# SE2 - THEORY QUESTIONS – TELEGRAM

1. Differences between agile and waterfall

A — Agile is an incremental and iterative approach that divides the project into sprints each ending with the release of a working part of the software, allowing changes to the requirements at any time, focusing more on reacting to the changes of the customer needs.
Waterfall is a linear and sequential approach that divides the project into phases, defining precisely the requirements at the start and ends with only one release of the completed project; scope changes are avoided once the project started. In agile a project manager is not required, while in waterfall plays a key role in every phase.

2. Scrum - Describe estimation Poker

A — The game starts with a set of cards with different values expressed as "sizes" (small, medium, large, extra large) or through numeric values (e.g. values of the Fibonacci series). Every team member picks a card that thinks fits the story under estimation (in terms of effort needed to complete it) and shows it at the same time as the others; higher and lower estimations discuss the rationale and the estimation is done again until an agreement is reached. Eventually consensus is achieved with feedback from the product owner.

3. Scrum - Describe what is a Story and its template

A — A story is an essential description of a desired functionality defined by the product owner. It should be "DEEP" (Detailed, Estimated, Emergent, Prioritized) and as a Product Backlog Item should be characterized by the INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable) and easy to turn into automated tests. The template is: 'As a <actor type> I want <to do something> So that <some value is created>'.

4. Scrum - Describe two characteristics of INVEST for a story in scrum

A — INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable)
• Independent: it has to be implementable independently from the realization of other user stories
• Negotiable: there shouldn’t be a specific contract for features
• Valuable: it should increase the business value
• Estimable: it should be estimable in terms of effort up to a good approximation
• Small: it should fit into a sprint
• Testable: it should be possible to realize automated tests to verify it’s doneness.

5. Scrum - Describe team on scrum

A — In scrum, the team is self-organizing (but assisted at the start by the Scrum Master that steps back increasingly with time) and has a size of 5 to 9 members. There should be a good skill variety among the individuals, that cooperate and are willing to work outside their comfort zone to make the team achieve the goal.

6. Scrum - describe what is a sprint and its characteristics

A — A Sprint is the basic iteration in the scrum approach, it has a length that can vary from a couple of weeks to a couple of months (shorter times are preferred). It starts with the sprint planning done by the team and it produces a piece of working software to be demonstrated and reviewed at the end.

7. Scrum - what is a task board and the elements inside it

A — a task board is a surface divided into sections (usually 3, being "to-do", "in progress" and "done"), where the team can place the tasks, representing visually the progress of the project in the current sprint. It can be realized either through memos over a flat surface or by using dedicated tracking softwares like YouTrack.

8. Scrum - Extreme programming and its main characteristics, for what projects it is suggested

A — Extreme programming is an agile methodology that takes to the extreme what in the agile methodology is considered as best practice: test-driven development, weekly or daily system-wide integration testing, pair programming, coding always the simplest and fastest solution (YAGNI principles "You Aren't Gonna Need It") to avoid investing effort in something that could become useless with the next change of requirements. It is suggested for small-medium projects that need to adapt fast to frequent requirement changes, like any agile methodology.

9. Scrum - Describe what is a story and how it is used in the scrum context

A — A story is an essential description of a desired functionality defined by the product owner. It should be "DEEP" (Detailed, Estimated, Emergent, Prioritized) and as a Product Backlog Item should be characterized by the INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable) and easy to turn into automated tests.

10. Scrum - What is daily scrum?

A — It's a meeting involving only the team members that is carried out daily at any time suitable for the team. In order to keep it brief it's performed standing up and has a maximum duration of 15 minutes and it's focused on three points: what has been done, what will be done and the current obstacles.

