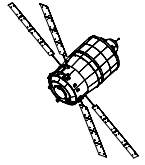


Automated Transfer Vehicle (ATV) Mission and Safety Critical Software Development



**Klaus Ludwig, Eric Zekri,
Marie-Odile Devic
European Space Agency
3rd IAASS Conference
Rome, 23 October 2008**



ATV Jules Verne Mission

- **ATV Jules Verne just completed its maiden flight as an unmanned logistic vehicle**

Launch: 9 March 2008
Docking to ISS: 3 April 2008
ISS-Undocking: 5 September 2008
ATV Re-Entry: 29 September 2008



- **ATV flight software has been a key to the mission success of Jules Verne**
- **The ATV flight software consists of mission critical software (FAS) and safety critical software (MSU SW)**
- **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**

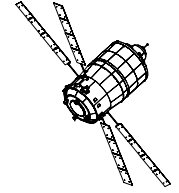


Flight Application Software (FAS)

- **Mission and Vehicle Control**
- **Guidance, Navigation & Control**
- **Survival Mode**
- **Flight Control Monitoring**

Monitoring and Safing Unit Software (MSU)

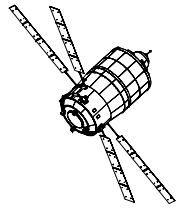
- **Functional Monitoring**
- **Coarse Monitoring**
- **CAM Sequencer**
- **Navigation & Control**



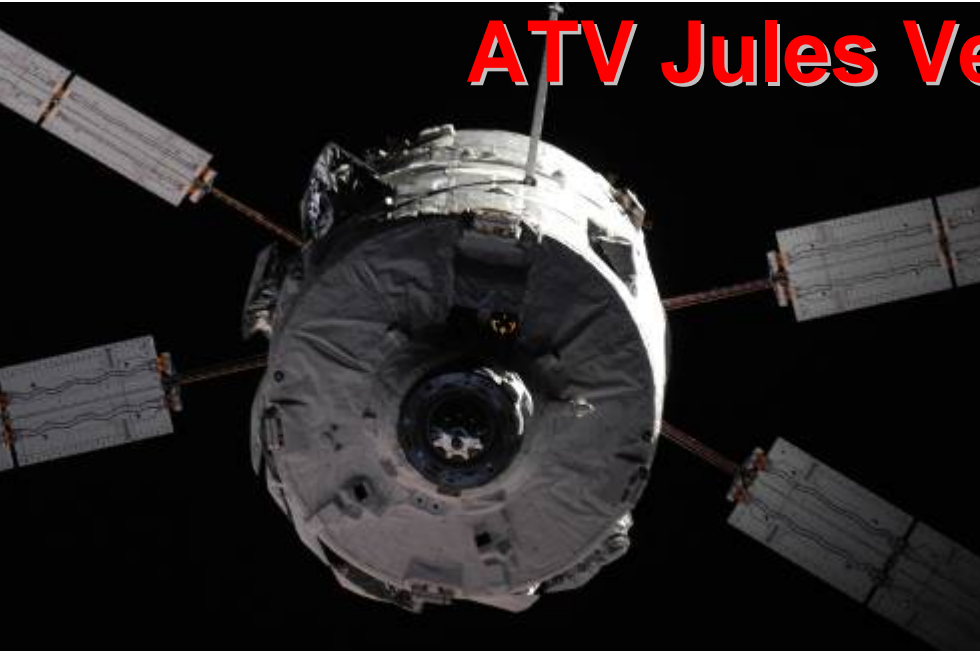
ATV Jules Verne Mission

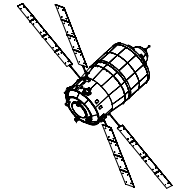
- Rendezvous of Jules Verne with the International Space Station (ISS) on 3 April was a tremendous boost for the space community in Europe.
- Prior to making the ultimate rendezvous attempt the ATV had to demonstrate its robustness.
- One of the most critical in-orbit tests to be carried out was called the Collision Avoidance Manoeuvre or CAM
- 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 as it involves shutting down all of the normal control systems



