Example of Disaster caused by Overflow Error

For a video see: http://www.youtube.com/watch?v=z-r9cYp3tTE

On June 4, 1996 the French Ariane 5 rocket exploded on its first research mission. The cost of the rocket and its scientific cargo was approximately $500 million. The reason for the explosion was a computer arithmetic error -- an overflow error. [Pictures from http://www.ima.umn.edu/~arnold/disasters/ariane.html ]

A SIAM News article is below the most relevant parts are in red.

From SIAM News, Vol. 29. Number 8, October 1996:

Inquiry Board Traces Ariane 5 Failure to Overflow Error

Readers of SIAM News may remember that on June 4, less than a minute into its first flight, the French rocket Ariane 5 self-destructed. As it happened, the board appointed by CNES (Centre national des études spatiales) and ESA (the European Space Agency) to investigate the failure was chaired by applied mathematician Jacques-Louis Lions of the Collège de France. The story of the uncovering of the software error that led to the crash is summarized here, based on an English translation of parts of the board’s report, which was completed within six weeks of the explosion.

On the basis of the documentation and information available to the board, the weather at the launch site in Kourou, French Guiana, on the morning of June 4, 1996, was acceptable for a launch. In particular, there was no risk of lightning since the strength of the electric field measured at the launch site was negligible. The only uncertainty was whether visibility criteria would be fulfilled.

The countdown went smoothly until 7 minutes before the scheduled liftoff time, when the launch was put on hold because the visibility criteria were not met. Visibility conditions improved as forecast, however, and the launch was initiated. Ignition of the Vulcain engine and the two solid
boosters was nominal, as was liftoff. The flight of the vehicle was nominal until approximately 37 seconds after liftoff. Shortly after that time, the vehicle suddenly veered off its flight path, broke up, and exploded. A preliminary investigation of flight data showed:

- nominal behavior of the launcher up to 36 seconds after liftoff;
- failure of the back-up inertial reference system, followed immediately by failure of the active inertial reference system;
- swivelling into the extreme position of the nozzles of the two solid boosters and, slightly later, of the Vulcain engine, causing the launcher to veer abruptly; and
- self-destruction of the launcher, correctly triggered by rupture of the links between the solid boosters and the core stage.

The origin of the failure was thus rapidly narrowed down to the flight control system, and more particularly to the two inertial reference systems (SRIs), which obviously ceased to function almost simultaneously at about 36.7 seconds.

**Information Available**

The information available to the board included telemetry data received on the ground until 42 seconds after liftoff, trajectory data from radar stations, optical observations (infrared camera, films), and results of the inspection of recovered material. The telemetry data received in Kourou were transferred to CNES/Toulouse, where they were converted into parameter over time plots.

The self-destruction of the launcher occurred near the launch pad, at an altitude of approximately 4000 meters. Recovery of the debris, which was scattered over an area of approximately 12 square kilometers east of the launch pad, proved difficult since this area is nearly all mangrove swamp or savanna.

Nevertheless, it was possible to retrieve the two SRIs from the debris. Of particular interest was the one that had worked in active mode and stopped functioning last, and for which certain information was therefore not available in the telemetry data (provision for transmission of this information to the ground was confined to whichever of the two units might fail first). The results of the examination of this unit were very helpful in the analysis of the failure sequence.

**Analysis of the Failure**

In general terms, the flight control system of Ariane 5 is of a standard design. The attitude of the launcher and its movements in space are measured by an SRI. It has its own internal computer, in which angles and velocities are calculated on the basis of information from a “strap-down” inertial platform, with laser gyros and accelerometers. The data from the SRI are transmitted through the databus to the on-board computer (OBC), which executes the flight program and controls the nozzles of the solid boosters and the cryogenic Vulcain engine, via servovalves and hydraulic actuators.

For improved reliability, there is considerable redundancy at the equipment level. Two SRIs operate in parallel, with identical hardware and software. One SRI is active, and one is in “hot”
stand-by; if the OBC detects that the active SRI has failed, it immediately switches to the other one, provided that this unit is functioning properly. Likewise, there are two OBCs, and a number of other units in the flight control system are duplicated as well.

The design of the SRI used in Ariane 5 is almost identical to that of Ariane 4, particularly with regard to the software. Based on the extensive documentation and data made available to the board, the following chain of events was established, starting with the destruction of the launcher and tracing back in time toward the primary cause:

The launcher began to disintegrate at about 39 seconds because of high aerodynamic loads resulting from an angle of attack of more than 20 degrees, which led to separation of the boosters from the main stage, which in turn triggered the self-destruct system of the launcher.

This angle of attack was caused by full nozzle deflections of the solid boosters and the main Vulcain engine.

The nozzle deflections were commanded by the OBC software on the basis of data transmitted by the active SRI (SRI 2). Part of the data for that time did not consist of proper flight data, but rather showed a diagnostic bit pattern of the computer of SRI 2, which was interpreted as flight data.

SRI 2 did not send correct attitude data because the unit had declared a failure due to a software exception.

The OBC could not switch to the back-up SRI (SRI 1) because that unit had already ceased to function during the previous data cycle (72-millisecond period) for the same reason as the SRI 2.

The internal SRI software exception was caused during execution of a data conversion from a 64-bit floating-point number to a 16-bit signed integer value. The value of the floating-point number was greater than what could be represented by a 16-bit signed integer. The result was an operand error. The data conversion instructions (in Ada code) were not protected from causing operand errors, although other conversions of comparable variables in the same place in the code were protected.

