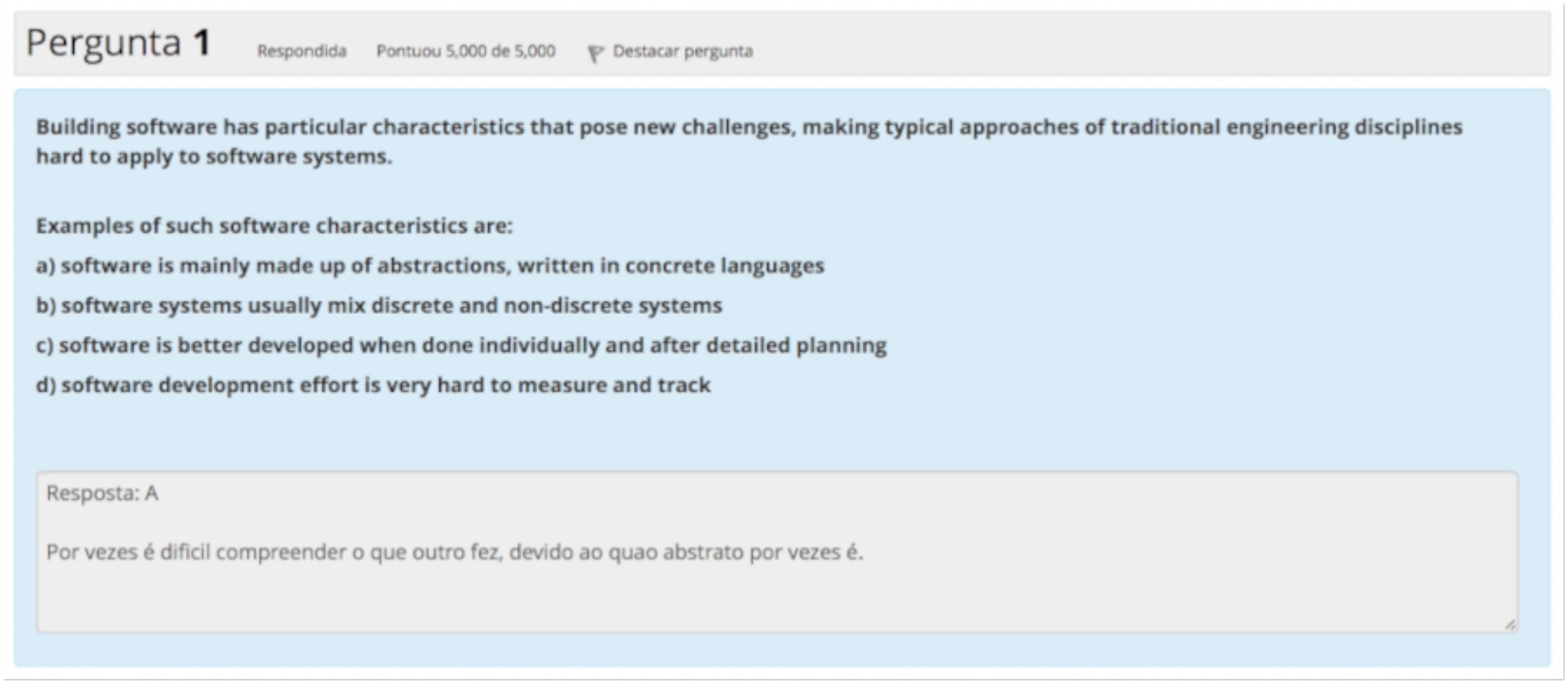


Pergunta **1**

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Building software has particular characteristics that pose new challenges, making typical approaches of traditional engineering disciplines hard to apply to software systems.

