Mentimeter

IN5140

Smart Processes and Agile Methods in Software Engineering

Group Session 8

Managing Technical Debt

Mentimeter

Pratical Info on Deliverable 2

Mentimeter

Deliverable 2: Retrospectives & BPMN on Feature Mock-up

Goal:

Reflect on your group’s process from Deliverable 1 and write a short retrospective report

and visualize your group's process using a BPMN diagram

1️⃣ BPMN Diagram (1 page)

- Model your group’s workflow in Deliverable 1
- Include **communication with Product Owners (POs)**
- Follow BPMN syntax (lanes, pools, gateways, message flows, start/end)
- Add short **explanation & assumptions**

If you’re PO group:
Describe how you coordinated and communicated with multiple groups

Mentimeter

2️⃣ Retrospective Report (2 pages)

- Describe team roles
- Reflect on **challenges, requirement changes, and lessons learned**
- Summarize group collaboration and process improvement insights

📄 Format & Submission

- Total: **3 pages (1 + 2)**
- Language: English or Norwegian (consistent)
- Submit both documents in **Devilry**
- **Deadline:** Oct 26, 11:59 PM

Mentimeter

Feedback on the lecture

share what you think!

Mentimeter

Feedback on the lecture

Cool to hear from people in the industry

Good

Interesting to get real life examples from the industry

It was good to hear about how they work in Industry

Mentimeter

Discussion

Managing Technical Debt

Mentimeter

Recap on Technical debt

1. What is *technical debt (explain with the concept of debt, interest, principal)*
2. what are its two main types of technical debt (explain in terms of intentional, unintentional?
3. What does *security debt* mean, and why does it matter?
4. Can technical debt ever be beneficial, and how can a team decide when it's worth taking on versus avoiding?
5. Which technical debts should be prioritized first, and why?

Mentimeter

Managing technical debt and TD management

1. Why do we need to manage technical debt? why does it matter?
2. How can we manage technical debt?
    a. (describe in terms of code smells, architectural smells, refactoring, complexity)
3. How do large companies manage technical debt (i.e., 7 steps)
4. What is *Agile Technical Debt Management*
5. How do proactive, continuous, and reactive approaches differ?
6. How can we tell how mature our technical debt management is?

Mentimeter

Extra topics

1. What is *Lehman's Law of Software Evolution*, and how does it relate to technical debt?
2. How does technical debt affect a team's *flow* and productivity?
3. What are the most common causes of technical debt in projects?
4. How can AI help us manage technical debt and where are its limits?

Mentimeter

15 minute break

Mentimeter

Previous Exam Questions

[Exam 2022]Technical debt is always negative (True, False) and explain why

False: In some cases where you have to reach the market fast, technical debt can be a necessary "sacrifice" to achieve that.

false, sometimes it is smart to have it like startup companies to get quicker to the Market

False: - TD can be useful for startup, because they can focus on the product and features more, which is more important at the start - TD can be useful for prototyping and getting into the market fast

False, in some cases, like with startups and prototyping, it can be benefitial to get the product on market fast or use the least resources possible

False - when you have limited resources, like in a start-up, technical debt can be beneficial because there is a high chance of failure.

False, because technical dept can be intentional. It could for instance be used to produce a prototype quickly, or for when you need to get on the market fast

Mentimeter

Are there any cases where technical debt can be useful?

- High business risk involved
  - e.g. start-ups
- Prototyping

Mentimeter

(2018, 2019)

2. Describe the three main concepts of Technical Debt: the debt, the principal and the interest.

1

Debt: what consecuenses can Happen because of it, so like to fix the Code or design in the future,

Intrest: is like why we do it, so like to Focus on startup features to gain costumers

Dept: short-term choices that could create a more costly future Interest: the experienced negative effects of debt

Principle: Its like a financial dept, if you make bad code today, you pay later. Interest: The extra work or cost caused by TD Debt: The technical shortcuts taken to save time/work

Dept - dept in the context of software development is making decisions/implementations that could be beneficial in the short term, but not in the long term

Dept is the total of problem that need to be fixed. For example how much you own the bank + interest. I think

Debt: principal + interest Interest: the negative effects of the debt Principal: ex. you take shortcuts to get the code done quickly

Principal: The actual dept, the work and cost needed to refactor and improve the TD