The error occurred in a part of the software that controls only the alignment of the strap-down inertial platform. The results computed by this software module are meaningful only before liftoff. After liftoff, this function serves no purpose. The alignment function is operative for 50 seconds after initiation of the flight mode of the SRIs (3 seconds before liftoff for Ariane 5). Consequently, when liftoff occurs, the function continues for approximately 40 seconds of flight. This time sequence is based on a requirement of Ariane 4 that is not shared by Ariane 5.

The operand error occurred because of an unexpected high value of an internal alignment function result, called BH (horizontal bias), which is related to the horizontal velocity sensed by the platform. This value is calculated as an indicator for alignment precision over time. The value of BH was much higher than expected because the early part of the trajectory of Ariane 5 differs from that of Ariane 4 and results in considerably higher horizontal velocity values.
The internal SRI events that led to the failure have been reproduced by simulation calculations. Furthermore, both SRIs were recovered during the board’s investigation, and the failure context was determined precisely from memory readouts. In addition, the board examined the software code, which was shown to be consistent with the failure scenario.

The board feels, therefore, that it is established beyond reasonable doubt that the chain of events set out above reflects the technical causes of the failure.

Comments on the Failure Scenario

In the failure scenario, the primary technical causes are the operand error that occurred in the conversion of the horizontal bias variable BH, and the lack of protection of this conversion, which caused the SRI computer to stop.

The board was told that not all the conversions were protected because a maximum workload target of 80% had been set for the SRI computer. To determine the vulnerability of unprotected code, an analysis was performed on every operation that could give rise to an exception, including an operand error. In particular, the conversion of floating-point values to integers was analyzed; operations involving seven variables were at risk of leading to operand errors. 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 direct reference to justification for this decision was found in the source code. Given the large amount of documentation associated with any industrial application, the assumption, although agreed upon, was essentially obscured, although not deliberately, from any external review.

The three remaining variables, including the one denoting horizontal bias, were unprotected because further reasoning indicated either that they were physically limited or that there was a large margin of safety---reasoning that in the case of the variable BH turned out to be faulty.

There is no evidence that any trajectory data were 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 specifications. Although the source of the operand error has been identified, this in itself did not cause the mission to fail. The specification of the exception-handling mechanism also contributed to the failure. In the event of any kind of exception, according to the system specification, the failure should be indicated on the databus, the failure context should be stored in an EEPROM memory (which was recovered and read out for Ariane 5), and, finally, the SRI processor should be shut down.

It was the decision to cease the processor operation that finally proved fatal. Restart is not feasible since attitude is too difficult to recalculate after a processor shutdown; therefore, the SRI becomes useless. The reason behind this drastic action lies in the custom within the Ariane program of addressing only random hardware failures. From this point of view, exception- or error-handling mechanisms are designed for random hardware failures, which can quite rationally be handled by a backup system.
Although the failure resulted from a systematic software design error, mechanisms can be introduced to mitigate this type of problem. For example, the computers within the SRIs could have continued to provide their best estimates of the required attitude information. There is reason for concern that a software exception should be allowed, or even required, to cause a processor to halt while handling mission-critical equipment. Indeed, the loss of a proper software function is hazardous because the same software runs in both SRI units. In the case of Ariane 5, this resulted in the switching off of two still healthy critical units of equipment.

The original requirement accounting for the continued operation of the alignment software after liftoff was brought forward more than 10 years ago for the earlier models of Ariane, in order to cope with the rather unlikely event of a hold in the countdown, e.g., between –9 seconds, when flight mode starts in the SRI of Ariane 4, and –5 seconds, when the resetting of certain events initiated in the launcher would take several hours. The period selected for this continued alignment operation, 50 seconds after the start of flight mode, was based on the time needed for the ground equipment to resume full control of the launcher in the event of a hold. This special feature made it possible with the earlier versions of Ariane to restart the countdown without waiting for normal alignment, which takes 45 minutes or more, so that a short launch window could still be used. In fact, this feature was used once, in 1989. The same requirement does not apply to Ariane 5, which has a different preparation sequence, and it was maintained for commonality reasons, presumably based on the view that, unless proven necessary, it was not wise to make changes in software that worked well on Ariane 4.

Even in the cases in which the requirement is still found to be valid, it is questionable for the alignment function to be operating after liftoff. Alignment of mechanical and laser strap-down platforms involves the use of complex mathematical filter functions to properly align the x-axis to the gravity axis and to find north from sensing of the earth’s rotation. The assumption of preflight alignment is that the launcher is positioned at a known and fixed position. Therefore, the alignment function is totally disrupted when performed during flight: The measured movements of the launcher are interpreted as sensor offsets and other coefficients characterizing sensor behavior.

Returning to the software error, software is an expression of a highly detailed design and does not fail in the same sense as a mechanical system. Furthermore, software is flexible and expressive and thus encourages highly demanding requirements, which in turn lead to complex implementations that are difficult to assess.

An underlying theme in the development of Ariane 5 is a bias toward the mitigation of random failure. The supplier of the SRI followed the specifications given to it, which stipulated that in the event of any detected exception the processor was to be stopped. The exception that occurred was due not to random failure but to a design error. The exception was detected but was handled inappropriately because the view had been taken that software should be considered correct until it is shown to be at fault. The board has reason to believe that this view is also accepted in other areas of Ariane 5 software design. The board favors the opposite view—that software should be assumed to be faulty until application of the currently accepted best practice methods can demonstrate that it is correct.
This means that critical software---in the sense that failure of the software puts the mission at risk---must be identified at a very detailed level, that exceptional behavior must be confined, and that a reasonable back-up policy must take software failures into account.