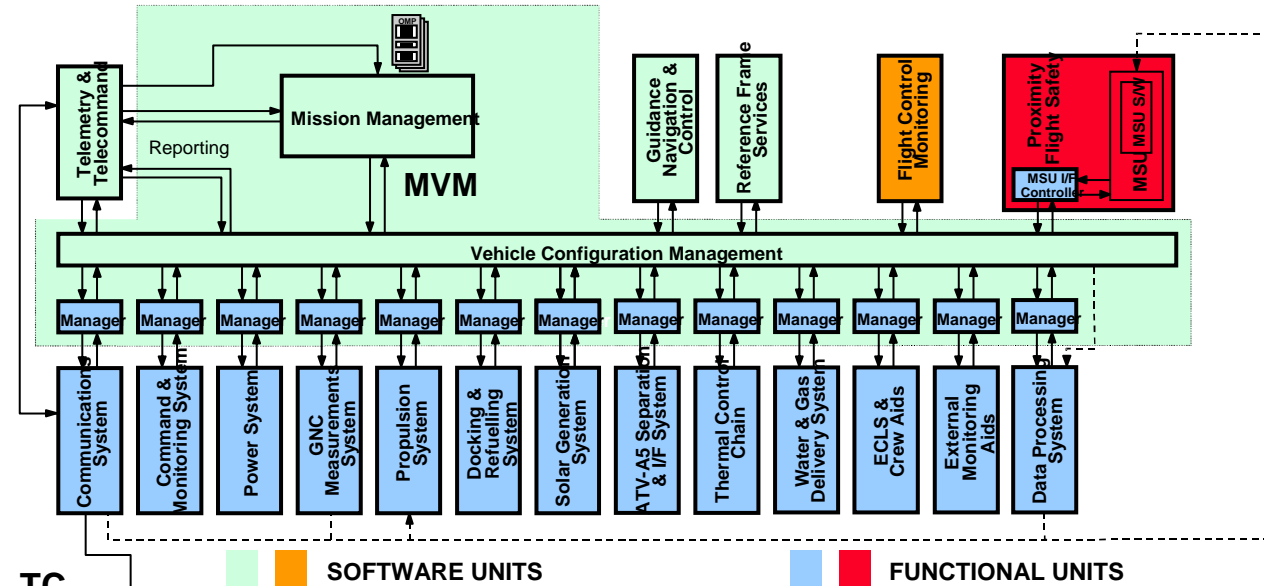


ATV Jules Verne Mission





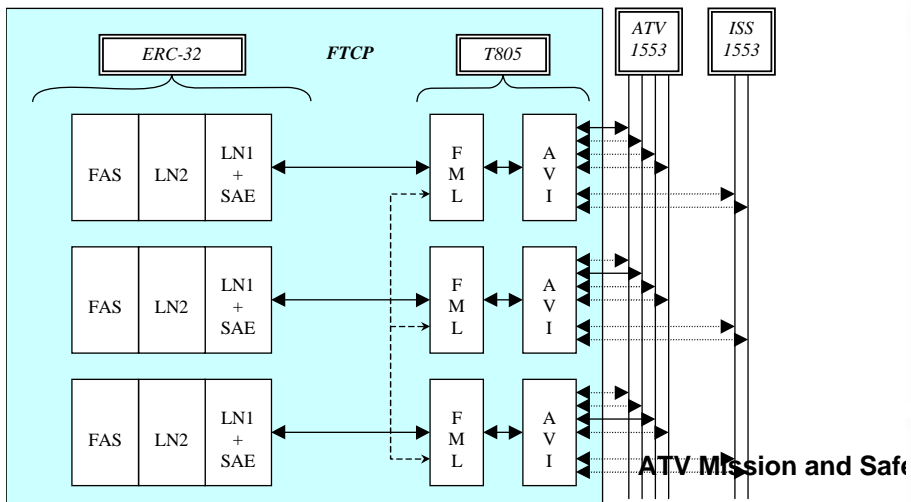
ATV Software Architecture



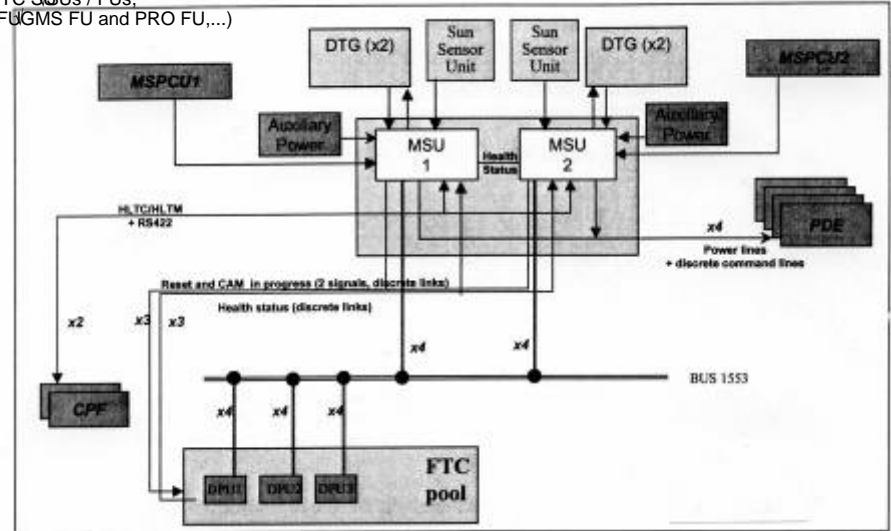
■ **SOFTWARE UNITS**
■ **FUNCTIONAL UNITS**

