University of Illinois at Urbana-Champaign
Department of Computer Science

Questions From Previous Exams

CS 428 Software Engineering II

1 Process

[2] 1. According to the agile manifesto, what are four important aspects that actually make a process agile?

Solution:

Important aspects of an agile process are:

- emphasis on an iterative process,
- favoring communication with clients,
- emphasis on continuous integration and having a working software,
- emphasis on individuals and interaction,
- having an adaptive process.

[6] 2. List three advantages and three disadvantages of splitting a software development team into groups? (Midterm, Spring 2010)

Solution:

Advantages include:

1. Group structure can reflect on the architectural structure, improving parallelism
2. Specialization of groups (e.g., business logic vs. GUI)
3. Easier management for very large teams

Disadvantages include:

1. Knowledge loss
2. Bottlenecks
3. Coordination

[2] 3. What does XP advocate about splitting development teams into groups? (Midterm, Spring 2010)

Solution:

Do not split into fixed groups. Move people around: use pair programming to share knowledge. Have collective code ownership.

2 Planning Game

2 4. What is PERT? Give a one sentence description. (Midterm, Spring 2009)

Solution:

The *Program (or Project) Evaluation and Review Technique* (PERT), is a model for project management designed to analyze and represent the tasks involved in completing a given project.

2 5. As discussed in the readings from Wikipedia, PERT charts are a tool for planning and managing projects. Give one advantage and one disadvantage of using a PERT. (Midterm, Spring 2010)

Solution:

The Wikipedia entry for PERT (http://en.wikipedia.org/wiki/Program_Evaluation_and_Review_Technique) explicitly lists the following advantages and disadvantages.

Advantages:

- PERT chart explicitly defines and makes visible dependencies (precedence relationships) between the work breakdown structure (WBS) elements
- PERT facilitates identification of the critical path and makes this visible
- PERT facilitates identification of early start, late start, and slack for each activity
- PERT provides for potentially reduced project duration due to better understanding of dependencies leading to improved overlapping of activities and tasks where feasible.

Disadvantages

- There can be potentially hundreds or thousands of activities and individual dependency relationships
- The network charts tend to be large and unwieldy requiring several pages to print and requiring special size paper
- The lack of a time frame on most PERT charts makes it harder to show status although colors can help (e.g., specific color for completed nodes)
- When the PERT charts become unwieldy, they are no longer used to manage the project

5 6. The two biggest causes of project failure are 1) inadequate schedule and 2) not managing requirements. What does XP advocate to ensure an accurate schedule? What does it advocate to control changing requirements. (Spring 2009)

Solution:

XP ensures that the development teams remain on schedule by advocating:

- advance planning that involves coming up with user stories and iteration plans,
- having very short iterations,
- constant communication amongst the team members to remove any coordination bottlenecks,
- emphasis on the simplest solution. Functionality is added incrementally as and when required.

To control changing requirements, XP ensures the developers get constant feedback. Feedback is obtained from the customer by showing them the current version of the working software after each iteration (that are short). Feedback is obtained from the team through regular meetings and continuous communication. Also, unit tests provide feedback to the developer about the state of the system with respect to the requirements/specifications, after changes in the system.

[3] 7. Describe how the cost of software change relates to the phase of the project. Then explain two advantages of a highly iterative process from this point of view. (Spring 2013)

Solution:

The cost of defects grows exponentially from requirements to release. The costs of changing requirements grows exponentially in the same manner.
By using a frequent release process, this requirement-design-code-test release cycle is shortened to the length of an iteration. Therefore any defects and requirements change that creep in at most affect an iteration, and thus it is less painful to recover. Moreover, a highly iterative process enables constant validation and re-alignment of requirements by ensuring that at the end of each iteration there is a functional, partially complete system with which the client can evaluate.

3 Risks

[4] 8. Briefly describe two ways to manage risks in XP. (Midterm, Spring 2009)

Solution:

1. Having short iterations

2. Do the simplest thing that could possibly work
3. Customer picks most important stories first
4. Developers estimate time. Stories that they cannot estimate (or that have long estimates) are risky. Developers must work (prototype) to make a reliable estimate

[3] 9. (a) The lectures and readings discussed several types of risks for software development projects. Describe three specific kinds of risks, each with an example. (Midterm, Spring 2012)

Solution:
Three main type of risks we discussed in the lecture: product risks, project risks, and business risks. Some specific examples include:

- Problem harder than we thought
- We waste time, take too long
- Customers run out of money and kill the project

[3] (b) You work on a mobile application. You are part of a team of 4. For the current project, only one member, Alan, knows the complete specifications and the core system. Think about one risk that can affect the project. Also provide the risks type, and describe a way to mitigate it. (Spring 2013)

Solution:
One of the risks is that Alan might leave the team. A way to mitigate is to have other team members pair program with him to learn the system. Also, Alan could be asked to write the specifications of the system in a document for future reference.

