QUESTIONS FROM ESOF EXAMS

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

1. In the late 30's, to improve the development of computer software
2. In the late 60's, to address the gorwing importance of software
3. In the 90's, to compile the history of software and its practices, languages, and tools
4. In the 90's, to make a distinction between the science and the engineering of software

R.: 2. Software was becoming more and more and 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 establish software engineering, where the term was first used and defined.

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.

1. Version Control, pair programming, continuous integration;
2. Adopt better programming languages, small teams, design patterns;
3. Pair programming, refactoring, test first;
4. Coding standards, documentation with class diagram.

R.: 3. 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 restructured 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 them is developed.

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

1. Agile methodologies will only ensure a product increment each sprint. Traditional methodologies are more strict, hence, more capable of meeting deadlines;
2. The ScrumMaster is responsible for ensuring the team is efficient and delivers the software at the right time;

3. Effort estimations and previous sprint velocity can be used to extrapolate when the backlog will be complete;
4. A well-defined process will ensure development efficiency, which will guarantee the project deadline.

R.: 3. In each sprint the team's effort and velocity is measured 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.

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.

1. Requirements engineering comprises several activities, from elicitation to validation, being this last one the easier to perform.
2. 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.
3. 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.
4. 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.

R.: 3. 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 *2*), 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.

Software maintenance prediction and change prediction are useful because:

1. At the release of the system, financial measures can be taken to ensure that an effective maintenance team can be hired.
2. To collect statistics on how the system will perform in terms of maintainability.
3. As software ages and becomes "legacy", if maintaining the system remains feasible and cost-effective.
4. It serves to prove (or not) that maintenance high cost preventive measures were applied during development.

R.: 3. Maintenance is an important aspect in software development as maintaining 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 maintaining or should just be totally replaced.

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

1. "Test-first" and refactoring.
2. Regression testing and automated tests.
3. Code coverage and system documentation.
4. Automated Tests as acceptance criteria and regression testing.

R.: 1. 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 unnecessary code and, thus, make a larger codebase. With this in mind, at each new test the code addition should be minimal, so refactoring is encouraged as a way to, at each step, review what has been done and possibly restructure in a better way according to well known refactor techniques, contributing again to a smaller and more balanced codebase.

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 organizations 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?

1. the team elements self-organize to select the features to include in the next release
2. the developers estimate the time to implement a set of features
3. the project manager defines the features and the effort required to implement the releases
4. the team defines what to implement in the next release and the project defines how to implement it

R.: 2. By doing that the manager can more accurately estimate the required time for the next release and plan out with the team the feasibility of said features.

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?

1. The development process used to construct the system.
2. The change proposals.

3. The existing development documentation.

4. The evolution team.

R.: 3. Since projects tend to be developed using methodologies such as agile methods that tend to minimize documentation, it gets harder with each change to understand how the software is organized and structured, making it harder to evolve.

Acceptance testing is a user testing process where the aim is to decide if the software is good enough to be deployed and used in its operation environment. But doing acceptance testing is not easy and straightforward, mainly because:

1. There is no automation.

2. It can only to be run after the whole system is developed and installed.

3. It must cover the needs of all stakeholders.

4. It doesn't cover usability issues.

R.: 3. Since the user/customer works with the development team to define the tests for the integration with the development, it is hard to know if the user can cover the needs of all the stakeholders.

RUP is composed by several engineering disciplines. Analysis and design is one of those disciplines. What roles are required by this discipline? Elaborate what they use as input and what is the output of their work.

1. The whole development team;

2. Lead developer and clients;

3. Requirements engineers and product manager;

4. Software architects and designers.

R.: 4. The architects perform an architectural analysis, that is then reviewed by the designers. The architects then develop the architectural design, describe concurrency and distribution, and later review their work before sending it back to the design team; the design team then implements the subsystem and class design, as well as the use case design. All work is reviewed.

The Agile manifesto defends:

```
 Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
```

In your opinion, what did the authors wanted to achieve by introducing the manifesto? Justify your option.

1. A faster way to develop software, by discarding the need for documentation requirements and source code;
2. A more efficient software development, by adopting novel development architectures and tools;
3. A cheaper way to develop software, by eliminating overheads and unnecessary roles;
4. A less bureaucratic approach to develop software by bringing clients and development team to work better and together.

R.: 4. Agile methods focus more on people than on processes and tools. With a customer viewing the delivery of increments in each iteration, he can get and provide feedback towards the product work. This creates trust between the clients and the developers, which forms a positive culture where both sides expect the project to succeed. This is possible due to the less formal approach of the agile methods.

The Rational Unified Process was introduced in the early 2000s. Choose the best description for RUP and shortly elaborate about it in your words.

1. An iteractive software development process framework;
2. An iteractive prescriptive development process;
3. A set of guidelines for using UML to capture requirements and interact with clients;
4. A framework for capturing and monitoring the implementation of software requirements.

R.: 1. RUP is a software development process that is divided into cycles, phases and iterations. The functionality of the system is delivered after a series of releases of increasing completeness, in each iteration, different requirements are selected to be developed, providing new releases with every iteration. Therefore, we can conclude that RUP is an interactive development process framework.

What is the main cause for software degradation?

1. Maintenance.
2. Lack of Re-engineering.
3. Poor Refactoring.
4. Low Technical Debt.

R.: 3. Bad refactoring lead unmaintainable code base, code changes and added new features with no refactoring may lead to code smells, code duplication and bad software design.

---

eXtreme Programming is praised by its software development efficiency. Pair programming is one of the practices contributing to said efficiency. What does pair programming consists of? Briefly detail the practice using your own words.

