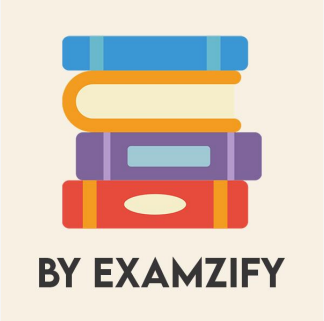
Arizona State University (ASU) CSE360 Introduction to Software Engineering Exam 1 Practice (Sample)

Study Guide



Everything you need from our exam experts!

Questions

1. What is the main objective of software acceptability?

   A. To ensure comprehensive feature coverage

   B. To meet customer specifications

   C. To execute with minimal CPU usage

   D. To ensure maintainability

2. What do the software specification activities involve?

   A. Ensuring user documentation is available

   B. Defining the software along with operational constraints

   C. Testing the software for quality assurance

   D. Maintaining the software throughout its life cycle

3. In Incremental Development, what happens to the cost of accommodating changing customer requirements?

   A. It increases significantly

   B. It stays the same as initial estimates

   C. It decreases

   D. It becomes unpredictable

4. Which phase concludes a sprint in Scrum methodology?

   A. The planning meeting

   B. The work and process review

   C. The resource allocation meeting

   D. The final iteration

5. In Agile Project Management, what is the principal responsibility of the project manager?

   A. To create detailed technical documents

   B. To manage the project to ensure timely software delivery

   C. To supervise all coding activities personally

   D. To implement a rigid plan-driven approach

6. Which of the following is a component of the software production process?

A. Feedback

B. Implementation

C. Research

D. Marketing

7. How is risk classification primarily defined?

A. By the emotional impact on team members

B. By the type of risk and what is affected

C. By the duration of the project

D. By the management style used

8. What is primarily ensured through effective risk management in software project management?

A. Increased team sizes

B. Reduction in project costs

C. Minimization of uncertainties

D. Complete avoidance of all potential setbacks

9. What does a real software process aim to accomplish?

A. Implement user feedback to enhance system features

B. A collaborative environment for developers

C. Specify, design, implement, and test a software system

D. Reduce the time taken for software delivery

10. What challenge is often encountered with the Incremental Development process?

A. Loss of documentation

B. Degradation of system structure over time

C. Excessive costs leading to project failure

D. Delay in feedback from customers

Answers

1. B
2. B
3. C
4. B
5. B
6. B
7. B
8. C
9. C
10. B

Explanations

1. What is the main objective of software acceptability?

A. To ensure comprehensive feature coverage

B. To meet customer specifications

C. To execute with minimal CPU usage

D. To ensure maintainability

The main objective of software acceptability is fundamentally tied to ensuring that a software product meets customer specifications. This aspect is critical because it directly influences the satisfaction of stakeholders and end-users. When a software application aligns with the requirements and expectations set forth by the customer, it is deemed acceptable, often ensuring that it fulfills its intended purpose and functions as required. Meeting customer specifications includes understanding the needs, preferences, and use cases of the users, which have been defined in the requirements analysis phase of software development. By focusing on this objective, developers and project stakeholders can ensure that the software is likely to be well-received and effectively utilized in its target environment. This alignment fosters trust, encourages user adoption, and contributes to the overall success of the project. While comprehensive feature coverage, minimal CPU usage, and maintainability are indeed important characteristics of good software, they do not capture the essence of acceptability as directly as meeting customer specifications does. Ensuring acceptability is about satisfying user needs and user-defined criteria, which is the core focus when determining if the software is ready for deployment.

2. What do the software specification activities involve?

A. Ensuring user documentation is available

B. Defining the software along with operational constraints

C. Testing the software for quality assurance

D. Maintaining the software throughout its life cycle

The software specification activities primarily focus on defining the software system itself, including its features, functionalities, and operational constraints. This involves gathering and documenting requirements from stakeholders, which forms the foundation for what the software should accomplish. It's essential for aligning the development team's understanding with the users' needs, ensuring that the final product meets those specified requirements. Specifications serve as a formal agreement on what the software will and will not do, which guides subsequent design, implementation, and testing phases. Operational constraints are also a critical part of this process, as they dictate the environments in which the software must operate, performance metrics, and possibly regulatory compliance aspects. By thoroughly detailing these elements, stakeholders can mitigate risks early and provide clear guidelines for developers, which ultimately leads to a more successful software project.

