Boost up Your Certification Score

Scrum
PSM-III
Professional Scrum Master III

For More Information – Visit link below:

https://www.examsboost.com/

Product Version

✓ Up to Date products, reliable and verified.
✓ Questions and Answers in PDF Format.

Visit us at: https://www.examsboost.com/test/psm-iii

Latest Version: 6.0

Question: 1

SIMULATION
Describe the difference between feature and component teams, and how they hold up when viewed from the perspective of the Scrum Guide.

A. See the Explanation below for complete answer

Answer: A

Explanation:
In Scrum, team structure significantly impacts the ability to deliver value. Two commonly discussed structures are component teams and feature teams. Although the Scrum Guide does not explicitly define these terms, it strongly favors the characteristics of feature teams through its definition of a Scrum Team.
Component teams are organized around technical specialties or system components, such as database, frontend, or middleware teams. Their work typically represents partial contributions to a product feature, requiring coordination and handoffs across multiple teams to deliver customer value. As a result, component teams often introduce dependencies, delay integration, and struggle to produce a usable Increment independently within a Sprint.
Feature teams, in contrast, are organized around delivering complete product features or Product Backlog Items. They are cross-functional and possess all the skills required to design, build, test, and deliver a "Done" Increment of value. Feature teams minimize dependencies and can independently deliver customer-facing functionality each Sprint.
From the Scrum Guide perspective, feature teams align more closely with Scrum principles:
The Scrum Guide states that Scrum Teams are cross-functional, which directly supports feature teams and challenges component team structures.
Scrum requires each Sprint to produce a usable Increment. Feature teams can meet this expectation, while component teams usually cannot without reliance on other teams.
Scrum is based on empiricism (transparency, inspection, and adaptation). Reduced dependencies in feature teams improve transparency and enable faster inspection and adaptation.
Scrum emphasizes value delivery and accountability. Feature teams maintain clear ownership of outcomes, whereas component teams fragment accountability across technical silos.
While component teams may exist due to legacy structures or technical constraints, they represent organizational impediments rather than an ideal Scrum implementation. From a Professional Scrum Master III perspective, moving toward feature teams supports agility, improves value delivery, and better enables Scrum as defined in the Scrum Guide.

Visit us at: https://www.examsboost.com/test/psm-iii

Question: 2

SIMULATION
In what ways does the Scrum Master attend the Sprint Retrospective?

A. See the Explanation below for complete answer

Answer: A

Explanation:
The Sprint Retrospective is a formal Scrum event where the Scrum Team inspects how the last Sprint went with respect to individuals, interactions, processes, tools, and their Definition of Done, and identifies improvements for future Sprints. The Scrum Master attends the Sprint Retrospective in multiple, complementary ways, consistent with the Scrum Guide.
First, the Scrum Master joins the Sprint Retrospective as a Scrum Team member. The Scrum Guide defines the Scrum Team as consisting of the Product Owner, Developers, and the Scrum Master. Therefore, the Scrum Master is not an external observer but a full participant in the event. As such, the Scrum Master actively inspects people, processes, and tools, and contributes insights based on their perspective and experience, while remaining respectful of the team's self-management.
Second, the Scrum Master often facilitates the Sprint Retrospective. According to the Scrum Guide, the Scrum Master is accountable for ensuring that Scrum events take place and are productive. Facilitation may include helping the team create a safe environment, encouraging openness, ensuring balanced participation, keeping the discussion focused on improvement, and helping the team stay within the timebox. However, facilitation does not imply control; the Scrum Master facilitates to serve the team, not to direct outcomes.
Third, the Scrum Master supports empiricism during the Retrospective. By fostering transparency, encouraging honest inspection, and helping the team identify actionable improvements, the Scrum Master strengthens the Scrum pillars of transparency, inspection, and adaptation. The Scrum Master may also help the team turn improvement ideas into concrete actions that can be planned for the next Sprint.
Finally, the Scrum Master helps ensure that the Sprint Retrospective results in meaningful adaptation. While the Scrum Team decides what improvements to implement, the Scrum Master supports the team in identifying impediments, coaching on improvement techniques, and helping remove organizational or systemic obstacles that are beyond the team's direct control.
In summary, the Scrum Master attends the Sprint Retrospective by joining as a full Scrum Team member, participating in inspection, often facilitating the event, and supporting continuous improvement and empiricism. This balanced participation ensures that the Retrospective remains a powerful mechanism for learning and adaptation rather than a ritualistic meeting.

