Software development tools**
Maintenance	**X ray astronomy**
Program verification (computers)	

ABSTRACT
The Italian/Dutch SAX satellite, launched at April 30, 1996, is a scientific spacecraft, built to explore celestial X-ray sources. The software for the Attitude and Orbit Control Subsystem on-board computer for SAX consists of special purpose Basic Software, which performs operating system functions, and of Application Software (ASW). The ASW has been developed by NLR, under contract with Fokker Space. Main goal of the application software is to ensure an accurate and stable pointing of the satellite's main axis during periods of up to 28 hours. Due to the mission critical nature of the ASW, which shall be able to operate during extended periods without ground contact, it has to be very reliable. This software has been developed in two phases. At first, an interactive full scale development process has been executed, supported by a seperate independent integration and test team. This has led to a high quality end product, ready for AOCS subsystem intergration. While the AOCS subsystem integration tests were executed (by Alenia Spazio), a number of new requirements were introduced. These were implemented during the second (maintenance) phase of ASW development, during which a smaller development team worked on the software, while the ASW integration tests were combined with AOCS subsystem integration tests. The differences in PA during the full scale development phase and the maintenance phase are compared. The operational experience with the attitude control software during the first months after launch is documented.The PA approach during the software has proven its value by the shown high operational reliability and high flexibility of the software. The latter flexibility was needed to solve a number of operational problems in the AOCS sensors. The quality of the delivered software can be expressed as 0.05 detected errors per 1000 lines of (non comment) code, which is an exceptionally low value.

Summary

The Italian/Dutch SAX satellite, launched at April 30, 1996, is a scientific spacecraft, built to explore celestial X-ray sources. The software for the Attitude and Orbit Control Subsystem on-board computer for SAX consists of special purpose Basic Software, which performs operating system functions, and of Application Software (ASW). The ASW has been developed by NLR, under contract with Fokker Space. Main goal of the application software is to ensure an accurate and stable pointing of the satellite's main axis during periods of up to 28 hours. Due to the mission critical nature of the ASW, which shall be able to operate during extended periods without ground contact, it has to be very reliable.

This software has been developed in two phases. At first, an iterative full scale development process has been executed, supported by a separate independent integration and test team. This has led to a high quality end product, ready for AOCS subsystem integration. While the AOCS subsystem integration tests were executed (by Alenia Spazio), a number of new requirements were introduced. These were implemented during the second (maintenance) phase of ASW development, during which a smaller development team worked on the software, while the ASW integration tests were combined with AOCS subsystem integration tests. The differences in PA during the full scale development phase and the maintenance phase are compared.

The operational experience with the attitude control software during the first months after launch is documented. The PA approach during the software development has proven its value by the shown high operational reliability and high flexibility of the software. The latter flexibility was needed to solve a number of operational problems in the AOCS sensors.
The quality of the delivered software can be expressed as 0.05 detected errors per 1000 lines of (non-comment) code, which is an exceptionally low value.

Contents

(21 pages in total)

1 Introduction

The Italian/Dutch SAX satellite is a scientific spacecraft, with as mission to explore celestial X-ray sources (ref. 1). It is funded by the Italian space agency (ASI) and the Netherlands Agency for Aerospace Programmes (NIVR). The satellite is to perform a systematic and comprehensive observation of celestial X-ray sources in the 0.1-200 keV energy range with particular emphasis on spectral and timing measurements. The satellite (Fig. 1) has been injected into a circular equatorial orbit with an inclination of less than 5° and an initial altitude of 600 km by an Atlas Centaur launcher at April 30, 1996. The nominal mission lifetime is two years, with a design goal of four years.

The Attitude and Orbit Control Subsystem (AOCS) of the satellite has been developed by Fokker Space B.V. under contract with the satellite prime contractor Alenia Spazio (Roma, Italy). As subcontractor of Fokker Space, the National Aerospace Laboratory NLR has developed the Application Software (ASW) for the AOCS on-board computer (ref. 2).

This software has been developed in two phases. At first, an iterative full scale development process has been executed, supported by a separate independent integration and test team. This has led to a high quality end product, ready for AOCS subsystem integration. While the AOCS subsystem integration tests were executed (by Alenia Spazio), a number of new requirements were introduced. These were implemented during the second (maintenance) phase of ASW development, during which a smaller development team worked on the software, while the ASW integration tests were combined with AOCS subsystem integration tests. This approach has led to a reduction in the turn-around time for the implementation of new requirements and a reduction in the needed test effort, but, as consequence, some specification errors and software errors could remain undetected until the subsystem integration tests.

