CO328: Agile development
Tutorial 01

E/15/123
E/15/280
E/15/316

1. What is meant by impediments in Scrum? Explain with examples.

An Impediment is anything that keeps the team from getting work done and that slows velocity. Impediments can be in many forms like

- a sick team member
- a missing resource
- lack of management support or even a cold team room.

If it's blocking the team from doing its work, it's an Impediment.

2. Explain the difference and similarity between SCRUM and AGILE?

Scrum methodology is an Agile framework that facilitates collaboration and efficiency in software development and testing. But while both involve incremental builds for projects, they also have their differences.

- Scrum is a more rigid method with less flexibility for change, and it’s ideal for those who need to produce results as quickly as possible.
- Agile is a continuous iteration of development and testing in the software development process whereas Scrum is an Agile process to focus on delivering the business value in the shortest time.
- Agile is more suited for smaller teams and for those who prefer a more straightforward design and execution, while Scrum is used more for creative and experimental approaches.
- The agile methodology delivers the software on a regular basis for feedback while Scrum delivers the software after each sprint.
- Agile involves collaborations and face-to-face interactions between the members of various cross-functional teams whereas Scrum collaboration is achieved in daily stand up meetings.
- In Agile process design and execution should be kept simple whereas in Scrum process design and execution can be innovative and experimental.
- In the Agile process, leadership plays a vital role; on the other hand, Scrum fosters a self-organizing, cross-functional team.

Scrum is always Agile, but Agile is not always Scrum. This means Scrum will encompass the same methodologies of Agile, but Agile may not share some of the same qualities as Scrum.

3. Explain what is meant by Daily Stand-Up?

A stand-up meeting is a meeting in which attendees typically participate while standing. The discomfort of standing for long periods is intended to keep the meetings short. It is a commitment and coordination meeting for the entire team and designed to ensure that the entire team is aware of impediments, what stories are done or not done, and what tasks are ready to be pulled from one team member's to-do list into someone else's.

4. Explain what is meant by Velocity?

At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity. It is calculated at the end of the Sprint by summing up the Points for all fully completed User Stories. But, points from partially-completed or incomplete stories should not be counted in calculating velocity.

Knowing velocity, the team can compute (or revise) an estimate of how long the project will take to complete, how much scope can be delivered by a specific date, based on the estimates associated with remaining user stories and assuming that velocity over the remaining iterations will remain approximately the same. It helps in predicting a date for a fixed amount of scope to be delivered. Velocity helps in understanding our limits while defining the amount of scope we will commit for a sprint. This is generally an accurate prediction, even though rarely a precise one.

5. Explain what is meant by increment.

Product Increment is one of the important deliverables or artifacts of Scrum. Product Increment is the integration of all the completed list of Product Backlog items during the sprint. As the name suggests, Product Increment goes on getting incremented in the subsequent sprints. So, in a particular sprint, the Product increment is the integration of all the completed list of Product Backlog Items whereas, in a Project, Product Increment is the integration of all the completed list of Sprint backlog items. With each sprint, the product increment increases in terms of delivered functionality.

6. Briefly explain the Agile Manifesto & its Principles.

The four core values of Agile software development are stated by the Agile Manifesto.

- Individuals and Interactions Over Processes and Tools

Valuing people more highly than processes or tools is easy to understand because it is the people who respond to business needs and drive the development process. If the process or the tools drive development, the team is less responsive to change and less likely to meet customer needs.

- Working Software Over Comprehensive Documentation

Earlier, enormous amounts of time were spent on documenting the product for development and ultimate delivery. Technical specifications, technical requirements, technical prospectus, interface design documents, test plans, documentation plans, and approvals required for each. The list was extensive and was a cause for the long delays in development. Agile documents requirements as user stories, which are sufficient for a software developer to begin the task of building a new function. The Agile Manifesto values documentation, but it values working software more.

- Customer Collaboration Over Contract Negotiation

Negotiation is the period when the customer and the product manager work out the details of delivery, with points along the way where the details may be renegotiated. Collaboration is a different creature entirely. With development models such as Waterfall, customers negotiate the requirements for the product, often in great detail, prior to any work starts. This meant the customer was involved in the process of development before development began and after it was completed, but not during the process.

- Responding to Change Over Following a Plan

Traditional software development regarded change as an expense, so it was to be avoided. The intention was to develop detailed, elaborate plans, with a defined set of features and with everything, generally, having as high a priority as everything else, and with a large number of many dependencies on delivering in a certain order so that the team can work on the next piece of the puzzle.

