Doğukan Ertunga Kurnaz
21702331
Section 1

Answers:

1.

a. Technical debt is a metaphor. It is a fictitious debt that must be repaid-with interest when received. It is a concept which is created by Ward Cunningham[1] who is one of the creators of the Agile Technique and one of the pioneers of Extreme Programming. This debt can occur in the software development stages. It is a problem that will arise when the software team temporarily solves an issue. Sometimes it may be preferred in order to shorten the deadline of the developed product, but this debt should definitely be closed before adding new features to the product. Otherwise, a problem with the product may become unfixable. In some cases, it is caused by the technical inadequacy of the software team. Developments made while there is technical debt in the software project may cause major irreversible problems in the future. There are a couple of ways to improve to reduce Technical Debt. Some of the Technical Debt can be addressed with static and dynamic code analysis tools.

b. In a Scrum environment, not only the Scrum Master, the whole team will be going to responsible for handling technical debt. That is because the nature of Scrum requires team collaboration and built-in checks & balances[2]. There may be a lead to guide other team members, but the whole team is in the process. Also, it is a good opportunity for juniors to see how seniors are approaching the technical debt.

c. It depends actually, the important part is how we handle and respond to those technical debts as a team. As a developer, I prefer to order the debts according to their effects on the product and their severity. Because I would have other tasks to do, I distribute those technical debts in my daily workflow from the backlog. But when I realize that, there will be a new feature that will use the debt part of the software, I quickly move to fix that technical debt.

d. Technical debts can be discussed in the daily Scrum meetings or in the meetings before the sprint week. The scrum master can define a sprint time or a special week to fix all these backlog items including technical debts. This week, all team members have to stop what they’re doing and start working on the technical debts or bugs.

2. The Nexus Scrum guide is a guide for developing and maintaining scaled product and software delivery initiatives. Nexus uses Scrum as a base. The Nexus framework connects and brings together the work of Scrum Teams[3]. Nexus defines responsibilities, activities, and works. Nexus is built on the foundation of Scrum and some parts of it will be familiar to those who have used Scrum. Nexus is a guide through which multiple numbers of users can be working on a single Product Backlog. Nexus Scrum is a way that enables a large number of teams to produce an Integrated Increment that meets a goal[3]. Cross-Team Refinement is a thing that, enables the Scrum team to reduce their dependency on each other. This is important because before the Sprint each team has a backlog that contains an item with dependency. For example, a product team has to fix a bug in the system but for this bug, another team's action is also needed beforehand. To make this clear Nexus use the Cross-Team-Refinement technique.

3. Extreme programming (XP) is a methodology for software development process. It aims to improve software quality by collecting feedbacks from the changing customer requirements. In this type of agile software development, the "releases" is very frequently and it includes short development cycles aimed at increasing productivity and introducing control points from which new customer requirements can be adopted. In XP[4]:

- All code must have unit tests
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

From the developer perspective, I think these rules can improve the software development process but actually it depends to the other aspects of the project. For example, mission critical systems such as healthcare systems should not be developed by using this technique. Testing the uncomplete software on mission is not a good idea. But it is beneficial for most daily use software systems which depends user's expectations. Because this expectations can be change very frequently it is more easy to implement a system which is like puzzle pieces.

4. In many ways, Extreme Programming and Scrum are very similar and compatible techniques. However, there are subtle but important differences between these two methods, which can be summarized as follows:
In the Scrum technique, product development sprints take approximately 1-2 months. This state is shorter in Extreme Programming (usually 1-2 weeks)[5]. In many ways, the content of

sprints cannot be changed in Scrum. Adding or removing a feature is not allowed. This situation is different in Extreme Programming. Extreme Programming teams accept these changes, these changes may come from the user of the product. In Extreme Programming, changes in these iterations are more flexible. To illustrate, the elements of the Scrum Sprint do not change after the planning phase[5]. However, this situation can be changed with some other features in Extreme Programming, when a feature that has not been started to be worked on is desired in Extreme Programming. Another difference between Extreme Programming and Scrum is that in Extreme Programming the priority is the client and their requests. Developed features can be prioritized by the customer[5]. The Scrum team, on the other hand, decides the order of these features.

Another big difference is that unlike Extreme Programming, Scrum does not create any engineering practice. For example, Extreme Programming uses test driven development (TDD)[4]. However, mandating a set of practices can have a negative impact, some on self-controlled teams (Scrum). This situation can be considered as a cons of Extreme Programming. Another shortcoming of extreme programming is that inexperienced teams tend to refactor code without automated tests or TDD (Test Driven Development). For this reason, most people agree that Scrum is better for the software development process.

Scrum	Extreme Programming
sprints take approximately 1-2 months	collaborative work takes usually 1-2 weeks
content of the sprint cannot be changed	content can be changed according to the demand from the customer
priority is defined by the team	priority is the client and their requests
does not create or uses any engineering practices	uses TDD, engineering practices
uses project management practices	does not using project management practices

References:

[1] “Technical Debt.” *Wikipedia*, Wikimedia Foundation, 8 Feb. 2022, https://en.wikipedia.org/wiki/Technical_debt.

[2] “Who Is Responsible for the Technical Debt in the Scrum Project?” *Quora*, https://www.quora.com/Who-is-responsible-for-the-technical-debt-in-the-Scrum-project.

[3] “Online Nexus Guide.” *Scrum.org*, https://www.scrum.org/resources/online-nexus-guide.

[4] Wells, Don. *Extreme Programming Rules*, http://www.extremeprogramming.org/rules.html.

[5] “Difference between Scrum and XP.” *GeeksforGeeks*, 10 June 2019, https://www.geeksforgeeks.org/difference-between-scrum-and-xp/.