In the sequel, a brief introduction is given of the ASW (section 2), followed by a description of the development and maintenance processes (section 3). Section 4 describes the PA approach during the full scale development process, and section 5 describes the PA approach during the maintenance project. A quantitative and qualitative evaluation is given in sections 6 and 7, followed by the operational experience with the software during the first months of the operations of the satellite. The paper is finished with some concluding remarks.

2 The SAX AOCS Application Software

The AOCS subsystem hardware consists of (fig. 2):

- The Attitude Control Computer (ACC). This computer is based on a 80C86 microprocessor extended with a 8087 co-processor. The ACC is fully redundant. Two identical, independent, computers are integrated into one unit. One of the two is cold standby.
- A set of attitude sensors (Sun Acquisition Sensors, Magnetometers, Gyroscopes, and Star Trackers).
- A set of actuators (Reaction Wheels, Magnetic Torquer Rods, and a Reaction Control Subsystem).

The AOCS is controlled by software running in the ACC. This software is divided into two packages:

- Basic Software (BSW), developed by Alenia Spazio that provides operating system services and basic interface services. It hides the ACC hardware peculiarities for the Application Software.
- Application Software (ASW), developed by NLR. This software provides all attitude control tasks and manages the application-dependent communication with the ground and with the AOCS sensors and actuators.

The basic requirements on the AOCS and hence on the ASW software are to provide the capability to:

- Command the main instrument axis of SAX to any direction (within the pointing constraints) with an accuracy of 90" and the other axes with an accuracy of 16.5 arcmin. The commanded attitude must be maintained during pointing periods of maximal 28 hours.
- Perform health checking on the sensors and actuators and the related redundancy management functions.
- Safeguard the satellite against violation of the safe pointing domain. This safe pointing domain is defined by power constraints and by the fact that the main scientific instrument will be destroyed by (indirect) radiation from the sun. Therefore, the angle between the main instrument axis of the satellite and the sunvector must be at least 60°.
- Autonomously acquire and maintain a safe attitude when no ground commanding is available and after excursions outside the safe pointing domain.
- Process ground commands and generate relevant telemetry for health checking on the ground and for attitude reconstruction in relation with the processing of the acquired scientific data.

The ASW software has been divided into two parts:

- Basic Attitude Control (BAC) software. This part is highly reliable and safe. It is (initially) stored in Read Only Memory (ROM) in the ACC, together with the BSW. Main purpose of this software is to provide the functionality needed to acquire and keep a safe satellite attitude after power-up and fallback. Furthermore it contains the data handling functions needed to submit the state of the AOCS to the ground and to hand the control over (on ground command) to the EAC software. After an autonomous fallback to the BAC software, the ground operations team can analyze the reason for fallback and devise solutions to circumvent this reason.

- Extended Attitude Control (EAC) software. This software provides all functions required for the AOCS, including the control laws for accurate scientific pointings and the generation of full attitude reconstruction telemetry data. EAC is stored in Random Access Memory (RAM) of the ACC and can be loaded, activated and/or modified on ground command.

3 Development process

The software has been developed by NLR according to the rules of the ESA Software Engineering Standards (ref. 3), with as main exception the way of cooperation between Fokker Space and the NLR. The ESA standards cover the situation that the principal specifies his requirements in the form of a User Requirements Document (URD) which is used as sole input for the developer. In the case of the SAX AOCS software development, Fokker Space was not only responsible for the User Requirements of the software, but also for the algorithmic details of the application software (ref. 4). Based on a high-level URD for both the Basic Software and the Application Software, NLR has developed a Software Requirements Document and a Architectural Design Document for the ASW. The detailed design of the software, however, was developed in close cooperation between Fokker Space and NLR. Fokker Space detailed the algorithms for attitude control, unit health checking and redundancy management decisions. NLR developed the software design and the actual code. The algorithms were documented in so called 'Design Specification' documents, and in one document defining all relevant parameter values, the so called 'Parameter Interface Data Document (IDD)'. The approach thus followed can be characterised as an concurrent engineering process.

During the full scale development project of the ASW a separate NLR team acted as Independent Verification and Validation (IV&V) team. Using a dedicated real-time simulation environment for the ACC (SAX-TSA, ref. 5), this team developed validation tests for the integrated AOCS software (BSW and ASW), covering all aspects of the User Requirements Document.

