Chapter 3 & 4 Questions

Dr. Paul West

Department of Computer Science
College of Charleston

February 27, 2014

Question 1

- Which principles guided the choices?
  1. Use externally readable format also for internal files, when possible:
  2. Collect & analyze data about faults revealed and removed from the code:
  3. Separate testing and debugging into two phases:
  4. Distinguish test case design from execution:
  5. Produce complete fault reports:
  6. Use information from test case design to improve requirements & design specs:
  7. Provide interfaces for fully inspecting the internal state of a class:

Question 1

- Which principles guided the choices?
  1. Use externally readable format also for internal files, when possible: visibility & redundancy
  2. Collect & analyze data about faults revealed and removed from the code:
  3. Separate testing and debugging into two phases:
  4. Distinguish test case design from execution:
  5. Produce complete fault reports:
  6. Use information from test case design to improve requirements & design specs:
  7. Provide interfaces for fully inspecting the internal state of a class:

Question 1

- Which principles guided the choices?
  1. Use externally readable format also for internal files, when possible: visibility & redundancy
  2. Collect & analyze data about faults revealed and removed from the code: feedback
  3. Separate testing and debugging into two phases:
  4. Distinguish test case design from execution:
  5. Produce complete fault reports:
  6. Use information from test case design to improve requirements & design specs:
  7. Provide interfaces for fully inspecting the internal state of a class:

Question 1

- Which principles guided the choices?
    1. Use externally readable format also for internal files, when possible: visibility & redundancy
    2. Collect & analyze data about faults revealed and removed from the code: feedback
    3. Separate testing and debugging into two phases: partition
    4. Distinguish test case design from execution:
    5. Produce complete fault reports:
    6. Use information from test case design to improve requirements & design specs:
    7. Provide interfaces for fully inspecting the internal state of a class:

Question 1

- Which principles guided the choices?
  1. Use externally readable format also for internal files, when possible: visibility & redundancy
  2. Collect & analyze data about faults revealed and removed from the code: feedback
  3. Separate testing and debugging into two phases: partition
  4. Distinguish test case design from execution: partition & restriction
  5. Produce complete fault reports:
  6. Use information from test case design to improve requirements & design specs:
  7. Provide interfaces for fully inspecting the internal state of a class:

Question 1

- Which principles guided the choices?
  1. Use externally readable format also for internal files, when possible: visibility & redundancy
  2. Collect & analyze data about faults revealed and removed from the code: feedback
  3. Separate testing and debugging into two phases: partition
  4. Distinguish test case design from execution: partition & restriction
  5. Produce complete fault reports: visibility & feedback
  6. Use information from test case design to improve requirements & design specs:
  7. Provide interfaces for fully inspecting the internal state of a class:

Question 1

- Which principles guided the choices?
  1. Use externally readable format also for internal files, when possible: visibility & redundancy
  2. Collect & analyze data about faults revealed and removed from the code: feedback
  3. Separate testing and debugging into two phases: partition
  4. Distinguish test case design from execution: partition & restriction
  5. Produce complete fault reports: visibility & feedback
  6. Use information from test case design to improve requirements & design specs: feedback
  7. Provide interfaces for fully inspecting the internal state of a class:

Question 1

- Which principles guided the choices?
  1. Use externally readable format also for internal files, when possible: visibility & redundancy
  2. Collect & analyze data about faults revealed and removed from the code: feedback
  3. Separate testing and debugging into two phases: partition
  4. Distinguish test case design from execution: partition & restriction
  5. Produce complete fault reports: visibility & feedback
  6. Use information from test case design to improve requirements & design specs: feedback
  7. Provide interfaces for fully inspecting the internal state of a class: redundancy/visibility

Question 2

- A simple mechanism for augmenting fault tolerance consists of replicating computation and comparing the obtained results. Can we consider redundancy for fault tolerance an application of the redundancy principle?

Question 2

- A simple mechanism for augmenting fault tolerance consists of replicating computation and comparing the obtained results. Can we consider redundancy for fault tolerance an application of the redundancy principle?
- Yes, if the computations rely non-deterministic piece of hardware (IE: network)
- No, if the computations do not.

Question 3

- A system safety spec describes prohibited behaviors for the system. Explain how these can be viewed as an implementation of the redundancy principle.

Question 3

- A system safety spec describes prohibited behaviors for the system. Explain how these can be viewed as an implementation of the redundancy principle.
- Explicit checks for prohibitive behavior.

Question 4

- Process visibility can be increased by extracting information about the progress of the process. Indicate some information that can be easily produced to increase process visibility.

Question 4

- Process visibility can be increased by extracting information about the progress of the process. Indicate some information that can be easily produced to increase process visibility.
- Number of Modules/Functions completed
- Requirements satisfied/unsatisfied
- Completion of Tests (assuming Test Driven Development)
- API Completion

Question 1

- Under what circumstance might an incorrect program be 100% reliable?

Question 1

- Under what circumstance might an incorrect program be 100% reliable?
- An incorrect feature is never used.

Question 3

- If I am downloading a very large file over a slow modem, do I care more about the availability of my internet service provider or its mean time between failures?

Question 3

- If I am downloading a very large file over a slow modem, do I care more about the availability of my internet service provider or its mean time between failures?
- MTBF, especially if an interruption means restarting.

Question 4

- Can a program be correct but not safe.

Question 4

- Can a program be correct but not safe.
- Yes, a specification may not explain what to do during a power loss.

Question 6

- Characterize a domain you are familiar with in the tears of schedule, total cost, and dependability.

Question 6

- Characterize a domain you are familiar with in the tears of schedule, total cost, and dependability.
- Text Editor (VI)
- Schedule: Needs to be shipped with all OSs.
- Cost: Free, we will rely on Open Source programmers.
- Dependability: Always operates (no seg faults), very fast response to key presses, and does not correct my data.

Question 7

- Consider responsiveness as a desirable property of an Internet chat program. The informal requirement is that messages typed by each member of a chat session appear instantaneously on the displays of other users. Refine this informal requirement into a concrete specification that can be verified. Is anything lost in the refinement?

Question 7

- Consider responsiveness as a desirable property of an Internet chat program. The informal requirement is that messages typed by each member of a chat session appear instantaneously on the displays of other users. Refine this informal requirement into a concrete specification that can be verified. Is anything lost in the refinement?
- Messages shall appear on another screen within 1 second.
- Lost: Speed, and the assumption that message appear at the same time.