**Running head: Software Testing 1**

**Software Testing: Lessons from Ariane 5**

**Anonymous UoPeople Student**

**March 4, 2015**

**Running head: Software Testing 2**

**Software Testing: Lessons from Ariane 5**

On June 4, 1996, demonstration flight 501 of the Ariane 5 rocket launch system ended in disaster. According to the board of inquiry assigned by the Director General of ESA and the Chairman of CNES, the launch appeared normal until about 40 seconds into the flight, when the rocket veered off-course and exploded (Ladkin, 1998). The subsequent investigation by the inquiry board suggests that the disaster happened because although the system worked for the Ariane 4 launches, it was not thoroughly tested for Ariane 5. The failed launch of flight 501 can be used as a case study to learn more about the need for proper system testing.

**The Cause of Failure**

The Ariane 5 system was repurposed from that of the Ariane 4 launchers. According to the board of inquiry’s investigation, one seemingly small difference between the two launchers is that the Ariane 5 follows a slightly different trajectory in early flight stages (Lions, 1996). The change in launch trajectory produced higher horizontal bias and higher horizontal velocity values than the system expected and could handle (Lions, 1996). Data was too large to be converted from a 64-bit floating point value to a 16-bit signed integer value, and so an exception happened. This exception caused both the back-up and active Inertial Reference Systems to fail in the module that controls the alignment of the strap-down inertial platform. Ironically, this module serves no purpose during flight and could theoretically be ignored. Unfortunately, though, the exception was not ignored, leading the active SRI system to send a diagnostic bit pattern along with the flight data to the On-Board Computer. The On-Board Computer then incorrectly interpreted the diagnostic data as part of the altitude data, which led to a sharp change in trajectory. This extreme change in trajectory caused the rocket to break apart and trigger the self-destruct system (Lions, 1996).

**Could the Failure be Avoided?**

There were system tests which could have been performed which would have exposed the problems within the system. In fact, an analysis to determine if exceptions could arise from calculations, including Operand Errors, was indeed performed (Lions, 1996). The testing found 7 variables that could have an Operand Error, but only 4 of these variables were protected from error. Protection of the other three variables was not considered necessary as either the range of data was “physically limited or that there was a large margin of safety” (Lions, 1996) This decision not to protect these variables, one of which was the horizontal, was not taken lightly. In fact, the decision was made “jointly by project partners at several contractual levels” (Lions, 1996).

**More Testing was Needed**

The fact that further testing wasn’t done to ensure the system would not fail when contributors knew that Operand Errors were a possibility is regretful. The board of inquiry even suggests that a simulation of flight data would have revealed the points for system failure (Lions, 1996). Below is a sample test case for the simulations that might have been performed and prevented this disaster.

**Running head: Software Testing 2**

|  |  |
| --- | --- |
| **Project Name: Ariane 5** | |
| **Test Case for Simulations** | |
| **Test Case ID:** Flight 501 | **Test Designed by:** <Lions> |
| **Test Priority (Low/Medium/High):** High | **Test Designed date:** <1996> |
| **Module Name:** Strap-Down Inertial Platform Module | **Test Executed by:** <Lions> |
| **Test Title:** Flight Data Simulation | **Test Execution date:** <1996> |
| **Description:** Use simulation data to verify exception handling |  |
|  |  |
|  |  |
| **Pre-conditions:** All modules function up to this module. | |
| **Dependencies:** How will exceptions in this module affect other modules. | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Simulate flight data which will produce exceptions | Simulate large horizontal bias values | Module should send exception to on-board computer and it should be handled correctly. | Exception handled | Pass |  |
| 2 | Simulate Operand Error | Simulate floating point values that need to be converted to 16-bit signed values | Module should send exception to on-board computer and it should be handled correctly | Exception not handled | Fail |  |
|  |  |  |  |  |  |  |

**Post-conditions:**

Large flight data values that result in exceptions need to be handled with a diagnostic bit pattern. The diagnostic bit pattern needs to be recognized by the on-board computer and properly handled.

References

Ladkin, P. B. (1998). “The Ariane 5 Accident: A Programming Problem?” Article RVS-J-98-02. Retrieved from <http://www.cs.st-andrews.ac.uk/~ifs/Books/SE9/CaseStudies/Ariane5/SupportingDocs/Ladkin-Ariane5.html>

Lions, J. L. (1996). “ARIANE 5: Flight 501 Failure. Report by the Inquiry Board.” Retrieved from <http://www.di.unito.it/~damiani/ariane5rep.html>