Homework 00c

Name: Vanshaj Tyagi

Course: SSW-567-A

Question:

From your reading of the "Software's Chronic Crisis" paper, try to answer the following questions. Please submit your answers in a pdf and highlight the question number for each answer.

Be precise and avoid long answers. Support your answers by identifying the page/paragraph number for excerpts in the Software Crisis paper that you used to build your answer.

1- Are the problems of Denver Airport common?

2- What are the percentages of ‘operating failures’ Software as observed in the mid-1990s? What are the average increase in scheduling time?

3- What are the Software Crises?

4- What is ‘Software Engineering’? (answer from the paper)

5- Do you feel that outsourcing affects software standards and interoperability? Why or why not?

6- Is it easy to get the software right the first time? Maybe if we try harder and with military discipline?

7- What is unique about ‘Realtime Systems’? And what is different in ‘Distributed Systems’?

8- Did the Loral team experience go completely successful?

9- Would formal methods or mathematics solve the problem?

10- Why are we interested in component software? (from the paper)

Answer:

1. Yes, according to the paper, the problems seen at the Denver Airport are common, but their visibility is what makes them notable. The author cites studies showing that for every six large-scale software systems put into operation, two others are canceled, and three-quarters of all large systems are "operating failures" (paragraph 3)

2. The paper states that in the mid-1990s, three-quarters (75%) of all large systems were "operating failures" (page 86, paragraph 3). The average software development project overshoots its schedule by half (50%) (paragraph 3)

3. The "software crisis" refers to the difficulties of building large, complex software systems. It's a term coined in the autumn of 1968 by the NATO Science Committee to describe the industry's inability to meet the demands of an information-age society with mature engineering practices (paragraph 5).

4. The paper defines "software engineering" formally as "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software" (paragraph 5).

5. Yes, because practices differ across teams and countries. The paper notes U.S. companies shifting work to India, Russia, etc., where design and management skills may be weaker, leading to inconsistencies (paragraph 9 line 5 from A developing world)

6. No, the paper suggests it is not easy to get software right the first time, even with rigorous methods. It highlights that the Department of Defense's rigorous testing standards failed to prevent a bug in the Clementine satellite's software from causing a mission failure (paragraph 11)

7. Real-time systems are unique because errors are "devilishly difficult to spot" and often occur only when conditions are specific, like a car engine making a suspicious sound (paragraph 5). Distributed systems are different because they pose a major challenge due to their "complexity and fragility," and they often consist of a "great set of interconnected single points of failure, many of which you have not identified beforehand" (paragraph 13).

8. Not completely. While they managed bugs predictably and reached CMM level 5, some failures still occurred, such as shuttle mission delays in 1981 and 1992 due to software glitches (paragraph 4 from Mayday, Mayday)

9. Not entirely. Formal methods can detect early design errors and improve safety, but they are difficult to use, can only guarantee meeting specifications (not real-world surprises), and remain limited. (paragraph 47)

10. Because reusable, interchangeable parts could reduce costs and improve productivity, like industrial revolutions in other fields. They allow specialization and avoid rewriting code repeatedly. We are interested in component software to create an industry based on "interchangeable, reusable software parts." This is seen to move beyond the "preindustrial" approach of building everything from the ground up. (paragraph 6)