Examples of such software characteristics are:

a) software is mainly made up of abstractions, written in concrete languages

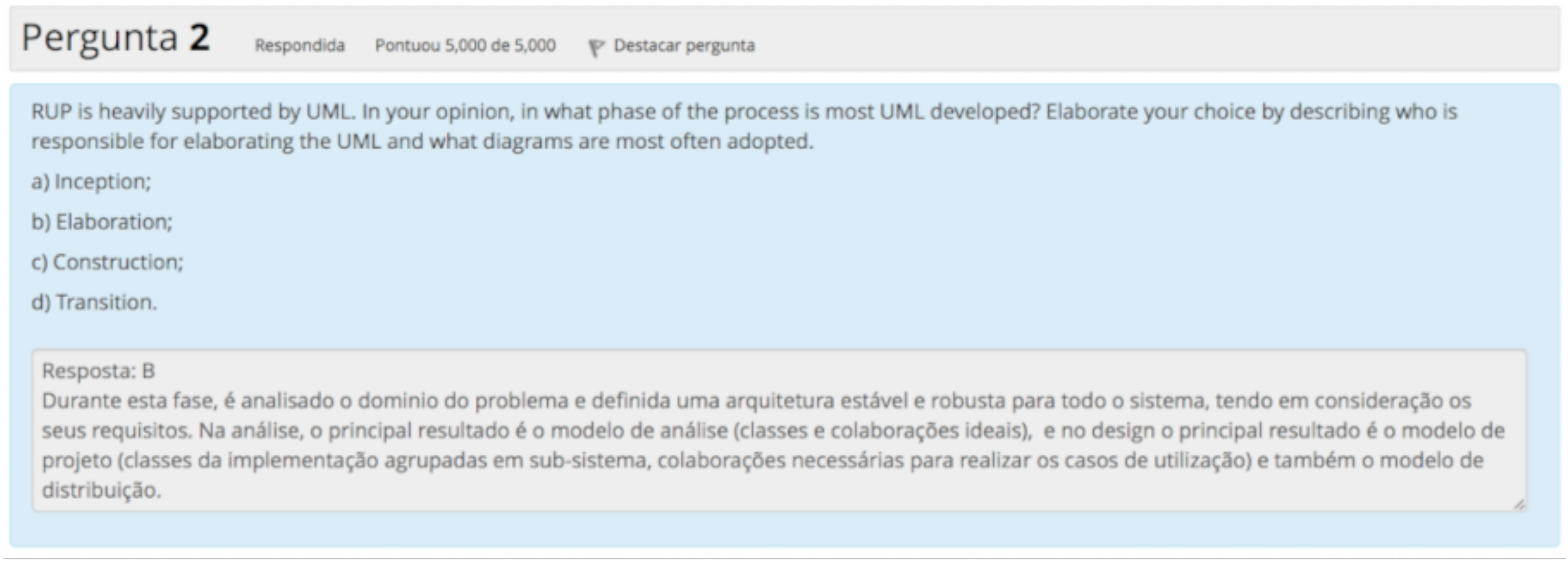
b) software systems usually mix discrete and non-discrete systems

c) software is better developed when done individually and after detailed planning

d) software development effort is very hard to measure and track

Resposta: A

Por vezes é dificil compreender o que outro fez, devido ao quao abstrato por vezes é.



Pergunta **2**

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

RUP is heavily supported by UML. In your opinion, in what phase of the process is most UML developed? Elaborate your choice by describing who is responsible for elaborating the UML and what diagrams are most often adopted.

