

**Ariane 5  
Flight 501 Failure—A Case Study**

**Ken Robinson**

**Department of Software Engineering  
School of Computer Science and Engineering  
University of New South Wales**

**16th December 1996**

## Flight 501 Failure—First Information

H0 = 093359 on 4th June 1996

H0+7.5s	Ignition of solid booster stages and normal lift-off
Up to H0+37s	Flight guidance and trajectory normal. At this moment the velocity of the launcher was Mach 0.7 (807 kph) and its altitude 3,500m. Sudden swivelling of both solid booster nozzles up to the limit, recorded by telemetry.
H0+37s to H0+39s	This caused the launcher to tilt sharply, giving rise to intense aerodynamic loads on the launcher structure resulting in breakage. Following loss of the launcher integrity, destruction of all launcher elements by the on-board neutralisation system.

## Flight Control System of Ariane 5

- SRI: Inertial Reference System (duplicated, same hardware and software). Computes angles and velocities on the basis of information from a “strap-down” inertial platform, with laser gyros and accelerometers. The software is quite old and had been used in Ariane 4.
- OBC: On Board Computer (duplicated). Executes the flight program and controls the nozzles of the solid boosters and the Vulcain cryogenic engine, via servo-valves and hydraulic actuators.

## The Chain of Events

- SRI-1 ceases to function; OBC switches to SRI-2.
- SRI-2 fails due to a software exception.
- OBC cannot switch to SRI-1; takes data that is actually part of a diagnostic message written to the bus by SRI-2.
- This data was interpreted as flight data and used to control the solid boosters.

## Analysis

At the time of the failure, the software in the two SRIs was doing a data conversion from 64-bit floating point to 16-bit integer. The floating point number had a value greater than what could be represented by a 16-bit signed integer; this resulted in an overflow software exception.

## Comments on the Failure 1

To determine the vulnerability of unprotected code, an analysis was performed on every operation which could give rise to an exception, including an Operand Error. In particular, the conversion of floating point values to integers was analysed and operations involving seven variables were at risk of leading to an Operand Error. This led to protection being added to four of the variables, evidence of which appears in the Ada code. However, three of the variables were left unprotected. No reference to justification of this decision was found directly in the source code.

A memorandum was found to say the code should be written and made more robust later.

## Comments on the Failure 2

The error occurred in a piece of code that was meaningless after lift-off. The code was used only to stabilise the rocket during the count-down period.

The code runs for 50 seconds; approximately 40s after lift-off. The bug occurred in Ariane 5 after 36s. The period of 50s was based on the time needed for the ground equipment to resume full control of the launcher in the event of a hold.

### Comments on the Failure 3

In Ariane 4 flights using the same type of inertial reference system there has been no such failure because the trajectory during the first 40 seconds of flight is such that the particular variable related to horizontal velocity cannot reach, with an adequate operational margin, a value beyond the limit present in the software.

Ariane 5 has a high initial acceleration and a trajectory which leads to a build-up of horizontal velocity which is five times more rapid than for Ariane 4. The higher horizontal velocity of Ariane 5 generated, within the 40-second timeframe, the excessive value which caused the inertial system computers to cease operation.

## Comments on the Failure 4

It was the decision to cease the processor operation which finally proved fatal. The reason behind this drastic action lies in the culture within the Ariane programme of only addressing random hardware failures. From this point of view exception—or error—handling mechanisms are designed for a random hardware failure which can quite rationally be handled by a backup system.

## Comments on the Failure 5

This piece of software was not re-tested prior to the Ariane 5 launch.

The SRI specification (which is supposed to be a requirements document for the SRI) does not contain the trajectory data as a functional requirement.

## Comments on the Failure 6

When the project test philosophy was defined, the importance of having the SRIs in the loop was recognized. At a later stage of the programme (in 1992), this decision was changed. It was decided not to have the actual SRIs in the loop for the following reasons: (only two selected)

The SRIs should be considered to be fully qualified at equipment level.

The simulation of failure modes is not possible with real equipment, but only with a model.

## Conclusion

The preceding comments display a serious misunderstanding of (non-software) engineers to understand the nature of software. Software is being treated like a hardware component, but software failure modes are quite different to hardware.

## Acknowledgements

The information presented here has been taken from the report that  
can be found at

<http://www.esrin.esa.it/htdocs/tidc/Press/Press96/ariane5rep.html>