The full scale development project was closed with the delivery of BAC and EAC software in the last quarter of 1993. Parallel with the finalisation of the development, AOCS subsystem tests were executed using the BAC software at Alenia Spazio. The AOCS subsystem tests applied a similar test environment, but with (flight representative) hardware sensors and actuators in the loop (ref. 6). This resulted in the existence of the following parallel activities:

- maintenance of the Design Specifications and Parameter IDD at Fokker Space;
- development and unit-level tests of new and modified software code at NLR;
- independent integration and validation tests of the software at NLR using SAX-TSA;
- subsystem integration tests at Alenia using SAX-TSA.

After the development, a new project was started to maintain the ASW, i.e. to update it according to new requirements and to the results of the AOCS subsystem integration tests. In view of budgetary and time constraints, it was decided by Alenia Spazio and Fokker Space to structure this maintenance around the execution of the subsystem integration tests. Fokker Space and NLR specialists were colocated at Alenia Spazio, shortening the communication lines. The development team at NLR was decreased to a core maintenance team. No separate validation of code changes was performed at NLR. The result of this approach was that Non-Conformance Reports (NCR's) could be solved efficiently and swiftly. During the subsystem integration, NCR's could be discussed on-site between Fokker Space, Alenia Spazio and NLR. Algorithm changes and the related (temporarily) code modifications were developed using Alenia Spazio facilities and the failed tests could be repeated immediately. However, in order to prevent a degradation of the overall software quality, the formal implementation of changes was done using the original software development environment at NLR, using the Product Assurance tools and procedures also applied during the full-scale development.

The maintenance has been executed this way from the end of 1993 to mid 1995. During this period, over 50% of the sourcefiles was modified at least once due to new and/or updated requirements.

4 Product Assurance approach during the full scale development project

The Product Assurance approach followed during the initial full scale development of the ASW consisted of the application of a number of Computer Aided Software Engineering (CASE) tools and the execution of a number of procedures, both for the development team and for the PA manager. All procedures were approved by Fokker Space for compliance with the SAX PA standards. They have also been certified to conform to the ISO 9001 standard.

- *Teamwork for structured analysis and design.* Teamwork has been used to input and check the Software Requirements and the Architectural Design. It supports the Structured Analysis method for (real-time) software systems as defined by Yourdon, DeMarco, Constantine and Hatley. The requirements and architectural design have mainly been described in Data Flow Diagrams and State Transition Diagrams. The data that is handled by the ASW has been described in detail in a Data Dictionary. Teamwork includes a powerful checking option to ensure the internal consistency of the thus obtained functional decomposition.
  The Detailed Design of the ASW has been developed using the Structured Design method, also supported by Teamwork. Each individual task of the ASW is defined with one Structure Chart and a number of Module Specifications. Also for this development phase, Teamwork provides a number of consistency checks, although these are less powerful than the checking options for the Structured Analysis method.
- *Design standard, supported by a checking tool.* A standard has been defined for the detailed design information, contained in the Module Specifications. This standard consists of the definition of a pseudolanguage containing the basic constructs for structured programming. Due to the simplicity of this language, it was an easy task to develop a software tool that checks whether the design adheres to this standard. This tool has been applied by the developers before the coding process of every module. During the preparation of deliverable (hardcopy) design documentation, all modules were checked (again) by PA for adherancy to the standard.
- *Automated code skeleton generation from the Teamwork design information.* Based on the simple design language, a tool has been developed to generate a code skeleton for every module from the Teamwork Module Specification information. Such skeleton contained the following information:
  * A copy of the detailed design text as comment;
  * The proper *#include* files to be used for internal interfaces;
  * The detailed design text, in which the logic constructs were converted to compilable C constructs.

The coding of the module could then be limited to a manual conversion of the assignments, algorithms and so on to proper C code.