1. Two developers working on their own features, byt assigned to help each other;
2. One developer working on part of a feature, while the other tackles the other part, merging their implementation afterwards;
3. One developer working on a feature, while being continuously reviewed by another, often trading their roles;
4. Two developers independently working on the same feature, discussing the two implementations and choosing the best solution.

R.: 3. As stated above, pair programming is a software development practice, where two developers work with only one computer, trying to minimize human error, increase productivity and maximize code-base knowledge.

Requirements elicitation and analysis include discovery, organization, prioritization, and specification od requirements. It seems a paradox, but it is common to say that stakeholders don't know what they really want, or don't know how to express it, and usually do not agree, creating requirements conflicts. Which of the following practices do you consider to be important to do for an effective requirements elicitation?

1. Specify the requirements in the most natural way as possible, to use the same language of the stakeholders.
2. Specify the requirements in the most structured format as possible, to eliminate all conflicts.
3. Specify the requirements as use cases, or user stories, to illustrate all possible interactions with the system.
4. Specify the requirements at the beginning of the project only, to avoid later changes.

R.: 3. Establishing the system requirements involve technical staff working with customers or other stakeholders to find which services/functionalities should the system provide, the application domain, the work environment and operational constraints. Since it involves such a large variety of people it is better to specify the requirements with use case models, user stories since stakeholders can relate to them and comment on their situation with respect to the story/case. This will avoid conflicts provided by the language or technical formats not commonly used by costumers/stakeholders.

What of the following IS NOT a software engineering knowledge area according to the Guide to the Software Engineering Body of Knowledge (SWEBOK)?

- Software Design
- Software Requirements

- Software Testing
- **User Experience Design**

Which of the following IS NOT a common activity in all software development processes?

- Design and implementation
- Validation
- Specification
- **Round-trip engineering**

The distinction between plan-driven and agile processes is that:

- In plan-driven processes, plans can't be changed (contrarily to agile)
- In agile processes, there is no planning
- **In agile processes, planning is incremental (contrarily to plan-driven processes, that do all the planning up-front)**
- In plan-driven processes, plans are documented (contrarily to agile processes)

The waterfall model is best suited for what kind of projects?

- For projects with significant uncertainties in requirements and/or technologies
- It is not suited for any project
- **For large, multi-site, projects, with stable requirements**
- For projects with frequent requirements changes

"Systematic reuse where systems are assembled from existing components or COTS (Component-Off-The-Shelf) systems" describes what process model?

- The "software prototyping" process model
- The "waterfall" process model
- **The "integration and configuration" process model**
- The "incremental development" process model

Which of the following IS NOT an advantage of incremental development?

- System functionality is available earlier
- The cost of accommodating changing requirements is reduced
- **System structure tends to improve as increments are added**
- More frequent & early customer feedback reduces failure risk

---

Which of the following best characterizes the RUP organization?

- A project is decomposed into several iterations, each wich can be decomposed into 4 phases (Inception, Elaboration, Construction and Transition).
- A project is decomposed into 6 phases (business modeling, requirements, analysis & design, implementation, test, deployment), each of which can be decomposed into several iterations.
- A project is decomposed into 4 phases (Inception, Elaboration, Construction and Transition), which can run in parallel.
- **A project is decomposed into 4 phases (Inception, Elaboration, Construction and Transition), each of which can be decomposed into several iterations.**

Which of the following IS NOT a XP practice?

- Test-driven Development
- **Design by Contract**
- Continuous Integration
- Pair Programming

Which of the following IS NOT a Scrum event?

- **Sprint kick-off meeting**
- Sprint retrospective
- Sprint review
- Sprint planning meeting

In Scrum, which of the following IS NOT a responsibility of the Product Owner?

- Prioritize the user stories
- Accept or reject work results
- Define the user stories of the product
- **Estimate user stories in story points**

In Scrum, what does the team velocity represent?

- The number of lines of code written by the team per sprint
- **The total number of story points (of user stories) implemented per sprint**
- The number of user stories implemented per sprint
- The number of hours worked by the team per sprint

Which of the following is NOT a requirements engineering activity?

- requirements specification
- requirements validation
- requirements elicitation
- **requirements implementation**

Which of the following techniques is most useful to discover new and innovative requirements?

- Prototyping
- Social observation and analysis
- Interviews
- **Brainstorming**

Which of the following is NOT a valid element or relationship in UML use case diagrams?

- **message**
- <>
- actor
- use case

Which of the following is NOT a valid element or relationship in a domain model (represented by a UML class diagram)?

- attribute
- **actor**
- class
- contraint

The focus of architectural design is to partition a system into a set of:

- classes
- data structures
- **components**
- use cases

Which of the following is NOT a view in the 4+1 view model of software architecture?

- Implementation View
- Logical View
- **Maintenance View**
- Deployment View

Which UML diagrams is best suited to show the system hardware and how software components are distributed by hardware nodes?

- **Deployment diagram**
- Component diagram
- Distribution diagram
- Communication diagram

Select the most appropriate architectural pattern for building each type of system.

- Distributed applications: **Client-server and n-tier systems**
- Interactive systems: **Model-View-Controller**
- Batch processing systems: **Pipes and filters**
- Operating systems: **Layered architecture**

Check if we are **building the right product** and **building the product right** refer respectively to

- Testing and Reviews
- **Validation and Verification**
- Verification and Validation
- Reviews and Testing

Which of the following is NOT a type of software review?

- Team inspection
- **Personal audit**
- Peer review
- Personal review

Which of the following is NOT a test level or phase?

- integration testing
- system testing
- **black box testing**
- unit testing

Retesting a system or component to check that modifications didn't cause unintended effects is

- Defect testing
- **Regression testing**
- Reliability testing
- Smoke testing