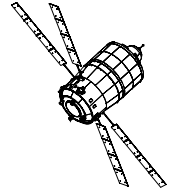
TC (Telemetry & Command)
TM (Telemetry)

Some internal interfaces not shown (TM/TC SISUs / FUs, GMS FU, GNC SU and FCM SU, PFS FUGMS FU and PRO FU,...)



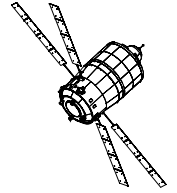
ATV Mission and Safety





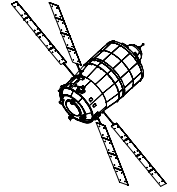
Flight Application Software (FAS)

- Although originally not classified as mission critical software, features of mission and safety critical have been added at PDR time to the FAS SW life-cycle
- FAS has been designed as highly data driven software, able to be “missionised” via the Mission Database and has been kept simple in its real time behaviour (4 task)
- 100% decision coverage on source code level has been ensured at unit level tests proving the absence of dead code
- Code and test procedures inspections have been performed by independent teams with predefined guidelines comprising criteria for
 - compliance with coding standards
 - assessing the complexity of the code
 - performing data and control flow analysis
 - special attention to evaluation of numerical stability of control algorithms
 - data context used for tests
- A tailored ISVV programme limited to the survival mode of FAS has been conducted by an independent ISVV contractor with the only goal to break the SW by
 - identification of vulnerable areas in FAS
 - identification of incomplete or missing tests
 - performance of stress tests



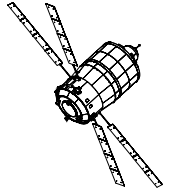
Flight Application Software (FAS)

- FAS has been developed in an incremental way with
 - full validation of each version
 - Software Error Effect analysis
 - Numerical validation of GNC algorithms on dedicated simulation facility
 - Comparison of GNC performances on target computer with simulation facility results
- FAS qualification had been completed prior to system level qualification campaign
- Later FAS versions have been qualified via a pre-defined regression test campaign
- Full re-qualification has been performed prior to launch
- A rigorous metrication programme has been followed through the entire life cycle of FAS
- At ATV launch all SPRs had been closed and no patch needed to fly
- During the Jules Verne mission only few anomalies have been detected, which can be fixed by a Mission Database update
- No FAS patch has been necessary during the mission



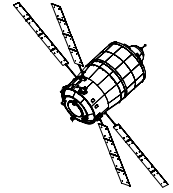
The MSU Software

- **Main Characteristics**
- **Due to its nature of being safety critical software, reliability and full testability have been design drivers from the very beginning, resulting in the following major characteristics of the software:**
 - **Simplicity and Modularity**
 - **MSU functions architected and coded as independent blocks**
 - **compact Ada code (16000 LOC)**
 - **Determinism & Freedom of Deadlocks**
 - **data driven**
 - **100% cyclic state automaton**
 - **Robustness and Numerical Stability**
 - **internal and explicit protection vis-à-vis numerical errors, computational divergences and error propagation**
 - **most stringent product assurance methods, approaches and tools to ensure safety and reliability**



The MSU Software

- **Requirement Phase**
 - Due to its synchronous properties, the MSU requirements have been modeled using SCADE and SCADE diagrams and dataflow dictionaries have been incorporated into the MSU SW specification
 - Full traceability of requirements to design, code and tests
- **Design Phase**
 - A simple 5 states automaton manages the MSU modules
 - Only one unique state per software cycle allowed in same cycle
 - Calls to sub-modules are ordered in data driven fashion avoiding deadlocks or starvation effects
 - 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)
 - RAMS were assessed by performing a functional Software Error Effects Analysis (SEEA), during the entire life cycle of the software. Any finding has been checked a a later stage for correction in the software.
 - No patching capability has been implemented in order to protect the integrity of the software