Mentimeter

Possible Explanation

TD : design or implementation that is expedient in the short term, but can make a future change most costly or impossible

- invisible, negative value
- Having some TD is ok, keep it under control
- Never try to repay all TD

- The principal (initial cost, cost of replaying on not taking the debt)
- The debt (long term cost, sub-optimal solution)
- The interest (cost of impact)

Mentimeter

(2019)

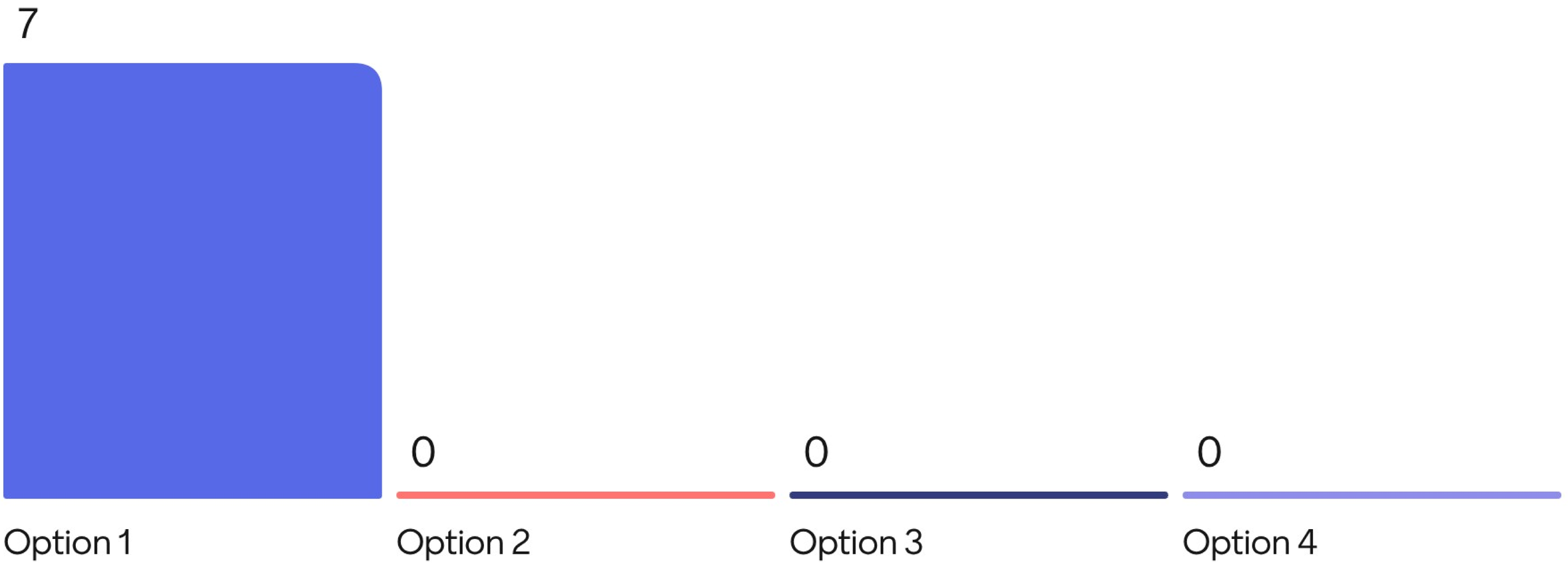
1.3 IN5140 H19 Exam Task 1.3

What is the interest of technical debt?

- The extra-negative impact paid because of the presence of the technical debt
- The effort required to fix the technical debt
- The investment that companies do to manage technical debt
- The technical issue that needs to be solved

