SWE 425: Software architecture and design Q&A1

Q#	Question	Answer
1	The later a problem is found in the software development process, the harder it is to fix. Explain why, in one sentence	Some answers are: • Other work may be built on a flawed foundation, so correcting the original problem may require significant rework. • If the software has already been widely distributed, it might be expensive to get a fix out. • The developer(s) involved might have forgotten the context surrounding the bug, so it'll be harder to fix. • The bug might be located in any existing code, and there's more to search through as time passes.
2	Give an example of a design pattern whose use is obvious from a class diagram but not from a sequence diagram. (Don't choose one that is built into (some) programming languages, such as inheritance.) Explain why, in 1 sentence.	Composite: the members of a class are of a type that allows similar operations (perhaps they implement an interface in common with the container class). Observer: especially easy if there's an (observes) notation on an arrow. For many patterns it's possible to argue either way.
3	Event-driven programming is appropriate for user interfaces and user interaction. • Give a different example domain.	Networking. Disk access. Operating system interrupts. Web servers. A thermostat (takes action when the temperature is too high or low).
4	Consider two components A and B. Two software engineers, Laurel and Hardy, measure the dependences between A and B. Laurel uses these dependences when computing cohesion, and Hardy uses these dependences when computing coupling. Is this possible, if both engineers are performing a sensible and useful computation? In 1–2 sentences, explain why or why not.	Yes. Laurel is considering a larger module C that contains both A and B as implementation details. Hardy is considering the implementation of C, and thinking of A and B as modules.
5	It is cheaper and faster to fix known bugs before you write new code. Why? In one phrase or sentence each, give three reasons. Give reasons that are as different from one another as possible.	• You are familiar with the code now. A related reason is that the bug will be harder to find and fix later.

Q#	Question	Answer
		• Later code may depend on this code. A related reason is that a bug may reveal a fundamental problem. • Leaving all bugs to the end will make it harder to understand and keep to the schedule, because it's hard to predict how long bug fixing will take. • An overfull bug database is demoralizing and is likely to be ignored. • You will be able to add tests for the bug once it's been fixed to avoid future issues. • Avoid feature creep.
6	After you find a bug but before fixing it, you should create a test case for it. In one sentence each, give three reasons that this is a good idea. Give reasons that are as distinct as possible.	• Ensures that your fix solves the problem. Don't add a test that succeeded to begin with! A related reason is to avoid writing a test for a bug that you fixed, but that isn't the problem indicated by the original bug fix. • It helps you understand the bug and define the desired system behavior. • It helps you know when you are done with bug fixing. A related reason is repeatability, and efficiency when debugging: the test is easy to run in an automated way to determine whether your fix works.
7	"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.
8	In 1 phrase each, describe four work-environment factors that are more strongly correlated with productivity and happiness than salary is. Circle	Some correct answers: (a) Positive feedback.

Q#	Question	Answer
	one that you can do to contribute to your coworkers' productivity and happiness.	(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.
9	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.	Some correct answers: (a) Goal 1: Good Developing a Facebook game capitalizing on the short-lived popularity of Flappy-Bird style games require 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'
10	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?	Spiral minimizes risk overall. Evolutionary prototyping minimizes risk of failing

Q#	Question	Answer
	Answer in one sentence.	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
11	In 1 sentence, describe a situation in which a distributed version control system (DVCS) is superior to a centralized VCS.	Here are some correct answers. Advantages of a DVCS 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.
12	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.	Some correct answers: (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.
13	In 1 sentence each, give two advantages of organizing teams around functionality, in the context of testing	Some correct answers: (a) It minimizes the distance between the developer and tester of a functionality,

Q#	Question	Answer
		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.
14	In 1 sentence each, describe two important differences between libraries and frameworks.	Some correct answers: (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-stopshop” for their user base, whereas libraries focus on a single functionality. More precisely, libraries tend to be more cohesive than frameworks.
15	In two or fewer sentences, support or refute the claim that all developers should participate in alpha testing at the end of every sprint or project milestone.	Support: Sufficient testing will increase delivered software quality and keep developers on track. This is particularly relevant in safety-critical software, where alpha testing may reveal severe defects. Refute: The resources required to implement alpha testing every sprint may detract too much from productive development. The benefit of the testing may not make up for the decreased productivity.

Q#	Question	Answer
16	In three sentences or fewer, describe the differences between spiral development and waterfall development.	Full credit answer will discuss how spiral relies on continuous releases of prototypes to reduce risk. Waterfall is divided into discrete phases over the course of an entire project. Spiral contains aspects of waterfall, just iterated multiple times during a project
17	In a few words, identify which phase of software development costs the most amount of money and resources?	Maintenance. Lecture and readings covered industrial reports about costs of fixing defects at various phases of development. Maintenance rules all.
18	A fresh startup called *FaceBack* has designed software that can identify the back of a person's head given a picture of their face. Some team member's support adopting an official company list restricting which programming languages can be used as the team grows. In three or fewer sentences, support or refute the use of such approved programming language lists in terms of risk management and process.	Support: A restricted set of programming languages may ease code reviews as reviewers know what to expect (lowered risk, process simplification). It will also lower overhead associated with enforcing code style (lowered risk). There may also be benefits to simplifying integration testing (process simplification). Refute: May diminish developer productivity, especially if parts of the team are more familiar with one language over another (increased risk and uncertainty; how to plan/allocate developer effort?). Additionally, such a practice may increase defects among developers who are not familiar with approved languages (increased risk). Encourages implementation bias by eliminating potential languages that may simplify some aspect of the project (increased risk, process complexity). Other answers may apply.

Q#	Question	Answer
19	Support or refute the claim that a statement with no live variables can be removed without affecting the program's correctness.	Support: Statements with dead variables implies the statement affects nothing. Such dead code can be removed without loss. Refute: Statements like x = 1/0 might appear dead, but removing them may influence intended behavior.