12 principles are articulated in the Agile Manifesto.

1. Customer satisfaction through early and continuous software delivery – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.
2. Accommodate changing requirements throughout the development process – The ability to avoid delays when a requirement or feature request changes.
3. Frequent delivery of working software – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.
4. Collaboration between the business stakeholders and developers throughout the project – Better decisions are made when the business and technical team are aligned.
5. Support, trust, and motivate the people involved – Motivated teams are more likely to deliver their best work than unhappy teams.
6. Enable face-to-face interactions – Communication is more successful when development teams are co-located.
7. Working software is the primary measure of progress – Delivering functional software to the customer is the ultimate factor that measures progress.

8. Agile processes to support a consistent development pace – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
9. Attention to technical detail and design enhances agility – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
10. Simplicity – Develop just enough to get the job done for right now.
11. Self-organizing teams encourage great architectures, requirements, and designs – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.
12. Regular reflections on how to become more effective – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

7. Explain the drawbacks of the Agile model.

- Less predictability.

For some software deliverables, developers cannot quantify the full extent of required efforts. This is especially true at the beginning of the development life cycle on larger products. Teams new to the agile methodology fear these unknowns. This fear drives frustration, poor practices, and often poor decisions. The more regimented, waterfall process makes it easy to quantify the effort, time, and cost of delivering the final product.

- More time and commitment.

Testers, customers, and developers must constantly interact with each other. This involves numerous face-to-face conversations, as they are the best form of communication. All involved in the project must have close cooperation. Daily users need to be available for prompt testing and sign off on each phase so developers can mark it off as complete before moving on to the next feature. This might ensure the product meets user expectations but is onerous and time-consuming. This demands more time and energy for everyone involved.

- Greater demands on developers and clients.

These principles require close collaboration and extensive user involvement. Though it is an engaging and rewarding system, it demands a big commitment to the entirety of the project to ensure success. Clients must go through training to aid in product development. Any lack of client participation will impact software quality and success. It also reflects poorly on the development company.

- Lack of necessary documentation.

Because requirements for software are clarified just in time for development, documentation is less detailed. This means that when new members join the team, they do not know the details about certain features or how they need to perform. This creates misunderstandings and difficulties.

- The project easily falls off track.

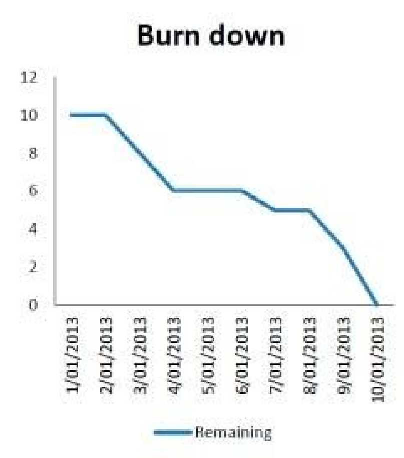
This method requires very little planning to get started and assumes the consumer's needs are ever-changing. With so little to go on, you can see how this could limit the agile model. Then, if a consumer's feedback or communications are not clear, a developer might focus on the wrong areas of development. It also has the potential for scope creep, and an ever-changing product becomes an ever-lasting one.

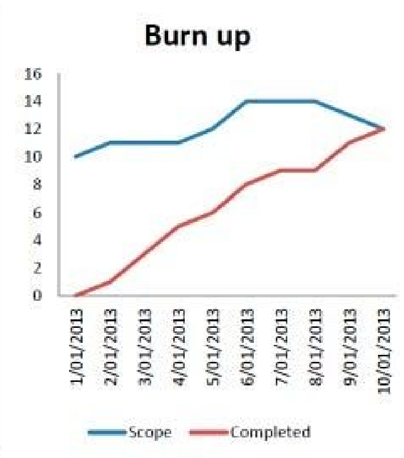
8. Explain the use of burn-up and burn-down charts.

Burn-up charts

A Burn Up Chart is a tool used to track how much work has been completed, and show the total amount of work for a project or iteration.   It's used by multiple software engineering methods but these charts are particularly popular in Agile and Scrum software project management. The completed work and total work is shown on the vertical axis in whatever units a project team feels works best, i.e., work-hours, work-days, story points, or any other work unit. The horizontal access displays time, usually in days, weeks, or iterations (sprints).
Burn-down charts

Burn Down Charts are simple and easy for project members and clients to understand. A line representing the remaining project work slowly decreases and approaches zero over time. However, this type of chart doesn't clearly show the effects of scope change on a project. If a client adds work mid-project the scope change would appear as negative progress by the development team on a Burn Down Chart.





