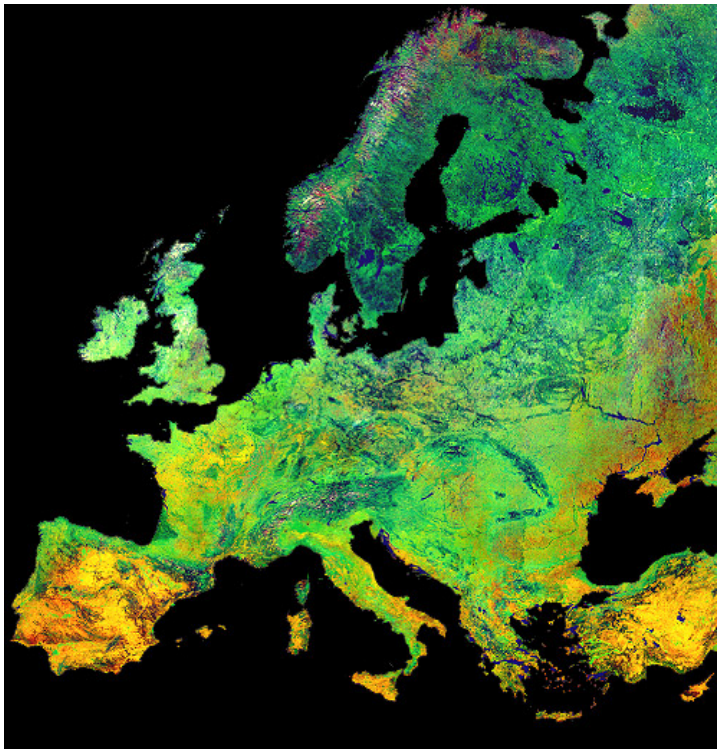
The Significance of Flight Software for Space Programmes
A European Learning Curve



Jack Bosma
European Space Agency
TRISMAC Keynote Address
Noordwijk, 14 April 2008

Evolution of Robust Software In ESA

- envelope of the European Space Agency has grown in scope and diversity

- flight software follows this trend and plays an increasingly deterministic role in
  - the definition of the system architecture
  - the complexity and duration of the on-ground test and verification phase
  - the flexibility of in-orbit operations

- ATV & Giove-A provide two case studies of how ESA is adapting to robust last resort software for programmes

The ATV Case

- ATV is unique in combining both the full automatic capabilities of an unmanned vehicle able to rendezvous and dock on its own, and the human spacecraft safety requirements when it is docked to the ISS

- Rendezvous of Jules Verne with the International Space Station (ISS) on 3 April was a tremendous boost for the space community in Europe.

- Prior to making the ultimate rendezvous attempt the ATV had to demonstrate its robustness.

- One of the most critical in-orbit tests to be carried out was called the Collision Avoidance Manoeuvre or CAM

ESA
ATV Docked to ISS

The ATV CAM

- CAM is necessary to reliably move ATV away from the ISS in case of problems during the final rendezvous & docking with ISS

- Upon detection of a critical failure or an unsafe situation, the spacecraft's Monitoring and Safing Unit (MSU) is designed to isolate the ATV's nominal systems and issue the CAM command

- CAM is a back-up mode and as it involves shutting down all of the normal control systems

- if any problems occur during the in-orbit demonstration it would have been difficult to recover the spacecraft

The MSU Software

- Failure Modes, Effects and Criticality analysis (FMECA) & Hardware/Software Interaction Analysis (HSIA) proved too high residual risk of the nominal software
- at the Preliminary Design Review (PDR) decision for a completely segregated, simple computer, known as the Monitoring and Safing Unit (MSU)
- MSU software monitors the complex nominal system and, upon detecting an unsafe state, takes over control and triggers a collision avoidance manoeuvre to prevent any potential accident
- Upon detection of a critical failure or an unsafe situation, the spacecraft's Monitoring and Safing Unit (MSU) is designed to isolate the ATV's nominal systems and issue the CAM command

