University of Washington
CSE 403 Software Engineering
Spring 2014

Midterm exam

May 12, 2014

Name: ***Solutions*** _____

CSE Net ID (username): _____

UW Net ID (username): _____

This exam is closed book, closed notes. You have **50 minutes** to complete it. It contains 21 questions and 7 pages (including this one), totaling 100 points.

Before you start, please check your copy to make sure it is complete. Turn in all pages, together, when you are finished. **Write your initials on the top of *ALL* pages** (in case a page gets separated during test-taking or grading).

Please write neatly; we cannot give credit for what we cannot read.

Good luck!

Page	Max	Score
2	12	
3	20	
4	22	
5	12	
6	16	
7	18	
Total	100	

1 True/False

(2 points each) Circle the correct answer. T is true, F is false.

1. **T** / **F** A way to estimate the duration of one very large task is to break it into smaller sub-tasks, estimate each one independently, and add up the estimates. However, this approach is usually not practical because of the difficulty of knowing all the sub-tasks and of unforeseen interactions among them.

   This approach is a good way to better understand the subtasks and their interactions, which is a large part of the purpose of planning.

2. **T** / **F** Coupling is an expression of the principle that dependencies should be minimized.

3. **T** / **F** Cohesion is an expression of the principle that dependencies should be minimized.

4. **T** / **F** The agile and waterfall development methodologies use the same software lifecycle, just at different granularities/tempos.

5. **T** / **F** If you are having trouble communicating with another group member, one effective strategy is to tell them how you think they feel and then ask if you're correct.

   We gave credit for both answers.

6. **T** / **F** For a static analysis such as that of Coverity, eliminating false positives (a report of a non-flaw) is more important than eliminating false negatives (a non-report of a flaw).

   Coverity is a bug-finding tool. By contrast, for a verification tool, it is more important to eliminate false negatives than false positives.

2 Very short answer

7. (2 points per example, 16 points total) Give two examples of each type of requirement, in 1 phrase each, or say that none exist.

   (a) Functional, behavioral requirements:
   user-oriented use cases: "can edit record", "can add new user"

   (b) Non-functional, behavioral requirements:
   performance, latency, fault-tolerance, confidentiality, security, accessibility

   (c) Functional, non-behavioral
   does not exist

   (d) Non-functional, non-behavioral
   standards/regulation compliance, test coverage, maintainability, extensibility, release deadlines, design guide compliance

8. (4 points) Non-functional requirements often imply some functional requirements. Give an example of this for a hypothetical electronic health records system.

   Non-functional requirement _____

   implies functional requirement _____

   must be accessible ⇒ font size must be adjustable
   compliance ⇒ must be able to show audit
   easy to learn ⇒ training/first run mode

9. (2 points each, 6 points total) "Software maintenance" is a bit of a misnomer since software does not wear out nor require lubrication/adjustment. State three tasks that are part of software maintenance, in one phrase each. Make them as different as possible.

   (a) ***Updating your software to work in new environments (new languages, libraries, architectures, surrounding systems).***

   (b) ***Finding and fixing bugs, performance problems, scalability problems, etc.***

   (c) ***Adding new features at the request of customers.***

10. (3 points each, 9 points total) Name the three main types of wrappers (design patterns for delegation). For each, either describe it or give an example use of it, in one phrase. If you cannot remember the names, then at least give the description/example.

    (a) Name: ***adapter***
    Compared to the delegate, the wrapper offers the same functionality, but a different interface. Examples include renaming a method, converting units (radians vs. degrees), or implementing one method in terms of another.

    (b) Name: ***decorator***
    Compared to the delegate, the wrapper offers different functionality, but the same interface. Examples include bordered windows in a window manager, or UnmodifiableList.

    (c) Name: ***proxy***
    Compared to the delegate, the wrapper offers the same functionality, and the same interface. The proxy controls access to other objects: remote method invocation, locking, security, lazy creation, etc.

11. (7 points) In 1 phrase each, describe four work-environment factors that are more strongly correlated with productivity and happiness than salary is. Circle one that you can do to contribute to your coworkers' productivity and happiness.

    (a) ***Positive feedback.***

    (b) ***Feeling like you contributed to building a high-impact, high-quality product.***

    (c) ***Enjoying the interactions with your teammates.***

    (d) ***Doing interesting work.***

    (e) Feeling like you're always learning and improving.

    (f) Feeling like they had real and sole responsibility for a part of the product.

    (g) It is not acceptable to provide answers that have no known correlation or depend on personal preference, such as a quiet work environment, free snacks/beer, regular working hours, or work/life balance.

3 Short answer

12. (3 points each) Below you will complete the phrase "*goal1*, *goal2*, and *goal3* – choose two" about the software engineering triangle. Then, for each of the goals, explain in 1 phrase or short sentence a situation in which it would be better to sacrifice that goal in order to achieve the other two.

    (a) Goal 1: ***Good***

    Developing an Facebook game capitalizing on the short-lived popularity of Flappy-Bird style games requires low time-to-market and low cost (because the application's success is unpredictable), so you sacrifice high quality.

    (b) Goal 2: ***Fast (Deliver soon)***

    Developing an open-source encryption library for which funding and volunteer developers are scarce requires low cost and high quality, so you must sacrifice low time-to-market.

    (c) Goal 3: ***Cheap***

    Developing software for a spacecraft that is launching very soon requires delivery soon and high quality, so you must sacrifice low cost.