[3] (c) In your class project you had to perform risk management. Describe three activities involved in risk management. (Spring 2013)

Solution:
Risk management mainly consists of:

- risk detection and classification (severity, probability)
- defining and updating mitigation plans for identified risks
- periodical risk monitoring. Risks change in severity and probability over time. Mitigation plans change over time

Projects usually have a risk table that contains above mentioned risks.

[3] 10. Describe a risk associated with your group's project. How is your group dealing with this risk? If it is not dealing with it, suggest how it should. (Midterm, Spring 2012)

Solution:

The answer, of course, is specific to a given project as are the possible ways one might manage or mitigate that risk. For example, to manage the risk of developers leaving the project, you could use pair programming to teach developers about different parts of the system and to make sure that knowledge of the system is spread as widely as possible.

4 Specifications

3 11. Are tests suitable to express specifications? In your answer provide three arguments. (Spring 2013)

Solution:

Yes.

- It is easy to see if the program meets the specifications.
- It is easy to understand both the specification and implementation.
- It forces you to think about API design.
- Writing tests to express specifications is easy for the developers. They can be implemented in the same language as the system language. The developer does not need to learn a new language.

4 12. Compare and contrast using test cases as specification versus using formal specifications. Please include two similarities and two dissimilarities. (Midterm, Spring 2012)

Solution:

Similarities:

- Using both methods a developer can abstract away the details of the system and express what the system is expected to do (the "what") without talking about the way the system concretely maintains that property (the "how").
- Both methods present the developer with a concrete language to express the specifications unambiguously (unlike natural language specifications).

Dissimilarities:

- One usually needs to learn a new language for writing formal specifications. This is not the case when using tests for writing specifications.

- Checking conformance of the system with respect to the specifications is easier with tests. In case of formal specifications, conformance is checked by sophisticated tools like model checkers or static analyzers. In case of tests, conformance can be checked by simply running them.
- Tests are too specific (usually test for a given input value or a given system state) unlike formal specifications.

[2] 13. What is a key difference between requirements and specifications? (Hint: it is not just the language they are written in, as both can be written in English.) (Midterm, Spring 2010)

Solution:

A key difference is that requirements are written from the (non-technical) perspective of the user, while specifications are written from the (technical) perspective of the developer and/or tester.

[2] 14. Parameterized unit tests (PUTs), also called theories in JUnit, allow writing test method with parameters. PUTs/theories are closer to specifications than to traditional unit tests. For example, the following PUT is for a set:

```
1  void testInsert(Set s, Object o) {  // for any set and object
2    s.insert(o);                      // after inserting the object into the set
3    assert s.contains(o);             // the set contains the object
4  }
```

Write a new PUT (in the language used in your project or in Java) for the following: when inserting an element into a set that didn't contain that element, the set size increases by one. (Hint: You can use s.size() to get the set size.) (Midterm, Spring 2010)

Solution:

```
1 void testInsertSize(Set s, Object o)
2   {
3     if (s.contains(o)) return;
4     int initialSize = s.size();
5     s.insert(o);
6     assertEquals(initalSize+1, s.size());
7   }
```

5 Software Quality Assurance

[2] 15. Read this statement: "External characteristics of quality are the only kind of software characteristics that users care about. Therefore programmers should focus only on external quality

characteristics throughout the development process". Is this true or false? Explain your answer. (Spring 2013)

Solution:

False. Although external quality characteristics are indeed the only software characteristics that users care about, the programmers focus should also be on the internal quality characteristics that provide long term support for, and in many cases induce, external qualities. With time, unkept software rots and it becomes hard to change, error prone, hard to reuse and hard to comprehend. This in turn brings a halt to the external quality characteristics since it would require tremendous costs to keep and extend them.
The above statement is however true if the system does not need to change afterwards.

[2] 16. The manager wants the highest possible defect detection rate. Which quality control techniques should he use? Why? (Spring 2013)

Solution:

The manager should use a combination of techniques since none of the techniques provide adequate detection rate in isolation.

[4] 17. Software has both external and internal quality characteristics. Give two external and two internal characteristics and briefly describe them. (Spring 2012)

Solution:

Refer to section 20.1 in Code Complete 2 which lists several examples of internal and external characteristics.
External characteristics: correctness, usability (look and feel of the UI, for example), integrity, robustness, etc.
Internal characteristics: maintainability, testability, portability, readability, etc.

[2] 18. In lecture 15 on software quality assurance, we discussed the notion that "Quality is Free". Explain what it means. (Spring 2010)

Solution:

Refer to the graph from the lecture slides that plots the effort of software development vs. the quality of the software. The phrase means that improving the quality of software does not necessarily require more effort on the part of the developers. In fact as you improve the quality, the development costs/efforts come down.

[2] 19. What is the difference between failure analysis and flaw analysis? (Spring 2010)

Solution:

Failure analysis determines what flaw in the program caused the failure to happen. Flaw analysis determines what is wrong in the development process that allowed the flaw to be created and go undetected.