Question: 3

SIMULATION

The Product Owner asks the Development Team to pick up a very urgent item late in Sprint that was not forecasted, nor is it related to the Sprint Goal. The Development Team believes it can pick this up, as it is close to meeting the Sprint Goal. But, this would involve not meeting their process improvement goal agreed upon during the last Sprint Retrospective. The Product Owner argues that, as it's the highest priority to satisfy the customer, the needs of the customer have a higher priority than the process improvement goal for the team. What is your view on this as a Scrum Master?

A. See the Explanation below for complete answer

Answer: A

Explanation:
From a Scrum Master's perspective, this situation must be approached by balancing respect for Scrum accountabilities, protection of empiricism, and long-term value delivery, rather than reacting solely to short-term urgency.
First, it is important to reaffirm that the Development Team owns the Sprint Backlog. According to the Scrum Guide, once the Sprint has started, changes to the Sprint Backlog are negotiated only between the Product Owner and the Development Team, and the Development Team has the final say on whether additional work can be taken on. Therefore, the Product Owner cannot unilaterally force the urgent item into the Sprint, even if it represents the highest customer priority. If the Development Team believes it can incorporate the item without jeopardizing the Sprint Goal, it may choose to do so—but this remains their decision.
Second, the Scrum Master should help the Product Owner understand that not all priorities are equal within a Sprint. The Sprint Goal provides focus and stability, and work that is not related to the Sprint Goal introduces risk. While satisfying the customer is important, Scrum explicitly values sustainable improvement and learning. The process improvement goal agreed upon during the Sprint Retrospective represents a deliberate investment in the team's effectiveness. Sacrificing this improvement for short-term delivery may create a local optimization that harms long-term customer value.
Third, the Scrum Master should coach both the Product Owner and the Development Team on the systemic impact of slowing process improvements. Continuous improvement is a core expectation of Scrum, and the Scrum Guide states that the Scrum Team should plan ways to increase quality and effectiveness. When improvement goals are repeatedly deprioritized, delivery predictability, quality, and morale eventually decline—directly affecting customers. Therefore, the Product Owner's argument that customer needs always outweigh improvement work reflects a short-term mindset that the Scrum Master should challenge through education and coaching.
Fourth, this situation should be inspected during the Sprint Retrospective. The team should reflect on why urgent, unplanned work appears late in the Sprint, whether it represents a recurring pattern, and how this impacts Sprint Goals and improvement commitments. The Scrum Master should facilitate this discussion to ensure transparency and learning, rather than blame.
Finally, if this behavior becomes a pattern, the Scrum Master must take a more active stance. This includes teaching and reminding the Scrum Team that at least one improvement from the Sprint Retrospective should be planned into the upcoming Sprint. This protects the intent of the Retrospective and ensures that improvement is not treated as optional or expendable work.

Visit us at: https://www.examsboost.com/test/psm-iii

Question: 4

SIMULATION
Your Scrum Team has one month Sprints. The development team argues that since this period is quite long, a Daily Scrum is a bit too much. They instead want a weekly update meeting. What is your opinion on this?