The MSU Software Features (1)

- **most stringent product assurance methods, approaches and tools to ensure safety and reliability**

- **Requirements**
  - **Modeling of requirements using a mathematical tool and verification of the model**
  - **Full traceability of requirements to design, code and tests**

- **Design**
  - **Emphasis on a simple design as driver (Data flow driven design with reduced states)**
  - **Full execution determinism ensured by design**
  - **The verification of the design included the use of mathematical models of the state machines and validation by a tool**

The MSU Software Features (2)

- **Coding**
  - **The code was subject of a metrication program with strong thresholds to ensure simplicity, testability, modularity and readability.**
  - **Coding rules were strictly applied**
  - **Definition of operational range for each interface, checked during acquisition**
  - **Internal data exchanges checked to avoid error propagation**
  - **Numerical errors prevented by systematic analysis of all arithmetical operations**
  - **All analysis, inspection fully documented in detail**
  - **No code reuse to avoid potential use in inappropriate context**
  - **Real time kernel reduced to its minimum and fully tailored for MSU needs**
  - **All hardware error protection features exploited (memory corruption protection, permanent Built-In-Tests, Watchdog)**

The MSU Software Features (3)

➢ Verification and validation

- **Extensive unit test ran on target with 100% coverage at assembly level including the kernel and removal of unreachable or unused code**
- **Software modules integration performed rigorously**
- **Comprehensive scope for validation testing (requirements coverage, numerical performance, stress, long duration test, correct initialization, RAM mapping)**
- **Independent Software Validation and Verification performed throughout the complete life cycle (active participation to reviews, code inspection, independent unit and integration tests on dedicated facilities)**
- **Trends were analyzed on software problem reports (SPR) to support quality control**
- **Regression testing strictly performed.**

The MSU Software Features (4)

➢ Dependability and safety assessment

- **RAMS were assessed throughout all incremental development cycles; since the start and for each evolution or modification introduced in the software. This was achieved by performing a functional Software Error and Error Analysis (SEEA), followed by an analysis at the design stage of potential failures that could be introduced by the software**

- **A final synthesis of the recommendations raised in these analyses showed that all recommendations had been taken into account.**

The MSU Software Features (5)

➢ Metrication Programme

- **The metrication program permits to derive quality properties from direct measures performed on code**
- **The quality properties targeted during the software development with the related metrics are presented in the table below**

Quality Property	Metric
Simplicity	Nesting levels
Simplicity and testability	Direct call number
	Cyclomatic complexity
Modularity	Number of statements
Readability	Rate of comments
Efficiency	Time performances
	Memory used size

The MSU Software Features (6)

➢ Metrication Programme

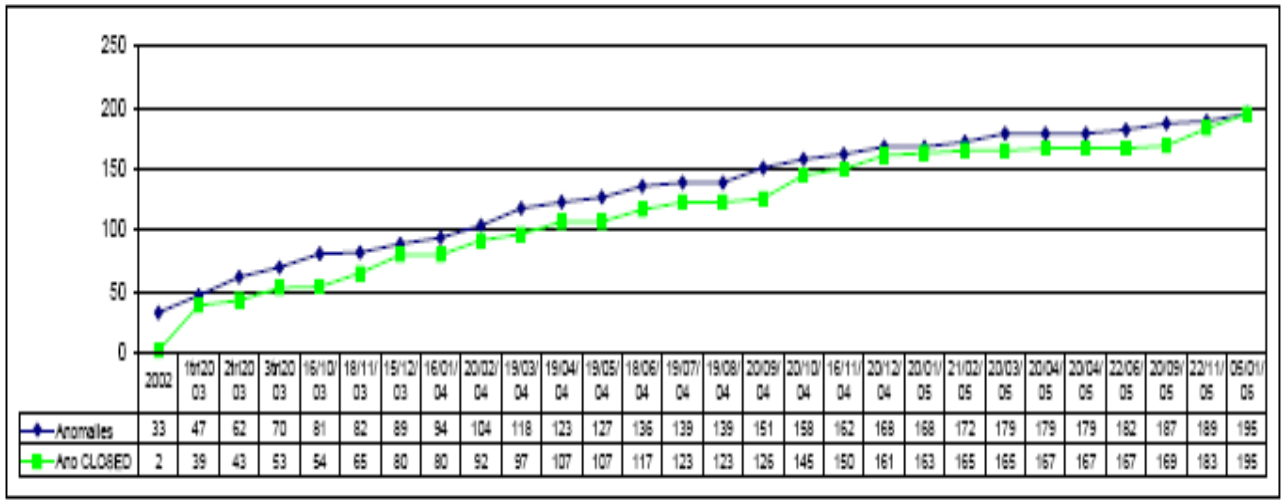
- **The table below presents some metrics and thresholds used for MSU and, for comparison, those used for the main, nominal system. Note that the thresholds for MSU, being of higher criticality, are significantly stricter than those for the main software.**