[4] 20. Explain why it is important to keep track of information about bugs found while developing software? Provide two examples of how this information can be used. (Spring 2010)

Solution:

Bug reports can be used to track information about the reported bug, its description, the severity of the bug, etc. This information can be used to

- track what flaws led to bugs as and when they are discovered.
- prioritize flaws that need to be fixed depending on the severity of the bugs.
- estimate reliability of the system or modules in the system based on the bug reports.

[3] 21. Some believe that quality is all about conformance to requirements and standards. Argue why that isnt true by showing what is missing from that statement. (Midterm, Spring 2009)

Solution:

1. Requirements and standards can evolve/change over time
2. Requirements can be unachievable

[3] 22. The stakeholders of a project often desire different kinds of quality attributes from the project. These attributes overlap: some of them help one another; some of them might be in conflict with one another. Explain how increasing the integrity of the system could decrease the efficiency of the system. (Midterm, Spring 2009)

Solution:

Refer to Section 20.1 of Code Complete 2. We accepted reasonable answers provided you were able to demonstrate that you know what integrity and efficiency mean in this context.

23. The users of your product care about the external qualities of it; the developers of your product care about the internal qualities of it.

2 (a) Name one external quality that your users care about and describe what your group is doing to achieve that quality.

Solution:
Ease of use, GUI look and feel, etc.

2 (b) Similarly, name one internal quality that your group cares about and describe what your group is doing to achieve that quality.

Solution:
Maintainability, coding standards, code coverage obtained through automatic testing, etc.

2 24. Give one advantage and one disadvantage of using tests as specifications. (Midterm, Spring 2008)

Solution:
Advantage: Easy to see if tests meet the specifications (run them!).
Disadvantage: Hard to test something that cannot happen (without complicated mocking).

2 25. Give one advantage and one disadvantage of using formal methods as specifications. (Midterm, Spring 2008)

Solution:
Advantage: Precise; can actually prove the existence/absence of certain properties about the system.
Disadvantage: Hard for most people to read and understand.

26. *Everything should be made as simple as possible, but not one bit simpler.*
- Albert Einstein (attributed) (Midterm, Spring 2008)

2 (a) Why is a simple design a good design?

Solution:
A simple design is easier to understand and modify. Thus, it is likely to be easier to implement.

2 (b) Besides simplicity, what are two other characteristics of a good design?

Solution:
Maintainability, reusability, testability, etc.

2 (c) Give two concrete steps that your group is doing to achieve a simple design.

Solution:

The answers should be based on the process that you are using. Simple Design: Do the simplest thing that could possibly work Test-driven development: Use tests to guide features that need to be implemented. Refactoring: Refactor to eliminate duplication, consolidate classes, form clearer abstractions.

6 Code Reviews

2 27. One of the main benefits of code reviews is finding defects. List two other benefits of doing code reviews. (Spring 2013)

Solution:

With enough eyeballs all bugs are shallow. Code reviews are also beneficial for suggesting code improvements, transferring programming knowledge, suggesting alternate solutions and increasing ownership of the code.

4 28. One goal of conducting technical reviews is improving the quality of the product being developed. Name two other goals of performing these reviews. (Spring 2010)

Solution:

- Help in assessing the quality of the software and the progress made by the software development team
- Provide a way to educate new developers and ensure that developers are consistent.

2 29. In Code Complete 2, Steve McConnell indicates that when performing technical reviews and inspections it is not desirable to solve identified problems during the review. Explain why. (Spring 2010)

Solution:

The purpose of the review meeting is to find as many defects as possible and not to discuss solutions with the reviewers.

2 30. Despite the cost of performing reviews, they can often reduce overall development costs. Why is this the case? (Spring 2010)

Solution:

Code reviews

- have been found to be very effective in finding defects
- improve the overall productivity of the team through collaborative participation
- help in assessing the quality of the software and the progress made by the software development team
- provide a way to educate new developers and ensure that developers are consistent.

[4] 31. Code walkthroughs have been shown to be less effective at finding problems than formal inspections. Give two reasons why you might still choose to conduct code walkthroughs. (Spring 2010)

Solution:

Code walkthroughs

- provide a way to obtain a low overhead feedback from a possibly large review group
- are good for training reviewers if they have not participated in code reviews before

[2] 32. Briefly describe the roles of the moderator and reviewer in a code review. (Spring 2009)

Solution:

Moderator: moderates the review meeting, solicits reviews from the reviewers about the problems they see in the code under review, ensures all components of the code under review are presented/discussed during the meeting.
Reviewers: provide feedback to the authors about the problems they see in the code under review with respect to the previously agreed checklist.

[2] 33. How is a code review different from a code walkthrough? (Spring 2008)

Solution:

As opposed to code reviews, walkthroughs

- involve a more high level and much less focused discussion about the code,
- are lower overhead for both the reviewers and the authors. The meetings can be shorter and prior preparation is not required for the reviewers.

- go through more code during the meeting than code reviews which are more focused,
- are good for training reviewers,
- are less effective than code reviews in detecting software errors.