The MSU Software

➤ Coding Phase

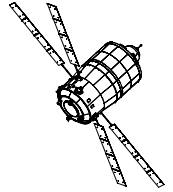
- Explicit protection of input and output data by check against predefined ranges
- Explicit protection of numerical computations
- Coherency of data enforced by data acquisition once per cycle
- Benefit if Ada built-in checks during testing on host
- Extensive code inspection according to pre-defined criteria with special attention to numerical stability

➤ Unit & Integration Test Phase

- Extensive unit test ran on target with 100% coverage at assembly level including the kernel and removal of unreachable or unused code
- Unit test performed on host, target simulator as well as target coupled with a 1750 emulator allowing breakpoints
- Compiler range checks used on host, without on target
- Integration test performed in an incremental way with same tools as for unit tests

➤ Validation Test Phase

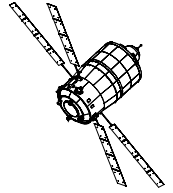
- Requirement verification
- Numerical performance incl. dedicated GNC validation on numerical test facility with many Monte Carlo runs
- Stress test & Long duration test (39 hours)
- Many scenarios with nominal and error conditions



The MSU Software

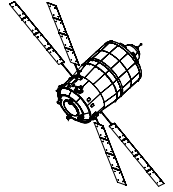
- **Independent Software Verification & Validation**
 - Entire SW life cycle covered by ISVV with active participation to all reviews, which allowed to correct any finding early in the SW life-cycle
 - Code and documentation review
 - Analysis of test approach and procedures
 - Independent Software Unit and Integration test performed on dedicated Test Facility
 - Stress Testing on dedicated Test Facility

- **Summary**
 - A total of 70 SPRs have been detected mainly during the unit test phase
 - No error has been found since qualification of the SW in 2006, neither during system qualification nor during the mission
 - MSU software submitted to one of the most rigorous engineering and software PA development programmes in the European Space Agency to date.
 - A rigorous metrication programme has been applied guaranteeing quality properties of the SW



Computer Based Control System (CBCS)

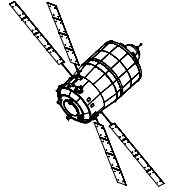
- **Compliance with NASA's ISS level requirement related to Software safety proven during NASA's Safety Review Panel meetings**
 - **Achieved by providing evidence of full traceability to ATV system and software must-work and must-not-work requirements**
 - **Achieved by identification of software functions involvement in the prevention of hazardous situations identified Hazard reports created by the ATV Safety community**
 - **Achieved by providing verification evidence to those software functions**



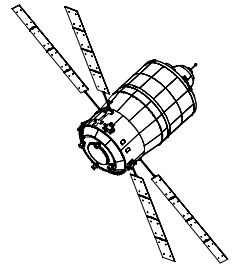
Conclusions

- Very stringent software engineering and quality standards and processes have been applied guaranteeing quality products
- ESA team collocated with EADS software team allowed very close cooperation on a day-to-day basis
- Rigorous metrification programme allowed efficient software development
- Proven robustness and reliability via ISVV not limited to safety critical software

Thank you for your attention!

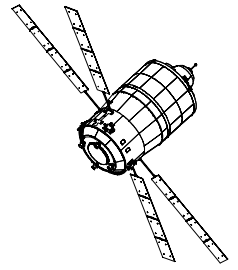


Back-up charts



HSF/ATV SW Categorisation

On Board SW categories	Category definition	Verification activities
Cat C SW	SW whose failure can NOT result in catastrophic consequences (loss of Life or Vehicle)	Documents review Code inspection Unit testing on host Integration testing on Host Validation testing on target
Cat B SW	SW whose failure could result in catastrophic consequences and adequate means of intervention are implemented in the system to prevent the occurrences of these consequences	Same as C + UT 100 % code coverage + Petri Net Analysis +Independent validation testing
Cat A SW	SW whose failure could result in catastrophic consequences and NO adequate means of intervention are implemented in the system to prevent the occurrences of these consequences	Same as B +Unit testing on target +Independent Code inspection +Independent unit testing +Independent integration testing +Independent validation testing



ATV SW in numbers

	FAS	MSU SW
Software category	B/C	A
Number of ADA code lines	430 000	15600
Number of executable statements	~45 000	~3 800
Exec code size (byte)	> 4 M	< 60K
Number of functional requirements	Over 3500	226
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

MSU SW

- **Unit testing:** 3086 test cases / 350 procedures
- **Integration:** 168 test cases / 68 procedures
- **Validation:** 723 test cases / 84 procedures
- **Qualification:** 32 scenarios / 16 with CAM

FAS

- **9802 test cases**
- **231 procedures(host) / 151 procedures(target)**
- **3931 test cases / 735 procedures**
- **68 scenarios**