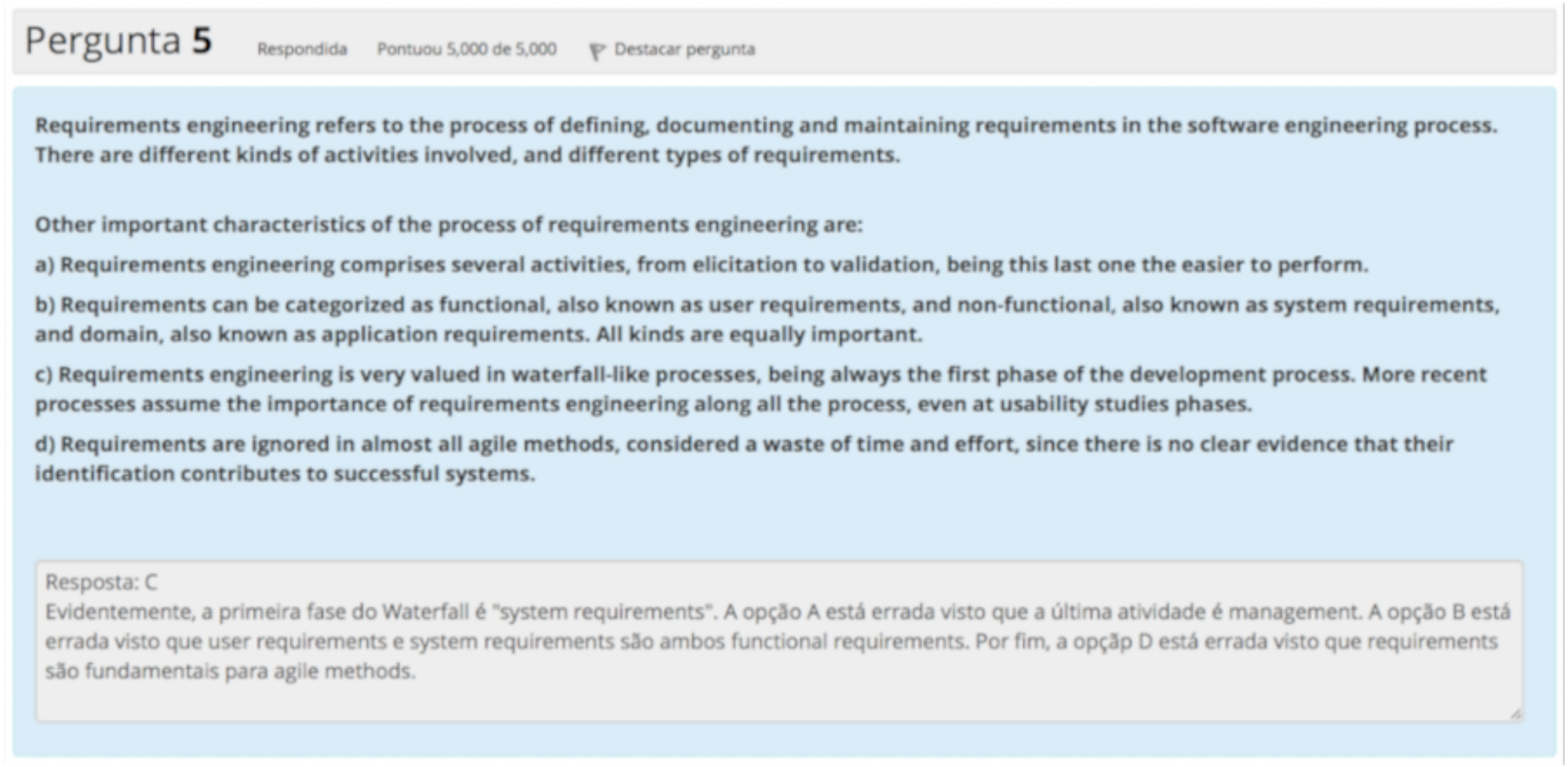
a) Inception;

b) Elaboration;

c) Construction;

d) Transition.

Resposta: B
Durante esta fase, é analisado o dominio do problema e definida uma arquitetura estável e robusta para todo o sistema, tendo em consideração os seus requisitos. Na análise, o principal resultado é o modelo de análise (classes e colaborações ideais), e no design o principal resultado é o modelo de projeto (classes da implementação agrupadas em sub-sistema, colaborações necessárias para realizar os casos de utilização) e também o modelo de distribuição.



Pergunta **5**

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Requirements engineering refers to the process of defining, documenting and maintaining requirements in the software engineering process. There are different kinds of activities involved, and different types of requirements.

Other important characteristics of the process of requirements engineering are:

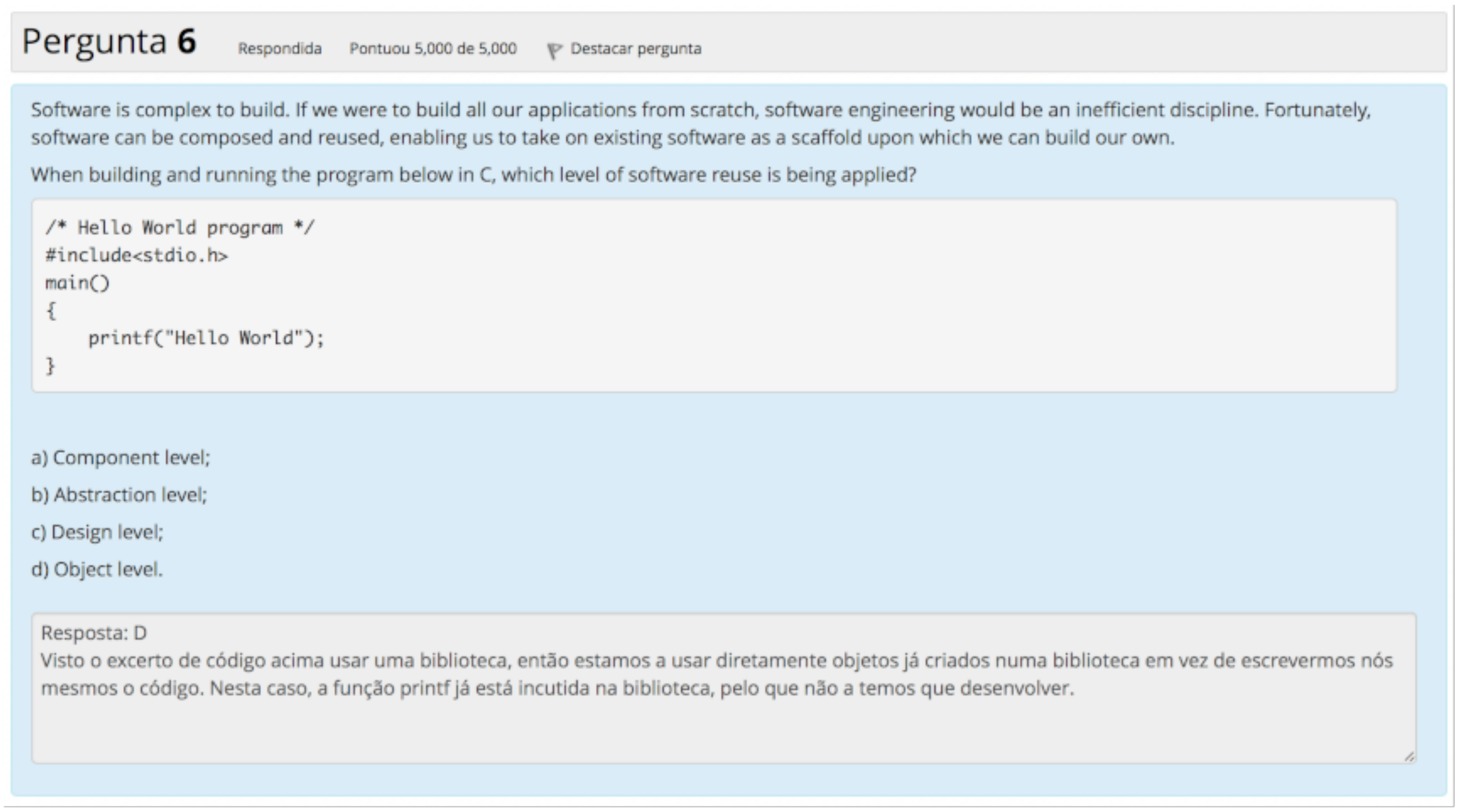
a) Requirements engineering comprises several activities, from elicitation to validation, being this last one the easier to perform.

b) Requirements can be categorized as functional, also known as user requirements, and non-functional, also known as system requirements, and domain, also known as application requirements. All kinds are equally important.

c) Requirements engineering is very valued in waterfall-like processes, being always the first phase of the development process. More recent processes assume the importance of requirements engineering along all the process, even at usability studies phases.

d) Requirements are ignored in almost all agile methods, considered a waste of time and effort, since there is no clear evidence that their identification contributes to successful systems.

Resposta: C
Evidentemente, a primeira fase do Waterfall é "system requirements". A opção A está errada visto que a última atividade é management. A opção B está errada visto que user requirements e system requirements são ambos functional requirements. Por fim, a opçăp D está errada visto que requirements são fundamentais para agile methods.



Pergunta **6**

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Software is complex to build. If we were to build all our applications from scratch, software engineering would be an inefficient discipline. Fortunately, software can be composed and reused, enabling us to take on existing software as a scaffold upon which we can build our own.