- *Code standard, supported by a standard-checking tool.* A standard has been defined for the use of the ANSI C language for the ASW. The main aim of this standard was to limit the use of C-constructs to a small, reliable and testable subset and to force a uniform coding style during the complete development. As for the design standard, also the coding standard was enforced by a checking-tool. No code could be brought under configuration control that did not obey to this coding standard.
- *Module test standards, based on code coverage analysis tools.* The module testing was supported by a (public-domain) test coverage tool. Module tests had to execute every statement of the code at least once. In those (rare) cases where this requirement proved to be too expensive, code walkthrough by an independent programmer was allowed after prior approval of the PA manager.
- *Configuration control tools and procedures.* All flight code was stored under strict configuration control, including the test descriptions, test code, test in- and output, and the related make-files. The configuration control procedure included a change review by all Work Package Managers involved. After implementation of an approved change, independent checks were executed by the Configuration Control Manager that the code was complete, that the design- and coding standards were adhered to, and that the module testing yielded complete coverage and was repeatable using the latest version of all software under configuration control. As each code module contained a copy of the design specification in the heading comment, a separate check was prescribed to ensure that the detailed design information in the Teamwork model was identical to this comment (also after module changes).
- *Automated document production from the Teamwork design information.* Using the standard interface software between Teamwork and the DeskTop Publishing package Framemaker, the design information has been transformed to printed documentation without introduction of errors. This automated conversion also minimised the required effort.
- *Requirements traceability matrices.* At every formal update of the hardcopy documentation, a independent manual check was done by verifying the traceability between the latest version of the design specifications and the detailed design. This check involved all ±650 detailed specifications.

In addition to these procedures, the notion of *egoless programming* (ref. 7) was imposed on the project team. All software developed in the project was considered to be owned by the complete team in stead of the individual developers. Consequently, it was not allowed to keep private copies of designs and/or source modules. The procedure to put code under configuration control included as final step the removal of the original source files from the working directory of the developer. The design and source of every module could be freely consulted and commented by all team members. Changes to the design and code could be prepared and proposed by any team member. A number of modules were thus the result of the subsequent contribution of three or more developers. Besides the obvious effect of redundancy in detail knowledge of the software, this way of working also stimulates early detection of errors.

5 Product Assurance approach during the maintenance project

During the maintenance project, the same Product Assurance approach was maintained as during the full scale development. However, as a result of the smaller team size, the independency of certain checks was reduced. The roles of module developer, Work Package Manager, and Configuration Control Manager could be executed by one and the same team member, however at different times. As it was recognised that this could have negative effects on the quality of the delivered work, the team members were stimulated to execute these roles conscientiously and to strictly follow the prescribed procedures.
A few procedural shortcuts were implemented, however. An instance was the production of executable images of the integrated code. The procedures prescribed that all tested code changes were put under configuration control before a formal executable image could be produced. As result of the requirement that the continued correct execution of all module tests was to be checked during the formalisation process, this process could take up to a few working days. During the initial development, it was proven that the process of formalising tested changes was transparent for the user of the executable. Therefore, it was decided that informal executable images could be used for formal testing at Alenia, on the condition that the related formal executable image was checked afterwards to be byte-for-byte identical with the informal one. This shortcut significantly decreased the turn-around time for requested changes and hence increased the efficiency of the entire AOCS subsystem integration team.

6 Quantitative evaluation of the approaches

During the independent validation tests, executed as part of the initial development project, a total of 146 NCR’s have been generated by NLR. In view of the development approach mentioned above, the quality of the NLR contribution to the Application Software can be measured in terms of NonConformances that could be solved by changing the ASW solely, without a corresponding change in the Design Specifications or the Parameter IDD. In the remainder of this paper, these NCR’s are indicated as "ASW-related NCR’s". Ref. 5 gives the following classification of these NCR’s:

- 27%: Basic Software;
- 27%: SAX-TSA test system, including the environment simulation software;
- 23%: Design Specifications or the Parameter IDD;
- 9%: Indication of unclear or inconsistent requirements;
- 8%: ASW-related NCR’s;
- 4%: Hardware ("Functional") Model of the ACC, that was used during the tests;
- 2%: other causes.

In total, 11 ASW-related NCR’s were generated during these validation tests. During the (independent and parallel) AOCS subsystem integration tests using the same ASW software, no ASW-related NCR’s were generated by Alenia.
The size of the initial version of the ASW was ±350 sourcefiles containing in total ±16,000 lines of non-comment source code. The quality of the code *before* validation tests can thus be expressed as being 0.7 errors per 1,000 lines of code (KLOC). In literature, figures are stated between 1 and 25 errors per KLOC for delivered code *after* validation tests (ref. 8).

