UNIVERSITY OF ANTWERP

The Ariane 5 Failure

Hoang Trung Hieu

December 10, 2014

1 INTRODUCTION

Ariane 5 is a rocket which is part of a series of European civilian expendable launch vehicles for space launch use. Ariane 5 was designed and developed by European Space Agency (ESA) as the replacement for the successful Ariane 4 rocket launcher. On 4th June 1996, the maiden flight of the Ariane 5 launcher ended in a failure which cost a loss of hundreds million USD and 1-year delay for the program.
A board of inquiry was established to start the investigation and on July 19th, the inquiry board released a full report on their explanation of the reasons for the failure. Moreover, the incident has been also motivated many studies from researchers analyzing the causes of the disaster and lesson learnt from multiple views.
Ariane 5 failure is believed to be a particular useful use case in illustrating sequence of issues in development process. This report will attempt to study the failure from engineering and behavior perspectives.
The report is structured as follows. Section 2 will give a short description of the Inertial Reference System (abbreviated SRI due to its French name) and a chain of technical events which caused the failure. In Section 3 and Section 4, we will go into detail on what caused the failure.

2 THE FAILURE

2.1 INERTIAL REFERENCE SYSTEM

Inertial Reference System is used to measure the altitude of the launcher and its movement in space. SRI has its own internal computer in which angles and velocities are calculated based on the information provided by inertial platform. Once the calculation is finished, the data

will be transmitted through the databus to On-Board Computer (OBC), which executes the flight commands and controls the nozzle that will bring the rocket on track.
In Ariane 5, there are 2 SRIs running in parallel to ensure the reliability of the system. If OBC detects that the active SRI failed, it immediately switches to the other one. The Ariane 5 SRI inherits the design and implementation of SRI from Ariane 4, particularly as regards the software[3].

2.2 What happened?

According to the Inquiry Board Report, a chain of technical events led to the failure of the Ariane 5

- During the conversion from 64-bit floating point to 16-bit integer, an software exception occurred in the internal SRI software
- The active SRI stopped working due to the software exception (Operand Error)
- The OBC could not switch to the second SRI as it was shut down as well
- The diagnostic bit pattern of the second SRI was sent to OBC.
- OBC erroneously interpreted the diagnostic bit pattern as extreme but valid data and commanded the nozzles to deflect to one side
- The rocket veered from its path, started disintegrating and eventually triggered the self-destruct system

Next sections will analyze more deeply on the main reasons why the incident occurred, from engineering and behavior perspectives.

3 Engineering Analysis

3.1 Non-diversified redundancy

In order to improve the reliability of the system, 02 SRIs were used to operate in parallel mode (hot standby). However, in the incident, the second SRI was shut down almost immediately after the failure of the active one. Two SRI ran the same software component and got identical exception which caused the processors stopped working.
In fact, designers of Ariane 5 wrongly assumed that an error should be caused by a malfunction of hardware only. Under such circumstance, the second SRI would be able to automatically take over the work of the active one. They had done proper redundancy mechanism if the failure was in hardware itself. Unfortunately, if the error comes from software then the back-up SRI would get the same problem and in fact there was no software redundancy in this case.
In a system like Ariane 5, the complexity supplied by computing services lies in their software rather than in the hardware layer and the software faults have been significantly prevalent.

System engineer should consider the need to provide two separate OBC tasks, one in charge of executing the flight commands while the other one was responsible for handling exception conditions. This will help to avoid the situation when both SRI are shut down due to non-diversified software redundancy.

3.2 Critical data validation

When the software exception was thrown, SRIs sent diagnostic bit pattern to OBC instead of valid inertial data. Even though such information is critical, the OBC did not do any further checking or validation on received data. Therefore, OBE considered erroneous, extreme values from SRI as legal and simply interpreted them to flight commands. In addition, no proof has been established showing that OBC module delivers correct input to flight task, in the presence of faulty values provided by SRIs. As the result, the nozzles were directed by wrong commands and lost its flight path.
The system did not specify operational restrictions from chosen implementation and did not handle excessive, non-compliant data that might appear. These identification and handler of such limitation was considered mandatory for every mission-critical device like SRI in Ariane 5. Unfortunately, the procedure was not followed.

3.3 Unprotected variables

As unhandled exceptions were the primary technical cause of Ariane 5 failure, we will discuss this issue in more detail than the other ones.
Software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. According to the Inquiry Board Report, there were seven variables at risk of leading to Operand Error exception but only four variables were protected with evidence appearing in Ada code.
Three of the variables (in Ada code) were left with no exception handler associated with the conversion. These were not simple programming or design error but intentional decision. Variables were not protected given incorrect assumption that they were either physically limited or that a large margin of safety were anticipated. The decision did not take into account the analysis of any realistic data, trajectory data were not used to analyze the behavior of the unprotected variables. Additionally, the implementation was completely inherited from Ariane 4 and trajectory data of Ariane 5 were not included into SRI requirements and specifications.
No reference to justification of unprotected variables was found in the source code comment and indeed "the decision was indeed taken from jointly agreement by project partners at several contractual levels" [1].
The failure caused by a series of issues but exception from unprotected variables can be considered as the starting point of the incident.
The "assumption failure" [4] led to problems in other development processes as well. For example, no test was performed to verify that the SRI would behave correctly when being

subjected to Ariane 5 characteristics such as flight time sequence or the trajectory because it was not mentioned in the specification.
Knowing the cause of the failure, it could have been avoided with reasonable cost if