    A common incorrect answer was to describe, for example, a project that sacrificed low cost because it needed high quality. A complete answer would have also described how that project also needed low time-to-market. It is possible to have low cost and high quality if you sacrifice low time-to-market.

13. (3 points) Both the evolutionary prototyping and the spiral development model greedily choose tasks by highest risk. How do they differ with respect to what risks they prioritize? Answer in one sentence.

    Spiral minimizes risk overall. Evolutionary prototyping minimizes risk of failing to meet customer needs (due to lack of feedback).

    Evolutionary prototyping minimizes the risk of not meeting the needs of clients, spiral minimizes overall risk by repeatedly increasing cost in a round-robin fashion for each phase of development.

14. (3 points) In 1 sentence, describe a situation in which a distributed version control system is superior to a centralized VCS.

    Many correct answers. Advantages of a distributed VCS include:

    - ***checkpoint work without publishing to teammates***
    - ***commit and examine history when not connected to the network***
    - ***history of actual units of work done, not logical history that linearizes to when each logical task was complete. (If you use this rationale, you need to explain why this would be important to know.)***
    - ***more effective merging algorithms***
    - ***share changes selectively with teammates***
    - ***flexibility in repository organization and workflow***
    - ***faster performance***

    A common incorrect answer states DVCS is better for large projects or teams. But, Microsoft and Google use centralized version control on huge projects.

    Another common incorrect answer is when different team members are concurrently working on different portions of the code. CVCS works well in that scenario.

15. (3 points) In 1 sentence, describe a situation in which a centralized VCS is superior to a distributed VCS.

    For novice developers, centralized VCS is easier to learn. It also has a simpler workflow.

    Centralized VCS allows a developer to check out a single folder, which can be good when you are working on just a part of an enormous repository but may cause problems if a developer fails to commit/update all relevant directories.

    It is not acceptable to say "small projects" or "projects with 1 developer" without further explanation/justification. Even a single developer can take advantage of DVCS features.

    It is not acceptable to say "projects where merge conflicts are unlikely": which are those?

16. (2 points each) When designing a system, why is it typically unwise to express designs in terms of code? Give two reasons that are as different as possible, in one phrase or sentence each.

    (a) ***Code is too detailed and specifies too much. Someone reading in is likely to get mired in trivia instead of high-level design issues.***

    (b) ***Code is expensive to create and change.***

    (c) Code constrains the design to the limitations of a specific programming language. Non-coders cannot understand the design; an example is managers and other members of your team. Saying that customers cannot understand the design is not adequate without further explanation. Customers care about visual and conceptual design but not about the kind of architectural design that you might express in code.

    Just saying "the language might change" is not enough: it's on the right track but is not enough to show understanding.

17. (3 points) What is the purpose of a method summary in Coverity, the static analysis tool described by Eric Lippert in his guest lecture?

    A method summary is a method specification: it describes how a method behaves, so that a tool can analyze clients without re-analyzing the method every time. A simple example of a method summary is the Java type signature, which allows the compiler to do type-checking without analyzing the body of every called method.

    A method summary has nothing to do with explaining/listing errors in the method body, with testing, with people understanding what a method does, with paths through code, or with call graphs.

18. (3 points) Eric Lippert stated in his guest lecture, "There is not a C language." What did he mean? Answer in 1 sentence.

    There are many dialects of the C language, and each one requires custom support in a tool that purports to analyze C programs.

    One reason is that the C language standard leaves some aspects of its semantics undefined, and different compilers make different implementation choices.

    Another reason is that many C compilers add proprietary extensions to the C language.

    Yet another reason is that the C standard has evolved over time.

    It is not acceptable to assert that C is not a language, or that it has no standard. Both of these assertions are false.

19. (3 points each) In 1 sentence each, give two advantages of organizing teams around functionality, in the context of testing.

    (a) ***It minimizes the distance between the developer and tester of a functionality, which encourages closer collaboration between testers and developers.***

    (b) ***It encourages testers to have a say in design and implementation decisions relating to a piece of functionality, which can result in software better suited to testing.***

    It is not acceptable to say that developers could write tests faster. They may be less familiar with testing practices.

20. (3 points each) In 1 sentence each, give two advantages of organizing teams around job functions/titles, in the context of testing.

    (a) ***It maximizes the distance between the developer and tester of a functionality, which prevents testers from becoming infected with testing biases through contact with developers.***

    (b) ***It encourages closer collaboration between testers, and increases the availability of other testers for help, advice and opportunities to improve testing skills.***

    It is not acceptable to say that organizing by title provides more hierarchy and order. Responsibility can be delegated in either arrangement.

    It is not acceptable to say that developer could write features faster because they aren't concerned with writing the tests. If a developer breaks a test, they must fix the code or the test, regardless of who wrote what. In the long run, catching feature bugs early with tests takes less time than learning about bugs from testers or users.

21. (3 points each) In 1 sentence each, describe two important differences between libraries and frameworks.

    (a) ***A framework typically controls the main execution of a program, whereas a library is used by a user-created main procedure.***

    (b) ***Libraries focus on re-using code and implementations, whereas frameworks focus on re-using the architecture or overall structure of a program.***

    (c) Libraries are typically more specialized than frameworks. A framework will often encompass many different functionalities in order to be a "one-stop-shop" for their user base, whereas libraries focus on a single functionality. More precisely, libraries tend to be more cohesive than frameworks.

    It is not acceptable to claim that frameworks are single-language, or that they are inherently more complex or harder to learn. Those facts are commonly true, but counterexamples exist for each of these claims, and they are not the most important differences.