During the maintenance project, the code modifications were validated by Alenia in the scope of the subsystem integration tests. During this period, a total of 10 ASW-related NCR’s was generated by Alenia. Of these, 2 NCR’s could be traced to bugs that originated in the initial development project. Internal NLR inspections on the delivered code led to two more NCR’s, related to wrong parameter values in the delivered code.
During the maintenance period, the size of the software increased to ±410 sourcefiles with ±20,000 lines of non-comment source code. It is difficult to measure the percentage of code that was modified during this period. An estimate is that ±8,000 lines of code has been changed and/or added during this period. This leads to a quality figure of about 1.5 errors per 1,000 lines of new/modified code *before* validation. The higher number of errors is probably caused by:

- the higher complexity of the maintenance task. It is well known that the change process of complex software is more error prone than original development of code;
- the (small) margins in both processor speed and memory size forced the maintenance team into optimisation of the code, which increased the software complexity further.

During the SAX satellite integration tests no ASW-related NCR’s were generated at all. During the operational use of the software (after launch) one minor NCR was generated (refer to section 8). This leads to an error density *after* validation and delivery of 0.05 errors/KLOC, which can be categorised as an extremely low value, refer also to ref. 9.

7 Qualitative evaluation of the approaches

Comparing the approaches during the two phases, the following topics are considered:
- Development budget.
- Development time.
- Quality.
- Flow of information.
- Organisational aspects.

7.1 Development budget

The two phases differed mainly in the absence of a separate IV&V team during the maintenance project. The possibility of redundant test activities executed by the IV&V team at NLR and the (parallel operating) subsystem integration team at Alenia was thus effectively removed. This led to an obvious cost saving.
On the other hand, the subsystem integration team had to spend more time on verification testing. This, together with the colocation costs, led to an cost increase.
Although difficult to quantify, is seems that the overall effect of the maintenance approach was a reduction of the cost.

7.2 Development time

During the maintenance project many requirement changes were triggered by the results of subsystem integration tests at Alenia. These results could be discussed on short notice between Alenia and the colocated teammembers of Fokker Space and NLR. A copy of the software development system was also available at Alenia. Thus, proposed (requirement) changes could be implemented quickly and tested in the form of informal software versions. This approach led to a short turn-around time for informal changes, which was not possible without the colocation and duplication of the development environment.
As in the process of formalising changes the separate IV&V tests were skipped, formal software versions could be made available to Alenia earlier. These formal versions were transferred electronically from NLR to Alenia, minimising transfer delays.
The decision to skip the separate IV&V tests complicated the process of finding the cause of NCR's. Especially in the field of testing the unit health checking software, it was sometimes difficult to decide whether a problem was caused by the detailed requirements, the software implementation, the unit simulation software, or the actual hardware in the loop.

7.3 Quality

The effects on quality are difficult to qualify. It was perceived during the maintenance project that iterative changes, triggered by NCR's and implemented under time pressure, could lead to

a reduction of the quality of the end product. Therefore, the procedure explicitly documented that code changes that were developed on-site were to be regarded as informal fixes. The formalisation of requirement and code changes was done at Fokker Space and NLR premises as a separate activity and followed all prescribed change review and testing procedures.
Although a small team worked on the software during the maintenance project, this approach enabled the consultancy of the complete original development team to minimise the risk of unwanted side effects of modifications. In a number of cases, the formal review process identified such errors, that were difficult to detect by testing.

7.4 Flow of information

As result of the partial colocation of the project teams at the Alenia premises, the communication lines between all parties involved were short. This improved the quality of the information flow, on the expense of the colocation costs.
It should be noted that the future possibilities of Computer Supported Cooperative Work, enhanced with high bandwidth international network connections, will offer similar improvements of information sharing without the expenses of colocation. Both NLR and Alenia are involved in European research projects to study these techniques (the RACE PAGEIN project, ref. 10 and the ACTS MULTICUBE project, ref. 11).

7.5 Organisational aspects.

The approach to form a colocated multi-organizational team during subsystem testing can lead to confusion with regard to the organisation and the decision-making process. Therefore, the hierarchical structure of contractor/sub-contractor/sub-sub-contractor was strictly maintained during the maintenance project. Although informal working agreements could be made between parties, all formal decisions were taken along the hierarchical lines. The author believes that this was beneficial to the project.