A. See the Explanation below for complete answer

Answer: A

Explanation:
From a Scrum Master's perspective, replacing the Daily Scrum with a weekly update meeting is not consistent with Scrum and would significantly weaken the team's ability to inspect and adapt effectively, regardless of the Sprint length.
First, Scrum explicitly defines the Daily Scrum as a required event. The Scrum Guide states that the Daily Scrum is a 15-minute event held every working day of the Sprint for the Developers. The length of the Sprint—whether one week or one month—does not change the purpose or necessity of this event. Therefore, by choosing not to have a Daily Scrum, the team would no longer be practicing Scrum, but rather a Scrum-like process.
Second, the Daily Scrum is not a status meeting. Its primary purpose is to allow the Developers to inspect progress toward the Sprint Goal, synchronize their work, and adapt the Sprint Backlog as needed. A weekly meeting dramatically reduces the frequency of inspection and adaptation, delaying the discovery of issues such as integration problems, misalignment, or risks to the Sprint Goal.
Third, removing the Daily Scrum negatively impacts transparency, one of Scrum's three pillars of empiricism. Without daily synchronization, important information about progress, impediments, and discoveries becomes stale or hidden. This reduced transparency increases the likelihood that work will drift away from agreed standards, fail to integrate properly, or no longer support the Sprint Goal by the end of the Sprint.
Fourth, the argument that a one-month Sprint justifies less frequent inspection reflects a misunderstanding of empiricism. Longer Sprints increase risk, which makes frequent inspection and adaptation more important, not less. The Daily Scrum provides a regular opportunity to realign the team and respond early to emerging problems, thereby reducing waste and rework.
Finally, as a Scrum Master, my role is to teach and coach the Scrum Team on the purpose and value of Scrum events. Rather than removing the Daily Scrum, I would help the Developers improve how they use it—for example, ensuring it focuses on progress toward the Sprint Goal and actionable planning for the next 24 hours, instead of turning into a reporting session.

Question: 5

SIMULATION

Visit us at: https://www.examsboost.com/test/psm-iii

You have been appointed the Scrum Master for a brand new product your organization is planning to develop. A Product Owner has also been appointed. Initially, fifteen developers will work on the product. What approaches are common for forming teams for this product, and how do they likely benefit or hinder the Product Development effort?

A. See the Explanation below for complete answer

Answer: A

Explanation:
When starting development of a brand new product with fifteen developers, forming effective teams is a critical early decision that significantly influences the success of product development. From a Scrum Master's perspective, multiple approaches are commonly used in practice. Each approach offers distinct benefits and drawbacks when evaluated against Scrum principles such as self-organization, cross-functionality, and value delivery.
1. Facilitating Teams to Self-Organize
One common approach is to facilitate the developers in forming teams themselves. This approach aligns strongly with Scrum, as the Scrum Guide states that Scrum Teams are self-managing and decide internally how best to accomplish their work.
Benefits:
Allowing teams to self-organize promotes empowerment, ownership, and accountability. Developers can use their existing knowledge of each other's strengths, weaknesses, and working styles to form balanced teams. This often increases motivation and psychological safety, both of which support high performance.
Hindrances:
For a new product, this process can be messy and time-consuming, especially if developers lack experience in forming effective teams. Teams may optimize for comfort or familiarity rather than cross-functionality, potentially leading to skill gaps or imbalanced teams.
2. Forming Two or Three Cross-Functional Feature Teams
Another common approach is to deliberately form two or three cross-functional feature teams, each containing all the skills necessary to deliver working product increments.
Benefits:
This approach closely matches how Scrum describes teams. Cross-functional feature teams can independently deliver integrated, "Done" Increments of the product, improving flow, reducing dependencies, and supporting empiricism. All necessary skills are available within the team, enabling faster inspection and adaptation.
Hindrances:
In the context of a brand new product, teams may not yet know which skills are actually required, making it difficult to form truly balanced teams upfront. Additionally, specialists may feel isolated and lose regular interaction with peers who share the same expertise across teams.
3. Forming Teams Based on Specialization (Component Teams)
A third approach is to organize teams according to technical specialization, such as front-end and back-end teams. These are often referred to as component teams.
Benefits:
This structure allows specialists to work closely together, enabling fast knowledge sharing, technical consistency, and deep expertise in specific components of the system. It can feel efficient, especially in the early stages of development.

Visit us at: https://www.examsboost.com/test/psm-iii

Hindrances:
From a Scrum perspective, this approach significantly hinders value delivery. Component teams struggle to deliver complete, integrated features independently and introduce dependencies and handoffs. This makes it harder to produce a usable Increment each Sprint and is not how Scrum describes teams, even though it remains a commonly used strategy in many organizations.
Scrum Master Perspective and Conclusion
As a Scrum Master, my role is not to mandate a single team structure, but to coach and facilitate the organization toward structures that best enable Scrum. While all three approaches are seen in practice, Scrum clearly favors self-organizing, cross-functional feature teams because they maximize learning, transparency, and the ability to deliver value each Sprint.

Visit us at: https://www.examsboost.com/test/psm-iii

Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

15 USD Discount Coupon Code:

G74JA8UF

FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer