1- The Denver Airport's problems are common, as the paper quotes: "2 of every 8 large-scale systems are canceled and 75% of those that run are failures"

2- According to the reading, about 75% of large systems are “operating failures”. The average project overshoots schedule by half; larger projects generally do worse

3- The software Crises refers to widespread issues with building large-scale software with allocated budget and time constraints. It was recognized by NATO in an effort to provide global resources to the issue, but the problem still arises today.

4- The paper states, "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software." This definition was established as a representation of an engineered, repeatable, and measurable process.

5- Outsourcing absolutely affects software standards & interoperability. Outsourcing to countries such as Russia, India, and the Philippines for low labor costs often also lack management expertise, quality of work, and raise risks of project quality.

6- No, errors are hard to eliminate even with in depth review. For example, the DOD applied strict standards to the Clementine satellite, and still a software bug caused the satellite to waste resources and fail its mission.

7-Realtime systems must respond instantly to external events, and errors often appear only under specific conditions. Failures in these systems, like spacecraft or nuclear reactors, can be catastrophic.

Distributed systems involve many interconnected computers. They are fragile because they create “a great set of interconnected single points of failure” that are hard to anticipate. Projects usually overrun budgets, and schedules.

8-The Loral team received a top rating on the Capability Maturity Model and could predict bugs with high accuracy, but shuttle missions still experienced software failures. The 1981 launch was delayed by a synchronization glitch, and the 1992 Intelsat-6 rescue mission was jeopardized by a rendezvous program bug - demonstrating that even the best teams cannot guarantee flawless results in their software.

9-Formal Methods and mathematics help but don't solve everything. The paper explains that formal methods catch design flaws early, but they aren't sufficient enough because they can only prove a system meets its specification, not that it will handle unexpected real-world conditions.

10-We're interested in component software because reusable, interchangeable software parts could transform programming from a handcrafted process into an industrial one. The article compares it to interchangeable musket parts, where instead of rewriting code from scratch, developers could assemble systems from built components. This would reduce errors, costs, and time.

I pledge that I have abided by the stevens honor system