When building and running the program below in C, which level of software reuse is being applied?

```
/* Hello World program */
#include<stdio.h>
main()
{
    printf("Hello World");
}
```

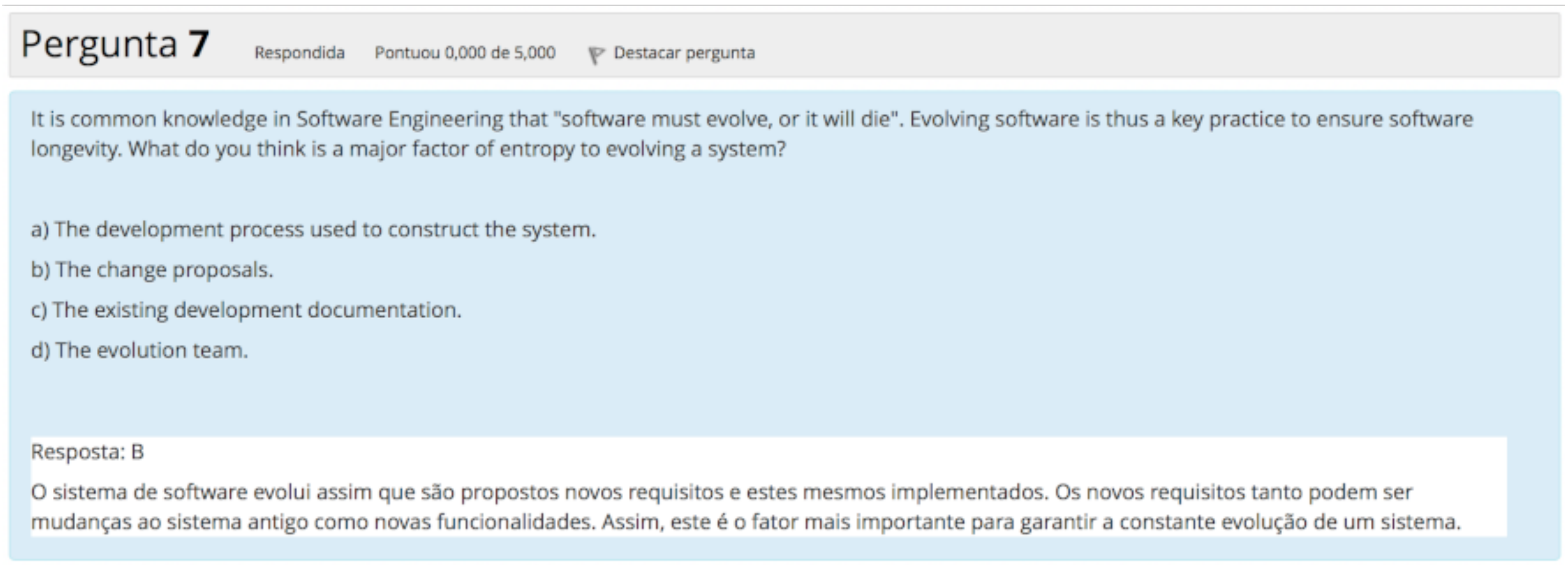
a) Component level;

b) Abstraction level;

c) Design level;

d) Object level.

Resposta: D
Visto o excerto de código acima usar uma biblioteca, então estamos a usar diretamente objetos já criados numa biblioteca em vez de escrevermos nós mesmos o código. Nesta caso, a função printf já está incutida na biblioteca, pelo que não a temos que desenvolver.



Pergunta **7**

Respondida Pontuou 0,000 de 5,000 Destacar pergunta

It is common knowledge in Software Engineering that "software must evolve, or it will die". Evolving software is thus a key practice to ensure software longevity. What do you think is a major factor of entropy to evolving a system?

a) The development process used to construct the system.