2

Mentimeter



Mentimeter

Possible Explanation

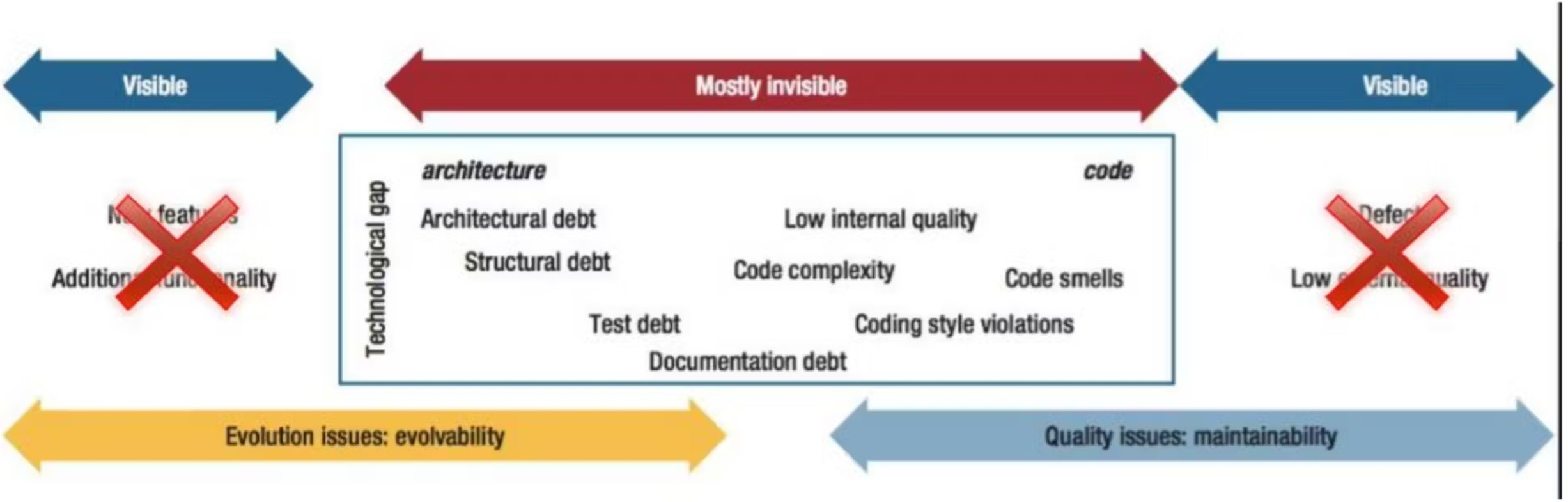
TD : design or implementation that is expedient in the short term, but can make a future change most costly or impossible

- invisible, negative value
- Having some TD is ok, keep it under control
- Never try to repay all TD

- The principal (initial cost, cost of replaying on not taking the debt)
- The debt (long term cost, sub-optimal solution)
- The interest (cost of impact)