- Variables at risk of exception should have been protected and associated with appropriate error handler. Terminating the execution of a program upon detection of exception or error is not acceptable form of error handling. In the case of SRIs, even though there was software exception, SRIs should continue providing its best estimates of the required information to the OBC.
- The critical variable values should have been verified before performing any physical action. The system should specify operational restrictions from chosen implementation and handle properly excessive, non-compliant data that might appear. These identification and handler of such limitation were considered mandatory for every mission-critical device like SRI in Ariane 5.
- Significant aspect of the system operating environment should have been taken into consideration for engineering systems (i.e the discrepancies in Ariane 4 and 5 characteristics)
- Extensive testing had been prepared which can cover as many behavioral states of the system as feasible in order to detect and figure out failure scenarios upfront.
- Applied a small but useful practice to comment code that is based on a particular assumption or condition in order to facilitate reuse, maintenance and checking/verification.

3.4 Reuse Error

The fatal unhandled exception occurred when converting data stayed in a module that was not required by Ariane 5. If this code had not been included in the project the fatal error would more than likely not have proved to be an issue [8].
SRI horizontal bias module was reused from 10 year-old software since Ariane 4. The main reason that made the re-used component failed was the absence of precise specification associated with that component. As the physical characteristics of Ariane 5 differ from that of Ariane 4, the reused module were not able adjust to cope with such changes.

4 Behavioral Analysis

4.1 Non-purposeful behavior

Trajectory data were not used to analyze the behavior of the unprotected variables, and it is even more important to note that it was jointly agreed not to include the Ariane 5 trajectory data in the SRI requirements and specification [1] => no suitable behavior was specified when the exception was thrown.
The alignment program was unnecessarily running after lift-off and "the exception condition" (BH overflow) was raised in this program. It was later acknowledged that different from

Ariane 4, there is no need to keep this program running after lift-off in the case of Ariane 5 => no reference to a specific goal of the system.

4.2 NON-TELEOLOGICAL BEHAVIORS

The behavior has strong dependence on prescribed environmental conditions. However, as it was working well with Ariane 4, no change took into consideration for Ariane 5 which had different characteristics. Therefore, new unanticipated environmental conditions violated the assumption.
The system was provided with no feed-back and exceptional behavior (extremely high BH value) was not identified in order to have reasonable fault-tolerance mechanism. While this approach enhances efficiency and results in a lean and cost-effective design, the system could not adjust dynamically in order to cope with software failure or significant changes[2].

4.3 INAPPROPRIATE BEHAVIOR

Inappropriate behavior also contributed to the failure. In the event of any kind of exception in SRI, the failure was indicated on the databus, stored in EEPROM memory and eventually the processor was to be stopped. Such behaviour on encountering an uncaught exception was predicated on the assumption that the error handling was to meant to "mitigate random hardware failure" while Backup software was a copy and behaved in exactly the same way as in primary software. This drastic action lies in the culture within the Ariane programme of only addressing random hardware failures [1].
That poor behavior had catastrofic effects on the whole behavior of the system and caused the failure. Recommendation from Inquiry Board suggested that the SRIs could have continued to provide their best estimates of the required altitude information.

5 CONCLUSION

The report discussed the failure of Ariane 5 from 02 different views, engineering and behavioral analysis process respectively.
On the basis of the analysis, a number of useful lessons could be learnt to prevent other such disaster. If it is of reader's interest, detail recommendations from Inquiry Board could be found in [1].

REFERENCES

[1] Inquiry Board, *European Space Agency, Ariane 5, Flight 501 Failure*. Report, 19 July 1996, available at http://www.esrin.esa.it/htdocs/tidc/Press/Press96/ariane5rep.html

[2] Vincenzo De Florio, *Antifragility = Elasticity + Resilience + Machine Learning: Models and Algorithms for Open System Fidelity*. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)

[3] Gerard Le Lann, *An Analysis of the Ariane 5 Flight 501 Failure- A System Engineering Perspective*. ECBS'97 Proceedings of the 1997 international conference on Engineering of computer-based systems

[4] Vincenzo De Florio, *Software Assumptions Failure Tolerance: Role, Strategies, and Visions*. Lecture Notes in Computer Science Volume 6420, 2010, pp 249-272

[5] Mordechai Ben-Ari, *The Bug That Destroyed a Rocket*. Journal of Computer Science Education, vol. 13 no. 2, pp. 15-16, 1999

[6] Mark Dowson, *The ARIANE 5 Software Failure*. Software Engineering Notes vol 22 no 2, 1997

[7] Ian Sommerville, *The Ariane 5 Launcher Failure*. http://www.slideshare.net/sommerville-videos/ariane-5-launcher-failure-30036896

[8] Michael Roper, *Disastrous Software Projects: The Failure of Ariane 5 Flight 501*. http://sydney.edu.au/engineering/it/ scilect/se/2000/task3/essays/mproper.txt