Checklist for Safety Critical Systems

Developed by Jaffe, Leveson, Heimdahl, Melhart
Translated to a plain English by Lutz

The guidelines in this checklist are primarily intended for safety-critical control systems. They are, however, suitable for all types of systems and point out a collection of areas that typically do not receive adequate attention.

1. Is the software's response to out-of-range values specified for every input?

2. Is the software's response to not receiving an expected input specified? (That is, are timeouts provided?) Does the software specify the length of the timeout, when to start counting the timeout and the latency of the timeout (the point past which the receipt of new inputs cannot change the output result, even if they arrive before the actual output)?

3. If input arrives when it shouldn't, is a response specified?

4. On a given input, will the software always follow the same path through the code (that is, is the software's behavior deterministic)?

5. Is each input bounded in time? That is, does the specification include the earliest time at which the input will be accepted and the latest time at which the data will be considered valid (to avoid making control decisions based on obsolete data)?

6. Is a minimum and maximum arrival rate specified for each input (for example, a capacity limit on interrupts signaling an input)? For each communication path, are checks performed in the software to avoid signal saturation?

7. If interrupts are masked or disabled, can events be lost?

8. Can any output be produced faster than it can be used (absorbed) by the interfacing module? Is overload behavior specified?

9. Is all data output to the buses from the sensors used by the software? If not, it is likely that some required function has been omitted from the specification.

10. Can input that is received before startup, while offline, or after shutdown influence the software's startup behavior? For example, are the values of any counters, timers, or signals retained in software or hardware during shutdown? If so, is the earliest or most-recent value retained?

11. In cases where performance degradation is the chosen error response, is the degradation predictable (for example, lower accuracy, longer response time)?

12. Are there sufficient delays incorporated into the error-recovery responses, e.g., to avoid returning to the normal state too quickly?

13. Are feedback loops (including echoes) specified, where appropriate, to compare the actual effects of outputs on the system with the predicted effects?

14. Are all modes and modules of the specified software reachable (used in some path through the code)? If not, the specification may include superfluous items.

15. If a hazards analysis has been done, does every path from a hazardous state (a failure-mode) lead to a low-risk state?

16. Are the inputs identified which, if not received (for example, due to sensor failure), can lead to a hazardous state or can prevent recovery (single point failures)?