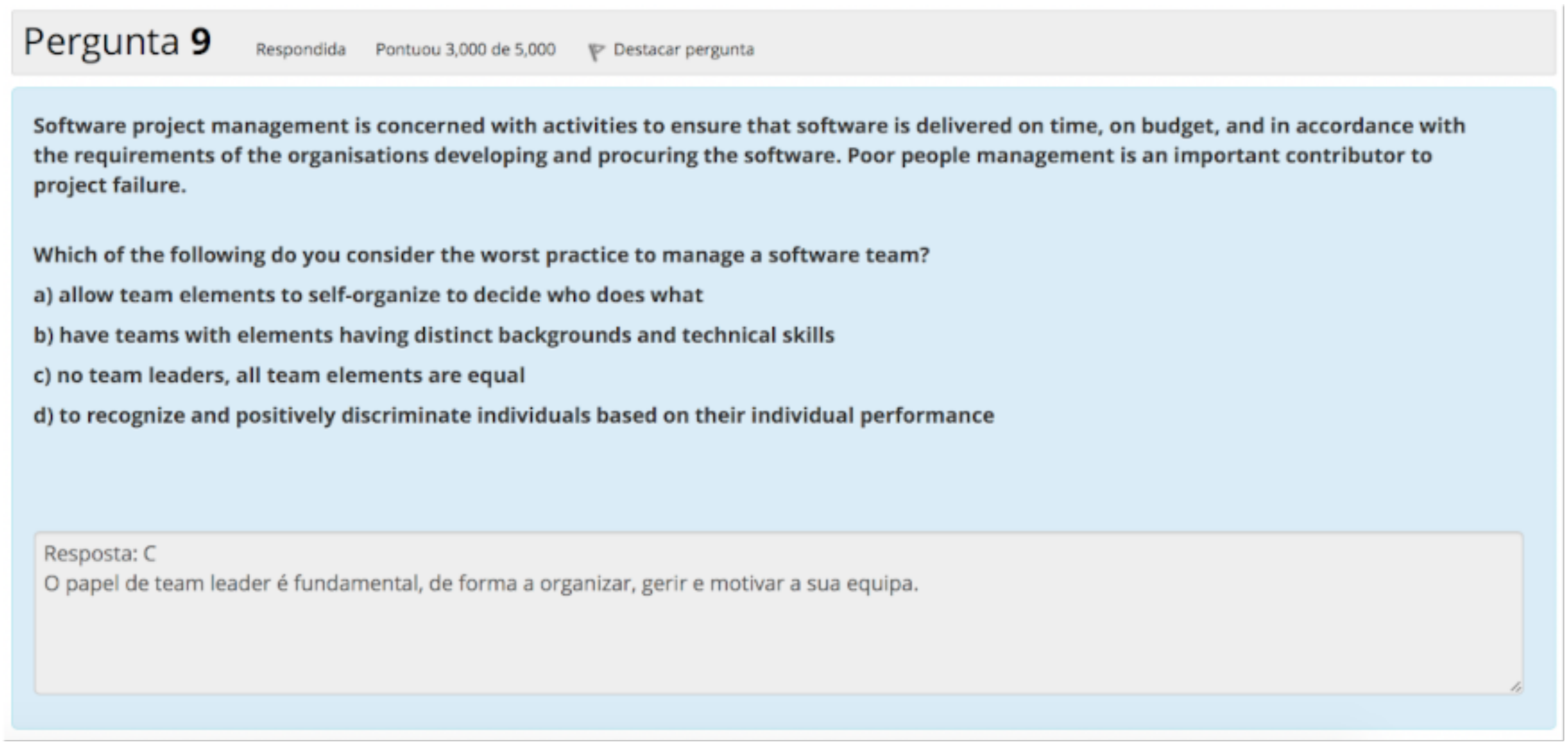
b) The change proposals.

c) The existing development documentation.

d) The evolution team.

Resposta: B

O sistema de software evolui assim que são propostos novos requisitos e estes mesmos implementados. Os novos requisitos tanto podem ser mudanças ao sistema antigo como novas funcionalidades. Assim, este é o fator mais importante para garantir a constante evolução de um sistema.



Pergunta **9**

Respondida Pontuou 3,000 de 5,000 Destacar pergunta

Software project management is concerned with activities to ensure that software is delivered on time, on budget, and in accordance with the requirements of the organisations developing and procuring the software. Poor people management is an important contributor to project failure.

Which of the following do you consider the worst practice to manage a software team?

