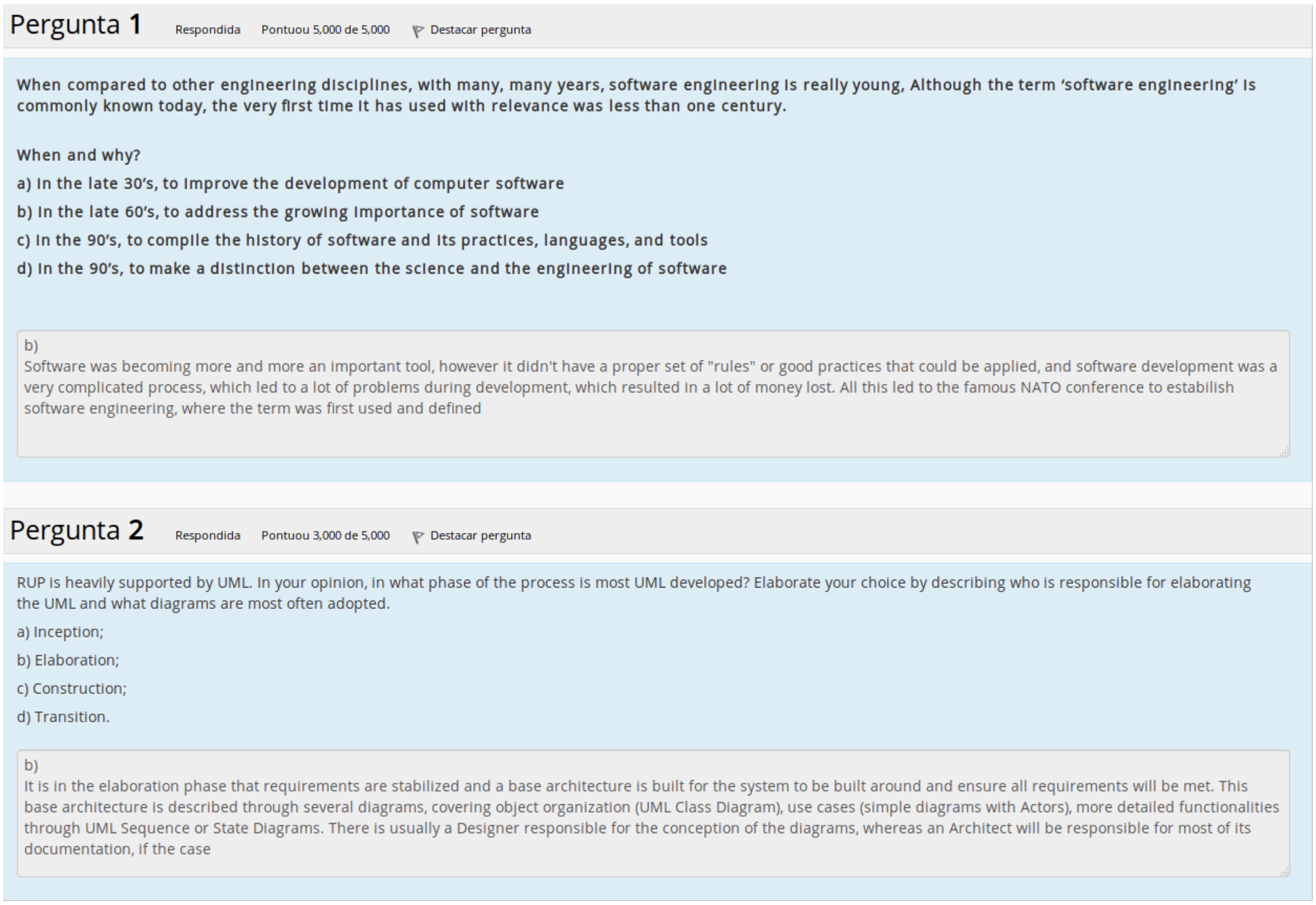


Pergunta 1

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

When compared to other engineering disciplines, with many, many years, software engineering is really young, Although the term 'software engineering' is commonly known today, the very first time it has used with relevance was less than one century.

When and why?

a) In the late 30's, to improve the development of computer software

b) In the late 60's, to address the growing importance of software

c) In the 90's, to compile the history of software and its practices, languages, and tools

d) In the 90's, to make a distinction between the science and the engineering of software

b)
Software was becoming more and more an important tool, however it didn't have a proper set of "rules" or good practices that could be applied, and software development was a very complicated process, which led to a lot of problems during development, which resulted in a lot of money lost. All this led to the famous NATO conference to estabilish software engineering, where the term was first used and defined

Pergunta 2

Respondida Pontuou 3,000 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.