8 Operational experience

After the successful launch of the satellite, an in-orbit checkout period of about two months was entered to check the behaviour of the complete satellite. The first subject to be tested was the AOCS system. All operational modes, except for the non-nominal and dangerous mode to execute an orbit change, were extensively tested. During this period, the ASW functioned (almost) flawless, but a number of unexpected phenomena occurred in the sensor packages:

- One of the four active gyroscopes performed out of scope. The on-board health check algorithms to compare the gyro package against the sunsensors (by far the most complex health check algorithm) detected this problem. After a reconfiguration of the on-board software and changing a number of on-board data conversion parameters a spare, cold-redundant, gyroscope could be used. This led to a new set of four active gyroscopes.
- The background noise, measured by the Star Trackers, was influenced more than anticipated by the difference between the sun-lit and eclipse periods. This effect reduced the operational usability of the Star Trackers. To compensate for this effect, the algorithms to control the Star Trackers was changed such that a calibration of the background noise was performed before each search-and-track command.
- The signal level of the main sunsensors appeared to degrade somewhat. Although this did not affect the accuracy of the measured sunvector, the level at which these sensors were declared ’dark’ had to be changed in the safeguard routine in order to prevent the (false) detection of inconsistencies between all sunsensors.

One minor software error was detected in the ASW software for the ’Default Pointing Mode’, which is used as backup control mode in case no scientific work is done with the satellite. This bug could lead to a (harmless) attitude error of maximal 10 arcminutes, which was still within the overall accuracy requirements. From the point of view of the operational crew, this error did not have any impact on the operations of the satellite. But, as the ASW had to be updated already in view of the Star Tracker problem, it was decided to repair this software error also.

The overall conclusion after the checkout period was that the ASW had proven its reliability beyond doubt. The built-in flexibility of the ASW enabled a proper handling of the sensor problems and enabled the successful completion of the checkout program.

9 Concluding remarks

In this paper, the approach has been described for the development of flight-critical application software. It has been shown that the approach followed during the initial full-scale development project has led to a high-quality end product. Some factors contributing to this high quality are:

- the implementation of the PA approach into the development suite;
- the enforcing of the standards by automated checking software;
- the automated information exchange between development and documentation tools.

After the initial development, an extended maintenance period was entered, during which a significant portion of the software was added or modified. During this period, a smaller team worked on the software, and one level of independent testing was omitted in order to save development time and budget. Although the maintenance of complex software is more error-prone than the initial development of such software, the quality level of the development process was only slightly reduced in the sense that a few more ASW-related NCR’s were to be solved during the validation tests of the modified software.

The operational experience during the in-orbit checkout period have proven that the Product Assurance approach followed during the development and maintenance of the software have indeed led to a highly reliable and yet highly flexible end product.

10 References

1. P. Maldari, et. al., *ESA Support for the Italian SAX Astronomy Satellite Mission*, ESA Bulletin, No 60 (1989).
   Refer also to http://www.sdc.asi.it/sax_main.html.
2. G.J. Dekker, G.J. Hameetman, *The development of flexible software for the Italian/Dutch satellite SAX*, AIAA Computing in Aerospace 8, Baltimore, Oct. 21-24, 1991. Also available as NLR publication NLR TP 91288 L.
3. *ESA Software Engineering Standards*, ESA PSS-05-0 (1991).
4. S. Kampen, J. Kouwen, *Design and development of the SAX-AOCS control software*, 1st ESA International Conference of Guidance, Navigation and Control Systems, Noordwijk, 4-7 June 1991, ESA SP-323 (1991).
5. F. Sonnenschein, M. Schoonmade, T. Zwartbol, *Space Systems Design and Development Testing*, Flight Vehicle Integration Panel Symposium, 3-6 October 1994, AGARD-CP-561.
6. G. Battistoni, C. Cassi, M.T. Ravazotti, *SAX AOCS subsystem simulation and tests, results evaluation and plan for future developments*, 2nd ESA International Conference on Spacecraft Guidance, Navigation and Control Systems, 12-15 April, 1994.
7. G.M. Weinberg, *The psychology of Computer Programming*, ISBN 0-442-29264-3,Van Nostrand Rheinhold Company, New York, 1971.
8. W.W. Gibbs, *Software's Chronic Crisis*, Scientific American, september 1994.
9. L. Hatton, *Software faults: the avoidable and the unavoidable: Lessons from real systems*, Proceedings of the ESA 1996 Product assurance Symposium and Software Product Workshop (ESA SP-377), Noordwijk, March 1996, pp. 271-275.
10. U. Lang, *A Software Environment for Cooperative Simulation and Visualisation in the aerospace Field*, Proceedings of the HPCN Conference, München, Springer Verlag, april 1994.
    Refer also to http://visu-www.onera.fr/PAGEIN.
11. (anon.) ACTS project MULTICUBE, refer to http://drogo.cselt.stet.it/sonah/MULTICUBE/index.html.