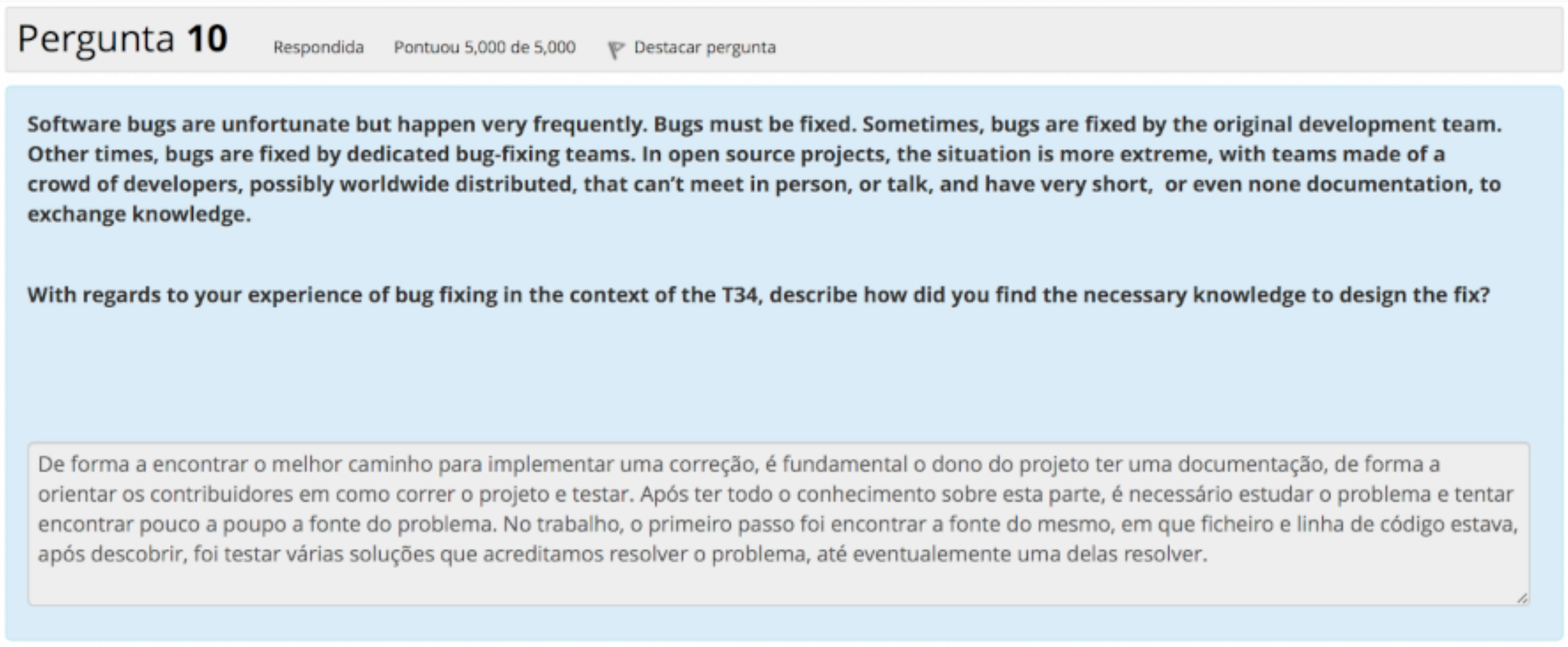
a) allow team elements to self-organize to decide who does what

b) have teams with elements having distinct backgrounds and technical skills

c) no team leaders, all team elements are equal

d) to recognize and positively discriminate individuals based on their individual performance

Resposta: C
O papel de team leader é fundamental, de forma a organizar, gerir e motivar a sua equipa.



Pergunta **10**

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Software bugs are unfortunate but happen very frequently. Bugs must be fixed. Sometimes, bugs are fixed by the original development team. Other times, bugs are fixed by dedicated bug-fixing teams. In open source projects, the situation is more extreme, with teams made of a crowd of developers, possibly worldwide distributed, that can't meet in person, or talk, and have very short, or even none documentation, to exchange knowledge.

With regards to your experience of bug fixing in the context of the T34, describe how did you find the necessary knowledge to design the fix?

De forma a encontrar o melhor caminho para implementar uma correção, é fundamental o dono do projeto ter uma documentação, de forma a orientar os contribuidores em como correr o projeto e testar. Após ter todo o conhecimento sobre esta parte, é necessário estudar o problema e tentar encontrar pouco a poupo a fonte do problema. No trabalho, o primeiro passo foi encontrar a fonte do mesmo, em que ficheiro e linha de código estava, após descobrir, foi testar várias soluções que acreditamos resolver o problema, até eventualemente uma delas resolver.