11. Scrum - Describe briefly Scrum process

A — The scrum process starts with the Product owner generating the Product Backlog in form of a ranked list of stories. Then the team in a sprint planning meeting selects from the top ranked stories as much as it can commit to deliver by the end of the sprint, producing the Sprint Backlog by dividing each story in tasks. Tasks are carried out during the sprint and all the team keeps up to date with the state of the project with daily scrum meetings. At the end of the sprint the team presents the produced working software in a sprint review to get feedback from the stakeholders. The team then reflects on how to improve it's own effectiveness in a sprint retrospective. Then it cycles again starting from the sprint planning, selecting the stories from the top of the ranked list of the Product Backlog that could have been changed by the product owner in the meanwhile.

12. Technical Debt - Describe the relationship between Software Quality and Software Engineering Code of Ethics

A — IEEE Code of Ethics reports as point 3 the duty to strive for high quality, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.

13. Technical Debt - describe a code smell and write an example

A — In the context of object oriented programming, code smells are methods and classes that violate the principles of good object-oriented design: clearly defined single responsibility, encapsulation, information hiding, few and clear interfaces, proper use of inheritance.
Code smells are categorized in three types: identity disharmonies, collaboration disharmonies, classification disharmonies.
To answer to the original question one could bring as an example
- an identity disharmony related to the proportion rule: e.g. one enormous class completely unproportioned to the others that centralizes logic and uses data of other classes. (a "God Class")
- a collaboration disharmony of a class where methods have an intensive coupling (methods calls too many methods from few unrelated classes)
- a classification disharmony of a class being a tradition breaker (provides a large set of services which are unrelated to those provided by its base class, does not specialize inherited services).

14. Technical Debt - Describe the Technical Debt metaphor and make an example

A — The Ward Cunningham explanation: with borrowed money you can do something sooner than you might otherwise, but then until you pay back that money you'll be paying interest. In the metaphore, "borrowing money" stands for "writing code to reflect your current understanding of a problem even if that understanding is partial", "pay back" stands for "refactoring the code later in order to make it reflect your new and more complete understanding of that problem". The debt is the cumulative set of changes that have to be applied to the software in order to pay back all the money. If you never pay it back, at a certain point the software structure and behavior will differ from the intended one so much that it will be impossible to make it progress.

15. Agile Contracts - Differences in contracts between Agile and traditional approaches

A — In traditional approaches there is a fixed scope of the project (lack of flexibility) with a variable time and cost of completion. Agile contracts go in the opposite direction fixing time and cost while leaving space to changes in the scope of the project.

16. Agile Contracts - Describe agile principles

A — 1. Individuals and interactions over processes and tools: building projects around motivated individual trusting them to get the job done, conveying information to and within a development team through face-to-face conversation, having self-organizing teams that reflects on how to become more effective at regular intervals, valuing individuals and interactions over processes and tools.
2. Working software over comprehensive documentation: simplicity, technical excellence and good design, having working software as the primary measure of progress, valuing it more than comprehensive documentation.
3. Customer collaboration over contract negotiation: business people and developers must work together daily to the project, maintaining a constant pace of development sustainable for sponsors, developers and users.
4. Responding to change over following a plan: welcome changing requirements even late in development to meet customer's changing needs and enhance its competitiveness, early and continuous delivery of valuable working software with a frequency varying from couples of weeks to couples of months.

17. Agile Contracts - Select a pricing scheme, describe it and describe the situations in which it is good

A — > *Flexibility levels*:

• Fixed Price Fixed Scope (FPFS): Price and Scope are fixed and this is optimal for managers to "order and forget" (so it's also easier to get the work assigned as suppliers), it's optimal for sales person so that it knows how much he'll get out of the sale, for customer that has a budget to respect and for supplier that knows in advance how much money they'll receive. Unfortunately this usually means accumulating technical debt due to low margins, especially with fixed time. Not adapt to agile working.

• Variable Price Variable Scope (VPVS): cap over a pricing scheme, non-binding release backlog allowing for flexibility, making it the starting point of common vision of the project between Product owner and team, making it a base for estimating the iteration price cap

> *Pricing schemes*:

• Lump sum/Fixed price: pay for the full product, used in case of Fixed Scope.