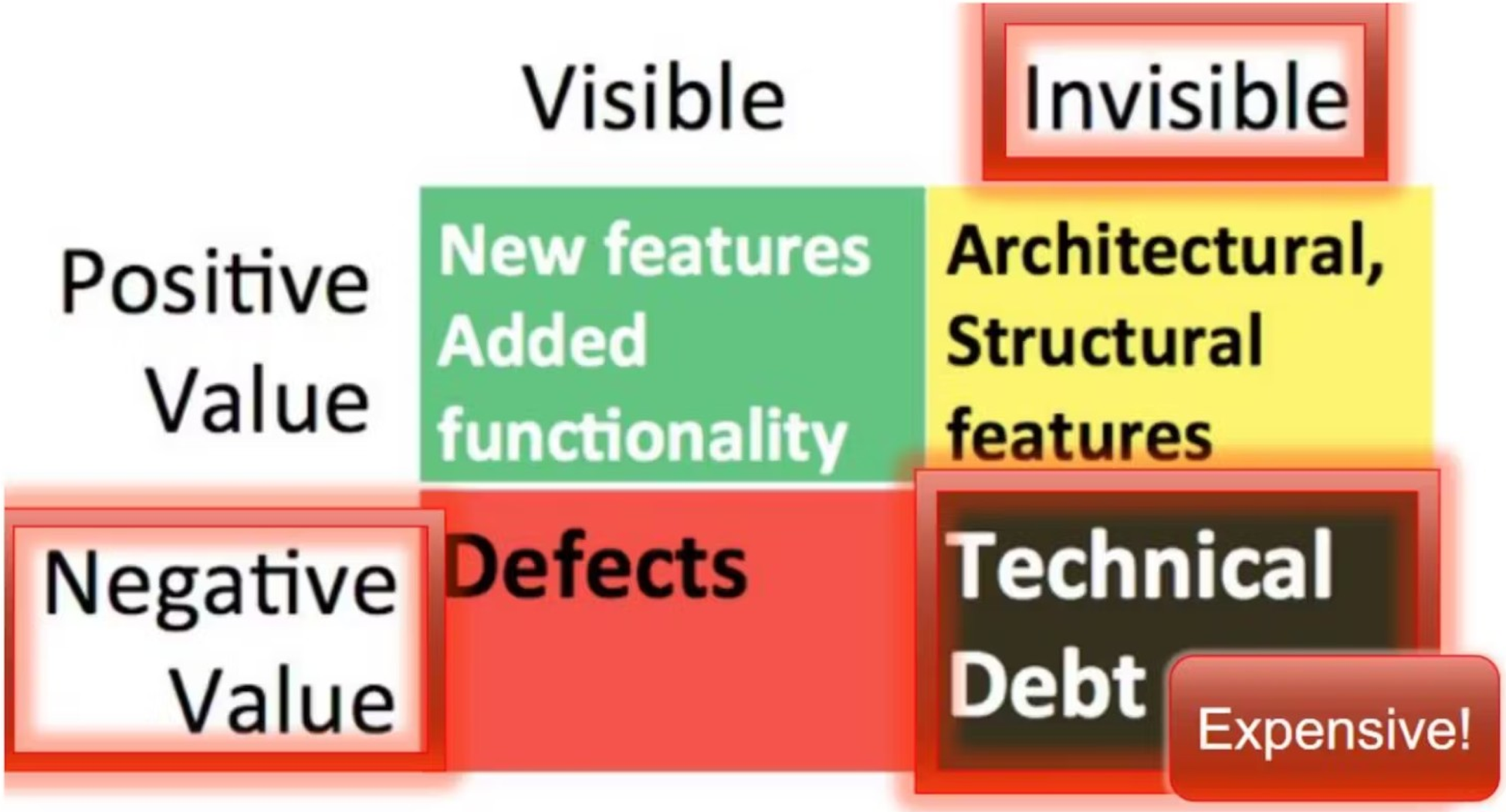
Mentimeter

Technical debt

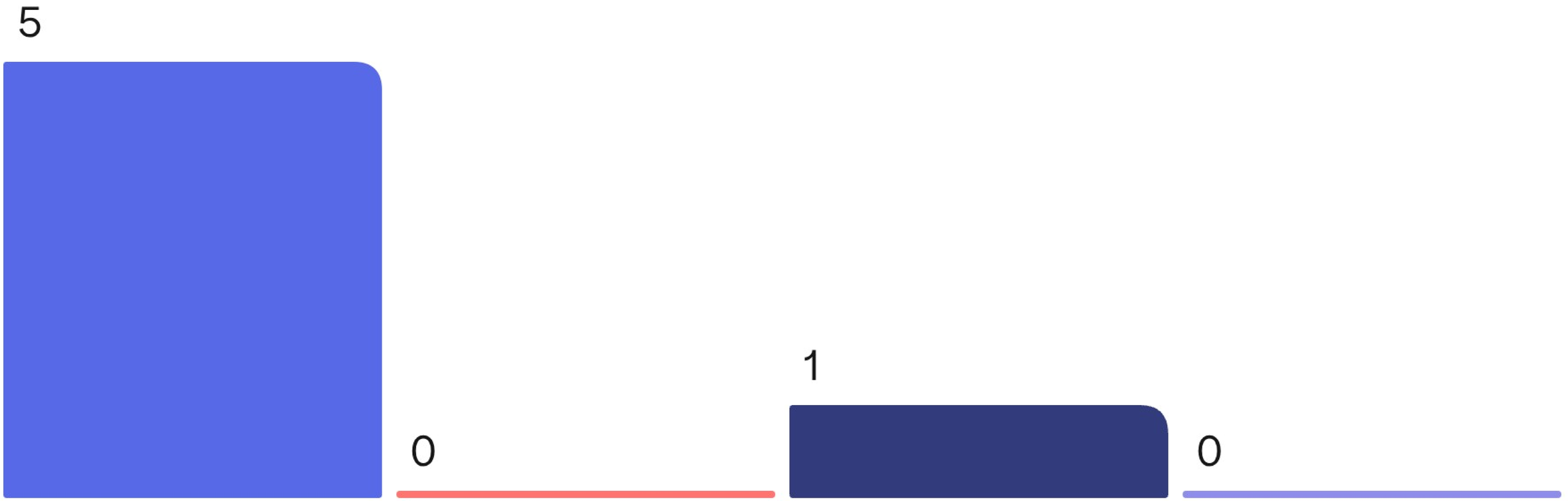


Mentimeter

Technical debt



[Exam 2024]A secruity debt is... [select one alterantive]



Mentimeter

Technical debt and Security debt

Technical debt definition

"In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability."