3. In Incremental Development, what happens to the cost of accommodating changing customer requirements?

A. It increases significantly

B. It stays the same as initial estimates

C. It decreases

D. It becomes unpredictable

In Incremental Development, the approach emphasizes building a system in small, manageable segments or increments, allowing feedback and adjustments with each completed segment. As a result, this method inherently accommodates changing customer requirements more effectively than traditional methods. When changes occur, they can be integrated into the next increment rather than requiring a complete overhaul of the project or entire system. Because this iterative process allows for validations and refinements after each increment, the cost of accommodating changes tends to decrease as adjustments can often be made more easily and with less disruption compared to a linear development approach. The flexibility built into Incremental Development encourages ongoing customer collaboration, making it cheaper and easier to implement changes based on user feedback, rather than facing higher costs associated with reworking large parts of a project later in the development cycle. This adaptability is a key strength of the Incremental Development model, as it supports a continuous alignment with customer needs and expectations.

4. Which phase concludes a sprint in Scrum methodology?

A. The planning meeting

B. The work and process review

C. The resource allocation meeting

D. The final iteration

In Scrum methodology, the phase that concludes a sprint is the work and process review, which is known as the Sprint Review. This meeting takes place at the end of each sprint and serves as a crucial opportunity for the development team to showcase the work that has been completed during the sprint. Stakeholders, including product owners and team members, gather to review the potentially shippable product increment and provide feedback. The Sprint Review is instrumental in ensuring transparency and alignment among team members and stakeholders about the progress made and future direction. It emphasizes collaboration, allowing the team to discuss what went well, what could be improved, and adjusts the backlog based on this feedback. This review sets the stage for the next sprint, ensuring that all members are on the same page and contributing to the overall Agile process. Other options don't encapsulate the conclusion of a sprint accurately. The planning meeting occurs at the beginning of a sprint, focusing on setting goals and planning what will be accomplished. The resource allocation meeting is typically not a formal part of the Scrum process, as Scrum is designed to be adaptive and self-organizing, reducing the need for extensive resource allocation discussions. The final iteration is not a recognized term within the Scrum framework and does not refer to an event in

5. In Agile Project Management, what is the principal responsibility of the project manager?

A. To create detailed technical documents

B. To manage the project to ensure timely software delivery

C. To supervise all coding activities personally

D. To implement a rigid plan-driven approach

In Agile Project Management, the principal responsibility of the project manager focuses on managing the project to ensure timely software delivery. This encompasses a variety of roles that are critical to the Agile process, including facilitating communication between team members, removing obstacles that may hinder progress, and ensuring that the team remains aligned with the project goals and timelines. Agile promotes flexibility and responsiveness to change, which means the project manager must adeptly balance stakeholder expectations while fostering a collaborative environment within the team. This role is more about guiding the team and supporting their work rather than micromanaging or adhering to a strict plan, which aligns with the core values of Agile methodologies that prioritize teamwork, customer collaboration, and adaptive planning. Other choices focus on aspects that are not central to Agile principles. For example, creating detailed technical documents contradicts the Agile preference for working software over comprehensive documentation. Supervising all coding activities personally would conflict with the Agile emphasis on empowering teams to self-organize and take ownership of their tasks. Lastly, implementing a rigid plan-driven approach goes against Agile's fundamental nature, which encourages adaptability and flexibility in project management.

6. Which of the following is a component of the software production process?

A. Feedback

B. Implementation

C. Research

D. Marketing

The software production process encompasses several critical components to ensure successful development and deployment of software. Implementation, as identified in the correct answer, refers to the phase where actual coding and development take place. This is a key component because it transforms design specifications into a functioning software product. During this phase, developers write code according to established designs, frameworks, and methodologies, making it integral to the overall production process. While feedback, research, and marketing play important roles in the broader context of software development, they do not represent the direct activities involved in software creation. Feedback usually pertains to the evaluation of the software after its implementation, helping in iterative improvement. Research often involves preliminary studies or investigations that inform software requirements and design but does not constitute the actual production of the software itself. Marketing, while crucial for promoting the software once completed, occurs outside the production process, focusing on user acquisition rather than development. Thus, implementation is distinctly recognized as a core component of software production.

