CISC/CMPE 327 Software Quality Assurance

Queen's University, 2020-fall

Review for Mini-Exam #1

Likely topics on mini-exam

- From Lecture 1:
  - Popular Views vs. Professional Views
  - Quality parameters: **technical** and **user**
    - correctness, reliability, capability, ...
    - usability, user documentation, availability, ...
  - The 3 general principles of quality assurance
  - Formal methods, testing, inspection, metrics
  - Know what a safety-critical system is

Likely topics on mini-exam

- From Lecture 2: 4 fundamental process activities
  - If it helps, remember “SDEV”, like “Software DEVeloper” (but also remember that the order is actually S, D, V, E)
  - Think about **why** the order is S, D, V, E.
    Say you do remember “Development”.
    Development needs some kind of specification ⇒ “Specification; Development”.
    Who needs the software?
    The customer ⇒ Validation.
    Software changes over time ⇒ Evolution.
  - You should be able to **briefly** describe what happens in each stage, not just the 4 words in that order

Likely topics on mini-exam

- From Lecture 2: Process models
  - Waterfall model
    - Drawback: “Early freezing” at various stages.
      Example: Freezing requirements risks doing lots of work against the wrong requirements.
    - Could work well when requirements are very stable
  - Prototyping model
    - Do a “quick and dirty hack” prototype, poor quality in most respects—**but** it gives the customer something to try out, identifying issues with requirements.
    - Drawback: wastes work—the prototype gets thrown away

Likely topics on mini-exam

- From Lecture 3:
  - Spiral model:
    - Continuously document and evaluate potential **risks**
    - Needs an **experienced** team
  - Iterative development:
    - Develop a subset of the software based on a subset of the requirements
    - Develop bigger subsets
    - Architecture is chosen early & needs a small team

Likely topics on mini-exam

- From Lecture 3 (continued):
  - Object-oriented development process
    - Analyze “essential system”, define “essential classes”
    - Derive additional classes
    - Define interfaces
    - Complete design and implement classes
    - Code generation
    - Drawbacks: no testing until late, architectural inflexibility (maybe the essential classes aren’t complete, maybe there are issues with requirements)

Likely topics on mini-exam

- From Lecture 4:
  - Scrum
  - Roles? Sprint? Process?
  - Standup meeting?
  - Product Backlog? Sprint Backlog?

  - Assume that you are the scrum master, how would you organize a sprint cycle?

Likely topics on mini-exam

- From Lecture 4:
  - Defects Prevention Process
    - You should have a rough idea of the steps involved
    - DPP vs Postmortem
    - maturity vs certification
  - {CMM, SPR, Baldrige, ISO...} ← **THESE WILL NOT BE ON THE MINI-EXAM**

Likely topics on mini-exam

- From Lecture 5: XP
  - Know the XP practices, especially:
    - simplicity
    - test first, test all the time
    - pair programming
    - on-site customer
  - Know some drawbacks of XP
  - Would XP be suitable for safety-critical software?
    I don't think there's one correct answer,
    but arguing (either way) demonstrates knowledge