• Time and Materials (TM): pay for the materials needed to produce the software and for the time of the people working on it.

• Fixed price per iteration: pay for each iteration of the agile process.

• Fixed price per Unit of Work: where Unit of Work is a certain amount of software measured in some way.

• Hybrid (shared pain/gain): fixed price within a certain range, the supplier gets a bit more if it delivers the full product earlier than estimated or a bit less if it delivers it in more time.

• Pay per use: the customer pays until it stops using the system.

POSSIBLE ANSWERS:

• Best one for Agile is TM: cost of each sprint can vary in practice, because team composition can vary and people in the team might be also assigned to other projects, resulting in variable effort investments from sprint to sprint. To mitigate this variability TM could be capped per iteration/project/release so that the customer knows it will not spend more than a certain amount of money per iteration. Useful with medium-small companies that have more flexibility.

• Best one when the customer has very high flexibility or no predefined requirements is Fixed Price per iteration: Supplier must estimate correctly to be able to fix a price

• Unit of Work: is tipically running and tested features, possibly measured with standards like function points and then associating a price to each function point to get the total price of a feature. Presents as issue the non-correlation between the effort (and money) needed to complete a feature (Unit of Work) and the business value of that feature, so the customer pays few money for important features and a lot for the non important ones.

18. Agile Contracts - describe one value from the agile manifesto

A — Simplicity - the art of maximizing the amount of work not done - is essential: this means that effort should be spent wisely, always realizing only what is needed when is needed in order to avoid losing time in something that could become useless in the future with eventual changes in the requirements.

19. Measurement - Product quality for the ISO model

A — Product quality model of standard ISO 2501 lists as quality characteristics: Performance efficiency, Interaction capability, Security, Flexibility, Functional suitability, Compatibility, Reliability, Maintainability

20. Measurement - Describe Goodhart’s law

A — Goodhart’s Law: "When a measure becomes a target, it ceases to be a good measure". This means that when the product of the work of someone it’s going to be evaluated with some specific characteristics, those characteristics will become so important to the supplier that he’ll focus primarily on that instead of focusing on the overall product quality. e.g. if you measure the work of a programmer by number of lines of code, that programmer might focus on generating extremely redundant and useless code instead of good quality and useful code.

21. Measurement - differences between construct and measure and an example in context of software quality

A — A Construct is a broad abstract concept that could have many complex aspects and is usually not directly observable, a Measure instead has a precise definition and is produced through a clear measurement procedure. To a construct could correspond more than one measure. Examples of measure in the context of software quality are: Defect Density and Mean Time Between Failures (MTBF)

*Questions about the team*

23. describe how you keep track of progress in your team and how you could improve in a future project
24. describe how you keep track of progress in your team and how you could improve in a future project
25. How have you organized the Sprint planning and how is it possible to improve it?
26. How test where conducted, what can be improved
27. describe how test where conducted and what can be improved
28. how we did the estimation for the project and how we can improve it
29. how you applied the definition of done
30. describe the estimation mechanism in your group and how you could improve it
31. how you managed sprint planning and how to improve
32. describe the process you followed for technical debt and how it can be improved
33. Which process did you follow to manage technical debt and how can you improve it?
34. describe the process you adopted when a bug was found
35. describe how you did the sprint planning and how could you improve it
36. how you managed sprint planning and how to improve
37. How do you fixed bug and how to improve
38. how the tests were managed in the team and how they could be improved
39. Process followed to manage TD and possible improvements
40. How have you organized the Sprint planning and how is it possible to improve it?
41. how you implemented the definition of done and how to improve it
42. how did you track your team's progress and what will you do to improve it in future projects
43. How did your team manage technical tasks and what would you do better in a future project
44. how the tests were managed and how they could be improved
45. what are the main deviations you did from scrum process and why?
46. synthesis your role in project, and what would you do to get better in future?
47. Which process did you follow to manage technical debts