Security debt definition

"Security debt is a set of design or implementation solutions that hinder or has the potential to hinder the achievement of a system's security goal"

Mentimeter

Technical Debt approaches

Select the correct alternative from the drop-down list:

1. Creating a business case for technical debt is a Select alternative .

2. Facilitate seminars and workshops on how to manage technical debt is a Select alternative.

3. Evaluating the roadmap before refactoring technical debt is a Select alternative

1p bonus if all are correct ones are chosen.

[Exam 2024]

Mentimeter

1. Proactive 2. Continuous 3. Reactive

1. Proactive 2. Continuous 3. Reactive

1. continouse 2. proaktive 3. reactive

1. Proactive 2. Reactive 3. Continous

1-continous, 2-proactive, 3-reactive

reactive, proactive, continous

6

Mentimeter

Technical Debt approaches

Select the correct alternative from the drop-down list:

1. Creating a business case for technical debt is a Select alternative ✖ (Continuous approach, Proactive approach, Reactive approach) .

2. Facilitate seminars and workshops on how to manage technical debt is a Select alternative ✖ (Continuous approach, Proactive approach, Reactive approach).

3. Evaluating the roadmap before refactoring technical debt is a Select alternative ✖ (Continuous approach, Reactive approach, Proactive approach)

[2024]

Mentimeter

How do we fix technical debt?

Proactive approaches:

- Education
- Culture
- Organization
- Process
- Guidelines
- Visualization

Continuous approaches:

- Semi-automatic identifications
- Code reviews
- Retrospectives
- Technical leadership
- Dedicated refactoring sprints

Reactive approaches:

- Impact map
- Roadmap evaluation
- Resources to remove TD
- TD information used in planning and budget

"Global Finance" is replacing its 20-year-old core banking system **Financio** with a new cloud-based system **Zenith**. However, **Financio** has:

- **Incomplete and outdated documentation**
- **Lost developer knowledge**
- **Poorly documented interdependent modules**
- **Formal, slow, document-based communication**
- **Conflicting departmental priorities**
- **Distributed teams and complex deployment process**

[2024]Identify the existing technical debts in Financio, and explain which various interests on the debts the company will have to pay.

Mentimeter

[2024]Identify the **existing technical debts** in Financio, and **explain which various interests** on the debts the company will have to pay.

Mentimeter

Some Technical Debts in Financio:

- **Documentation Debt:** Incomplete and outdated documentation is a major liability. This makes understanding, modifying, and replacing the system incredibly difficult.
- **Architectural Debt:** Poorly documented interfaces and interdependent modules signify potential code complexity and lack of modularity. This increases the risk of unforeseen issues and makes it harder to migrate to a new system.
- **Architectural Debt:** A 20-year-old system likely suffers from outdated architecture, making it inflexible and difficult to integrate with modern technologies.
- **Knowledge Debt:** The loss of original developers and undocumented knowledge creates a significant knowledge gap. This makes it hard to analyze existing functionality and make informed decisions about the new system.

Mentimeter

Some of the additional Interest on these Debts that could be mentioned:

- **Direct Cost of Remediation:** The 5% extra cost to investigate and document dependencies is a direct interest payment on the existing code debt.
- **Increased Development Time:** Unclear requirements, undocumented dependencies, and complex code lead to longer development times, directly translating to higher costs.

Mentimeter

- **Rework and Bug Fixing:** Misunderstandings due to poor documentation and knowledge gaps will result in errors, requiring rework and increasing testing costs.
- **Project Delays:** These issues contribute to missed deadlines, potentially leading to contractual penalties or lost business opportunities.
- **Risk of Project Failure:** In the worst-case scenario, the project could fail entirely due to insurmountable technical challenges, resulting in a massive loss of investment.

Other debts and interests that were mentioned and were plausible, were also accepted.

Mentimeter

Feedback

Mentimeter

Group Feedback & Requests: