ECE 322

SOFTWARE TESTING AND MAINTENANCE

Fall 2025

Assignment #1
SOLUTIONS

Due date: Monday, September 15, 2025 by 3:00 PM

Total: 50 points

Please be concise.

Problems 1 and 2 could have a variety of correct solutions; it is essential that there are some convincing supporting arguments provided.

Value 20 points

1. Study two papers entitled (located in the Readings folder):

- *No silver bullet* published in *IEEE Computer*
- *The software engineering silver bullet conundrum* published in *IEEE Software*

Based on this material, what are, in your opinion, the two most essential factors making software testing activities difficult? Distinguish between technical and non-technical (say, organizational, human, etc.) factors. Justify your opinion; provide some convincing arguments. Are these factors associated with inherent or accidental difficulties?

Value 20 points

2. Discuss an example of a serious software failure in 2024-2025. For this, complete some Web search and literature review. Clearly identify a source of your information (e.g., include a link to the website). Describe the nature of the software failure. Identify its origin. Were there any software testing efforts mentioned in relation to the resulting failure? Was there any follow-up action taken? Was there any plan to alleviate further problems?

Be critical in your assessment. Sometimes the quality of the available source of information could be questionable (which is a typical downfall of many Web resources). Use at least two different sources; they might offer various perspectives on the same problem. You may wish to organize your findings in a tabular format. Offer the most essential info; be concise.

In your writing, use the following template identifying failure description, nature of software failure, testing efforts regarding failure, follow-up action, and URL where the material was found.

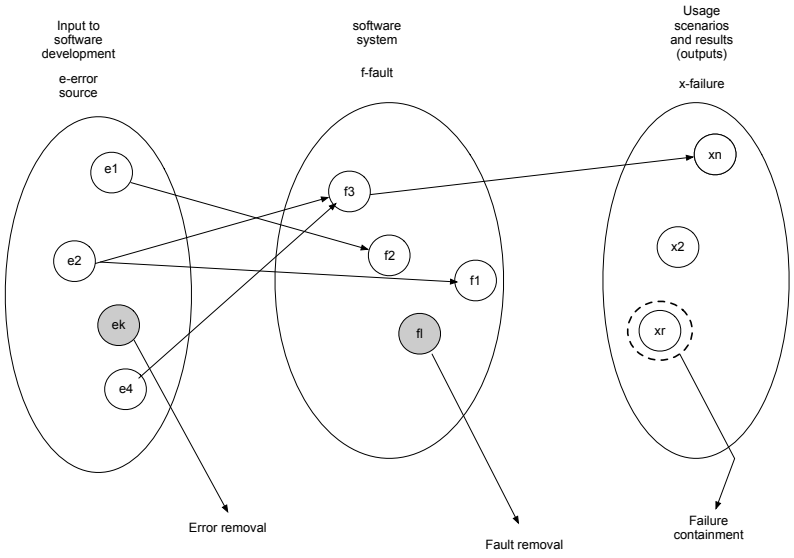
This is an example	
Failure description	**Therac-25 Accidents** Eleven Therac-25s, radiation therapy machines, were installed: five in the US and six in Canada. Six accidents involving massive overdoses to patients occurred between 1985 and 1987. The accidents occurred when the high-energy electron-beam was activated without the target having been rotated into place; the machine's software did not detect that this had occurred, and did not therefore determine that the patient was receiving a potentially lethal dose of radiation, or prevent this from occurring. The very high energy electron-beam directly struck the patients causing the feeling of an intense electric shock and the occurrence of thermal and radiation burns. In some cases, the injured patients died later from radiation poisoning.
Nature of software failure	Several features of the Therac-25 are important in understanding the accidents. Some of essential causes were: (1) The engineers had reused software from older models. These models had hardware interlocks that masked their software defects. Those hardware safety mechanisms had no way of reporting that they had been triggered, to at least indicate the existence of faulty software commands (2) The hardware provided no way for the software to verify that sensors were working correctly (3) The software was written in assembly language. While this was more common at the time than it is today, assembly language is harder to debug than most high-level languages.
Any testing efforts regarding the failure?	(1) Related problems were found in the Therac-20 software. These were not recognized until after the Therac-25 accidents because the Therac-20 included hardware safety interlocks and thus no injuries resulted. (2) After the 2nd incident the Atomic Energy of Canada Limited (AECL) sent a service technician to the Therac-25 machine. He was unable to recreate the malfunction and therefore concluded that nothing was wrong with the software. Some minor adjustments to the hardware were made.
Any follow up action taken? Any plan to alleviate further problems?	The machine was recalled in 1987 and the AECL made a variety of changes in the software of the Therac-25 radiation treatment system. The machine itself is still in use today.
URL	(1) http://en.wikipedia.org/wiki/Therac-25 (2) The Therac-25 Accidents (PDF): http://sunnyday.mit.edu/papers/therac.pdf (3) An Investigation of the Therac-25 Accidents (IEEE Computer) http://courses.cs.vt.edu/~cs3604/lib/Therac_25/Therac_1.html

Value 10 points

3. Consider a way of moving from faults to failures, as displayed below. Give an example of when a single error can give rise to several faults and a failure.

An example is when a programmer does not notice the size of a data type and stores a 5-bit number in an integer that only can store numbers up to 4 bits. This can lead to using the wrong number in the computations and produce wrong answers. Finally, faults and failure can be the consequences of this error.

You have to provide another example and elaborate more on how this error can cause faults and failures.



Solution. There are numerous sequences: error → fault → failure.

This sequence concerns a situation in the software development and involves software developers. As noted, an error is a mistake made by a developer who might not be well familiar with some aspects of coding, say, a complete knowledge of a proper initialization of variables or arrays, memory management (pointers) and alike. This could lead to an error. The particular lack of knowledge may result in a faulty construct in the software during coding (viz. a piece of code where some arrays are left uninitialized).
In this way a fault becomes a direct consequence of the error. Errors could be avoided (eliminated) through a reinforcement of a pertinent training. Once the fault has been input into system, it could be found by testing. Of course, not all errors are going to lead to faults; the same as not all faults are going to end up in a software failure.
For instance, in the design of some software unit, one overlooks to initialize matrices and variables. In the coding process, the variables/arrays have not been properly initialized. This is a software error. This single error might lead to a number of software faults

depending whether the compiler does this initialization automatically. For some environment (use of the compiler), the fault will not result in software failure.

Another example: arrays are indexed from 0; a[0], a[1]… b[0] b[1]…etc. the fault comes from the fact that the arrays were initialized starting from a[1], b[1]… This fault could result in a failure if one uses a[0] b[0]… in consecutive computing. Again, the error could results in a number of faults whenever matrices are used in the system.