7. How is risk classification primarily defined?

A. By the emotional impact on team members

B. By the type of risk and what is affected

C. By the duration of the project

D. By the management style used

Risk classification is primarily defined by the type of risk and what is affected because this approach allows for a structured and systematic way to analyze potential challenges that may arise during a project. By categorizing risks based on their nature—such as technical risks, project management risks, organizational risks, etc.—project managers can better understand the implications of each risk on project outcomes. Understanding what specifically is affected—whether it be resources, schedule, quality, or stakeholder satisfaction—enables teams to prioritize their responses effectively. This kind of classification helps in developing targeted mitigation strategies and allows teams to allocate resources where they are needed most, ultimately increasing the likelihood of project success. Other considerations, such as emotional impact, project duration, or management style, do not provide the same level of utility in a structured risk assessment framework. Emotional impacts, while relevant in team dynamics, do not offer a clear categorization of risks. Similarly, while project duration and management style can influence how risks are managed, they do not directly correspond to the nature of the risks themselves or the areas of the project they impact. Thus, focusing on the type of risk and its effects provides a more practical and actionable approach to risk management in software engineering.

8. What is primarily ensured through effective risk management in software project management?

A. Increased team sizes

B. Reduction in project costs

C. Minimization of uncertainties

D. Complete avoidance of all potential setbacks

Effective risk management in software project management is primarily aimed at the minimization of uncertainties. By identifying potential risks and assessing their impact, project managers can put in place strategies to either mitigate or avoid these risks, thus reducing the uncertainties that can derail a project. Minimizing uncertainties allows for a more predictable project environment, enhances decision-making capabilities, and leads to more effective resource allocation. This proactive approach ensures that the project team is better prepared for potential challenges, which in turn contributes to achieving project goals and maintaining schedules. While reduction in project costs can be a beneficial outcome of effective risk management, it is not the primary objective. Additionally, although increasing team sizes may seem advantageous, it does not directly correlate with effective risk management practices. Complete avoidance of all potential setbacks is unrealistic; instead, the focus is on managing and minimizing the impact of uncertainties that inevitably arise during the software development process.

9. What does a real software process aim to accomplish?

A. Implement user feedback to enhance system features

B. A collaborative environment for developers

C. Specify, design, implement, and test a software system

D. Reduce the time taken for software delivery

A real software process primarily aims to specify, design, implement, and test a software system, which encompasses the entire lifecycle of software development. This systematic approach ensures that all critical phases of software creation are addressed, from gathering requirements (specification) to building and coding the system (implementation), followed by thorough verification (testing) to ensure the software meets the intended purpose and quality standards. Each phase plays a vital role: specifying sets the foundation by understanding user needs, designing sets the blueprint for development, implementing brings the design to life through coding, and testing verifies that the end product functions correctly and fulfills the requirements. By focusing on these stages, a real software process ensures that the software developed is not only functional but also of high quality, which is essential for user satisfaction and project success.

10. What challenge is often encountered with the Incremental Development process?

A. Loss of documentation

B. Degradation of system structure over time

C. Excessive costs leading to project failure

D. Delay in feedback from customers

The Incremental Development process allows for iterative releases of software, enabling teams to add functionalities in manageable segments. While this approach has many advantages, one challenge commonly faced is the degradation of system structure over time. As new features are added incrementally, developers might prioritize immediate functionality over adhering to initial architectural principles. This can lead to a situation where the codebase becomes increasingly complex and harder to maintain. Without careful management and refactoring, the overall design integrity of the software can weaken, making it difficult to implement future changes efficiently. This can result in technical debt, where the cost of later modifications rises because of a deteriorating underlying structure, ultimately affecting maintainability and scalability. The other options present potential issues but do not encapsulate the systematic concern that arises specifically from the incremental nature of the development process, making the challenge of system structure degradation particularly salient in this context.