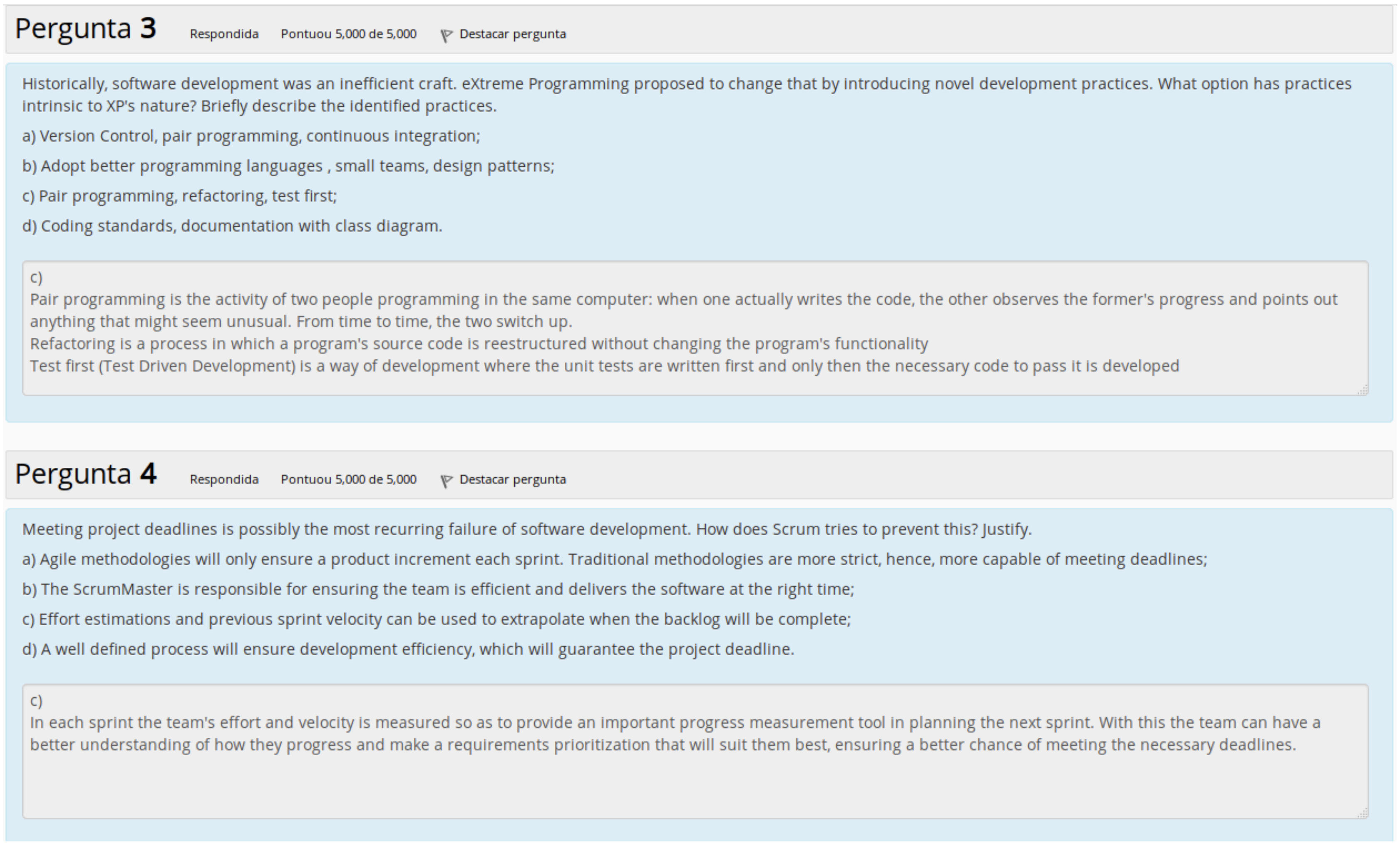
a) Inception;

b) Elaboration;

c) Construction;

d) Transition.

b)
It is in the elaboration phase that requirements are stabilized and a base architecture is built for the system to be built around and ensure all requirements will be met. This base architecture is described through several diagrams, covering object organization (UML Class Diagram), use cases (simple diagrams with Actors), more detailed functionalities through UML Sequence or State Diagrams. There is usually a Designer responsible for the conception of the diagrams, whereas an Architect will be responsible for most of its documentation, if the case