	Main SW	MSU SW
Software category	C	A
Number of ADA code lines	~250 000	~10 000
Number of executable statements	~45 000	~3 800
Exec code size (byte)	> 4 M	< 60K
Number of functional requirements	Over 3500	166
Metrics Thresholds (per function)		
Number of statements	< 250	< 100
Nesting levels	< 7	< 5
Direct call number	< 15	< 8
Comment rate	> 15%	> 30%
Cyclomatic complexity	< 21	< 11

The MSU Software Features (7)

➢ NCR Trend Analysis

- **Most of the faults detected in the MSU software were discovered during code inspection and the unit testing phase, and very few at integration and validation testing; none during qualification**



	2002	16/12/03	26/12/03	30/12/03	16/10/03	18/11/03	15/12/03	16/01/04	20/02/04	19/03/04	19/04/04	19/05/04	18/06/04	19/07/04	19/08/04	20/09/04	20/10/04	16/11/04	20/12/04	20/01/05	21/02/05	20/03/05	20/04/05	20/04/05	22/06/05	20/09/05	22/11/05	05/01/06
Anomalies	33	47	62	70	81	82	89	94	104	118	123	127	136	139	139	151	158	162	169	168	172	179	179	179	182	187	189	195
Ano CLOSED	2	39	43	53	54	65	80	80	92	97	107	107	117	123	123	126	145	150	161	163	165	165	167	167	167	169	183	195

The MSU Software Features (8)

➢ Summary

- **MSU software submitted to one of the most rigorous engineering and software PA development programmes in the European Space Agency to date.**

- **Early definition and strict coherence to design and code metrication rules proved their value.**

- **Most problems have been identified during code inspection and unit testing with very few in the integration & validation phase and none at all during the qualification phase**

ESA
Galileo Navigation System

The Giove-A Case

- As part of the design and development phase of the European Galileo satellite navigation system ESA commenced a test satellite phase called Galileo System Test Bed-V2 (GSTB-V2) in 2003

- The launch of Giove-A in December 2005 concluded a tight schedule development programme and allowed ESA to secure the frequencies allocated by the International Telecommunications Union (ITU) for the European Galileo system

- Giove-A like ATV has robust last resort software that resides in the Attitude Safety Module (ASM).

The Giove-A ASM

- ASM acts as a watchdog to the flight computer and takes control in the event of any anomalies that require the satellite to be placed in a safe mode.
- Although the ASM has not been tested as rigorously as the ATV MSU software, the test programme was nevertheless very intense for a satellite test bed.
- Given the tight schedule not all contingencies of the nominal flight software were fully tested
- Consequently Giove-A relied more heavily than usual on the ASM software as a fall-back to correct for in-orbit anomalies.
- The performance of the ASM in orbit has been flawless and the whole fault detection and recovery algorithms work extremely well
- Giove-A remains operational to date and will be accompanied by GIOVE-B now in line for launch from Baikonur scheduled for 27 April.