Which one of the following is most adequate to have a software prototype?

a) A customer-driven app for ordering food; b) A business-to-business software for supporting a supply chain; c) A healthcare software to handle critical data; d) All of the above.

Resposta: d) All projects should have a prototype, an initial version of a system that shows how the final result will be, so it can help verifying requirements already implemented and others that can be implemented in final version. As it is an initial version, with some restrictions and low costs, the developers don't lose too much time developing this prototypes as they spend on final product.

Being a framework, RUP is highly customizable to meet a project's needs. When adopting RUP, the implementing team must decide how to fit the framework's moving parts to best suit the project. Which of the items below could be characteristic of an Agile implementation of RUP?

a) Keeping the iterations long, adapting the documentation size and having the client validate each product increment; b) Adopt test-driven development and pair programming; c) Work with the client to fully specify the requirements during the first phase; d) Despite being a framework, it is not possible to instantiate RUP with the Agile values.

Resposta: b) An Agile implementation of RUP should take in count some of the specifications of this process. As test-driven-development is one of this characteristics, this implementation should include that. Pair programming is another Agile characteristic that should be adopted. Pair programming is the activity of two people programming in the same computer: while one writes the code, the other analyses that and criticizes, in order to have a better code in the final. Iterations of an Agile method should be short and it is possible change requirements during the project.

Requirements are often captured as User Stories. Who is responsible for writing the user stories? Provide a good example of a user story.

a) The whole team; b) The product owner; c) The scrum team; d) The development team.

Resposta: User stories should be written by all team in order to all team members work together and know better project specifications and functionalities. An example of a user story is: As a vacation traveler, I want to see photos of the hotels.

Which of these is not a good reason for keeping user stories small? Justify your answer.

a) so that they are easier to estimate; b) so that they can be delivered quickly; c) so that only one team member is needed for each user story; d) so that is easy to test them and decide if they are really done.

Resposta: The project and each feature should be developed by all team in order to reduce the errors. So, attribute a user story to only one team member is not one of small user stories objectives.

Regarding use case modelling, include relationships aim to:

a) Represent optional behavior b) Represent critical behavior c) Represent mandatory behavior d) Represent behavior imported from other system

Resposta: Include relationships are used when many use cases share some common behavior. That common behavior can be separated and described in a new use case which is included by the first ones. Inclusion is mandatory.

Open Source software that can be freely accessed, used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the Open Source Definition. Which of the following statement is false?

a) It is illegal to build and sell software that reuses open-source libraries; b) Open-source software can be used for personal financial reasons; c) A software is only open-source if distributed with an open-source license; d) Open-source licenses can be used to while creating commercial software.

Resposta: It is possible to build and sell software that reuses open-source components. Depending on the license that the software is published, the users have different rights and duties. For example, using a GNU GPL license, to share software based on this license, it's mandatory that it is distributed using the same license.

Software project management is concerned with activities to ensure that the software is delivered on time, on budget, and in accordance with the requirements of the organizations developing and procuring the software. Poor people management is an important contribute to project failure. Which of the following do you consider the worst practice to manage a software team? Justify your answer.

a) allow team elements to self-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: Recognize individual work tends to make each developer to work alone ("Heroic development"), which makes the projects depending on each member performance, growing up the number of errors on project.

The following is "not" a type of software maintenance:

a) Modifying the system to satisfy new requirements; b) Restarting the whole software system from scratch; c) Maintenance to adapt software to a different operating environment; d) Changing a system to fix bugs/ vulnerabilities in the way meets its requirements.

Resposta: Adding new features to the project, as said in a), is a type of software maintenance. Adapt software to new operating environment is another type of software maintenance. Correct software bugs and vulnerabilities is another way to maintain software. So, we can conclude that option b) isn't a type of software maintenance, as make a new software from scratch doesn't take in count the previous software.

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: Sometimes it's hard to understand what the other has done, due to how abstract it sometimes is.

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: During this phase, the problem domain is analysed and it's defined a stable and robust architecture for all system, taking in count its requisites. While analyzing, the main result is the analyse model (classes and ideal collaborations), and in design the main result is the project model (implementation classes grouped by sub-systems, necessary collaborations to realize the use cases) and the distribution model.

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?

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

a) Component level; b) Abstraction level; c) Design level; d) Object level

Resposta: 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.

Software bugs are unfortunate but happen very frequent. 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.

Resposta: In order to find the best way to implement a correction, it is essential that the project owner has a good documentation, so the contributors can easily know how to run the project, test and how it was done. After knowing all this, is is necessary to study the problem and try to find the problem source. Finally, after find the lines that have errors, the developers have to try different solutions until they can fix the bug.

Briefly describe a famous software failure, its causes, consequences and lessons learned.

Resposta: A famous software failure was related to the Therac-25, caused by a software error, more specifically a race condition. Because of it, huge amounts of radiation were emitted onto patients, and at least 5 died as a result. In this specific problem, a main issue was the developers' over.confidence since they didn't properly test the device's features. So, a lesson to be learned, besides being humble, is to always test the project before launching an executable iteration, by using a complete test suite of for example, integration, and/ or system tests.

Describe the role played by git branches in open source software development.

Resposta: Imagine a situation where you have to fix 2 bugs. In this situation, the best practice is to use git branches, each one representing a fix. So, starting from master branch, create a new branch for each fix and then, when they are fixed, create two pull requests in order to merge the new code onto the master branch of the main repository. In other contexts, a branch can be used to represent a new feature that can also later be merged onto the master branch of the repository. This can be used to protect the master branch in order to always have working increments.

Describe ways found and used by open source teams to encourage and support new contributors.

Resposta: Most large open-source projects have a labeling system on their Issues tab (some of them are created by GitHub, and others by the project maintainers). For example, in order to choose the first issues, start searching for issues labeled as "Good first issue' or 'Help wanted'. The project contributors are also very active and quickly answer on the issues' threads in order to clarify what should be done, and give code pointers. There's also README files and wikis, that explain how to contribute, build, test, and run the application, so that users can easily test their fix/ features in order to create a pull request.