Pergunta 3

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Historically, software development was an inefficient craft. eXtreme Programming proposed to change that by introducing novel development practices. What option has practices intrinsic to XP's nature? Briefly describe the identified practices.

a) Version Control, pair programming, continuous integration;

b) Adopt better programming languages , small teams, design patterns;

c) Pair programming, refactoring, test first;

d) Coding standards, documentation with class diagram.

c)
Pair programming is the activity of two people programming in the same computer: when one actually writes the code, the other observes the former's progress and points out anything that might seem unusual. From time to time, the two switch up.
Refactoring is a process in which a program's source code is reestructured without changing the program's functionality
Test first (Test Driven Development) is a way of development where the unit tests are written first and only then the necessary code to pass it is developed

Pergunta 4

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Meeting project deadlines is possibly the most recurring failure of software development. How does Scrum tries to prevent this? Justify.

a) Agile methodologies will only ensure a product increment each sprint. Traditional methodologies are more strict, hence, more capable of meeting deadlines;

b) The ScrumMaster is responsible for ensuring the team is efficient and delivers the software at the right time;

c) Effort estimations and previous sprint velocity can be used to extrapolate when the backlog will be complete;

d) A well defined process will ensure development efficiency, which will guarantee the project deadline.

c)
In each sprint the team's effort and velocity is measured so as to provide an important progress measurement tool in planning the next sprint. With this the team can have a better understanding of how they progress and make a requirements prioritization that will suit them best, ensuring a better chance of meeting the necessary deadlines.

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.

c)
Requirements are one of the most important aspects of a project as they describe what the system should (not) do. Therefore it is important they are well defined, complete and consistent, hence a discipline dedicated to it and different kinds of requirements defined (as stated in b), however non-functional might be more critical since that if they fail, the whole system may not work). Agile projects consider the importance of requirements along all the process and in the modern days requirements are in constant change and the project (and its team) must adapt frequently to (sometimes drastic) changes.

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.

d)
Here we are using functionalities already developed and compiled into a library (here, stdio.h) that allow us, in this case, to write a message to the screen. If we were to do this ourselves, from scratch, the code would have been much bigger.

Pergunta **7**

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Software maintenance prediction and change prediction are useful because:

a) At the release of the system, financial measures can be taken to ensure that an effective maintenance team can be hired.

b) To collect statistics on how the system will perform in terms of maintainability.

c) As software ages and becomes "legacy", if maintaining the system remains feasible and cost-effective.

d) It serves to prove (or not) that maintenance high cost preventive measures were applied during development.

c)
Maintenance is an important aspect in software development as maintaning software can be even more expensive than developing new software. Therefore, maintenance and change prediction are important tools to determine if a given software, as it ages, is worth maintaning or should just be totally replaced.

Pergunta **8**

Respondida Pontuou 5,000 de 5,000 Destacar pergunta

Test-driven development promotes smaller, terse and more balanced codebases. What activities/benefits do you think contribute the most to this outcome?

a) "Test-first" and refactoring.

b) Regression testing and automated tests.

c) Code coverage and system documentation.

d) Automated Tests as acceptance criteria and regression testing.

a)
The test-first methodology helps ensure the developer will create only the necessary code to pass the tests. This way there's a smaller chance the programmer will make unecessary code and, thus, make a larger codebase. With this in mind, at each new test the code addition should minimal, so refactoring is encouraged as a way to, at each step, review what has been done and possibly reestructure in a better way according to well known refactor techniques, contributing again to a smaller and more balanced codebase.

Pergunta 9

Respondida Pontuou 0,000 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. One of the activities of project management is planning the software releases: which features to deliver, and when.

Which of the following do you consider the best practice to plan a software project?

a) the team elements self-organize to select the features to include in the next release

b) the developers estimate the time to implement a set of features

c) the project manager defines the features and the effort required to implement the releases

d) the team defines what to implement in the next release and the project defines how to implement it

a)
It is important that all team elements have a voice in what features should be implemented and when should they be implemented as they will be the ones doing it and are more aware of their capabilities than anyone else. It is also a good way to prioritize said features so that, in case of an unexpected problem, at least some kind of working program can be delivered.

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 validate the fix?

Fortunately, my class' chosen project had a very active Slack community, so it was relatively easy to estabilish contact with the main core contributors of the project and get continued tips and feedback for our issues' development. On the other hand, in our case both issues were not bugs themselves, but feature requests, so there wasn't really a "fix". As soon as the features were developed, we tested what we thought to be the essencial use cases (and also extrapolated some from user feedback in the issues chosen, when the case) of said features and, obtaining a positive result, created the PR.