Questions

Bookmark this page

testing basics

1/1 point (graded)

In the 1990s, the Ariane 5 launch vehicle, designed and built for the European Space Agency, self-destructed 37 seconds after its first launch.

The reason was a control software bug that went undetected. The Ariane 5's guidance software was reused from the Ariane 4, which was a slower rocket. When the velocity calculation converted from a 64-bit floating point number (a `double` in Java terminology, though this software wasn't written in Java) to a 16-bit signed integer (a `short`), it overflowed the small integer and caused an exception to be thrown. The exception handler had been disabled for efficiency reasons, so the guidance software crashed. Without guidance, the rocket crashed too. The cost of the failure was $1 billion.

What ideas does this story demonstrate? Check all that apply.

- [x] Even high-quality safety-critical software may still have residual bugs.
- [ ] Testing all possible inputs is the best solution to this problem.
- [x] Software exhibits discontinuous behavior, unlike many physically-engineered systems.
- [ ] Static type checking could have detected this bug.

✔

Explanation
Testing all inputs is not feasible, because even just one 64-bit floating point number has 2^64 possible values, which is more than the age of the universe in microseconds.
Static type checking wouldn't have detected this bug, because the code intentionally converted a 64-bit `double` into a 16-bit `short`.

Submit Show Answer

Answers are displayed within the problem