9. Explain the role of the Scrum Master?

The scrum master is the team role responsible for ensuring the team lives agile values and principles and follows the processes and practices that the team agreed they would use.

The responsibilities of the Scrum Master:

- Clearing obstacles
- Establishing an environment where the team can be effective
- Addressing team dynamics
- Ensuring a good relationship between the team and the product owner as well as others outside the team
- Protecting the team from outside interruptions and distractions.

The scrum master role was created as part of the Scrum framework. The name was initially intended to indicate someone who is an expert at Scrum and can, therefore, coach others. The role does not generally have any actual authority. People filling this role have to lead from a position of influence, often taking a servant-leadership stance.

10. Explain what is meant by story point in Scrum?

The scrum guide tells us that estimates should be provided by people that will be doing the work but it doesn't tell us how we should provide estimates. It leaves that decision to us. A common tactic used by scrum teams is to estimate using a unit of measurement referred to as the Story Point.
"A Story Point is a relative unit of measure, decided upon and used by individual Scrum teams, to provide relative estimates of effort for completing requirements".
A story point is a high-level estimation of complexity involved in the user stories, usually done before sprint planning, during release planning or at a pre-planning phase. Story points along with sprint velocity provide a guideline about the stories to be completed in the coming sprints.

Story Points are intended to make team estimating easier. Instead of looking at a product backlog item and estimating it in hours, teams consider only how much effort a product backlog item will require, relative to other product backlog items.

11. Name a few other Agile frameworks.

Kanban
This agile framework gives you a visualized workflow so you can break work down into small pieces. Kanban helps you identify bottlenecks and waste and reduces wait time by giving you well-defined project limits, explicit process policies, and helping you measure and manage flow.

eXtreme Programming. Often used with scrum, XP is an example of how Agile can heighten customer satisfaction. Rather than deliver everything the customer could ever want far in the future, it gives them what they need now, fast. XP is centered on frequent releases and short development cycles. It uses code review, pair programming, unit testing, and frequent communication with the customer.

12. Is it ever suggested to use waterfall over Scrum? If yes, explain when.
Yes,
There is no such a thing that for one to exist the other has to disappear.

Waterfall is also not the past and Scrum (or Agile, or Lean), the future.

Anything that's strictly procedural works best in Waterfall. Followed by Gaussian-based methods such as Six Sigma.

Anything that's not strictly procedural, where we have goals but many options, trial and error, than we need a risk management approach, and this is PDCA, which is the core concept of Scrum, Agile, Lean, etc.

When both goals and steps are well known, well defined, easily repeatable (even if it's not like that now, you can see how those steps could be done by a machine, or robot), than it's Waterfall.

When both goals and steps are a "range" or just not entirely well known, you will benefit from a trial and error approach, with feedback loops. Research and Development, any manual craft, arts, software development, sports, fit in that category.

The discussion over "which is better" makes no sense if one doesn't know the circumstances and what to achieve.

13. What is the difference between the Sprint Planning Meeting and Sprint Retrospective Meeting?

Sprint review is the meeting at the end of the sprint where the Scrum team and all the stakeholders get together and discuss what has been accomplished during the sprint and whether the sprint goal has been met.
The meeting should be attended by the product owner and manager, Scrum Master, development team, management, and anyone else who is involved in the product creation process.

This is usually an informal type of meeting where the Scrum team presents the product increment, while the other stakeholders provide feedback. It shouldn't last more than four hours per month-long sprint. The Scrum Master needs to ensure the meeting doesn't run longer than that.

Sprint retrospective meeting takes place immediately after the sprint review. While sprint review is a discussion about what the team is building, sprint retrospective is focused on how they're building it.

The goal of sprint retrospective is improving the development process. The Scrum team reflects on the previous sprint and discusses what's working well, what could be improved, and how they could improve it to be more productive.

Although these improvements may be implemented even during the sprint, sprint retrospective gives the Scrum team a formal opportunity to discuss the process and motivates each team member to voice their opinion and ideas.

In simple terms, the **Sprint Review is focused on the product** and maximizing the business value of the results of the work of the previous sprint and the **Sprint Retrospective is focused on the process** and continuous process improvement.

14. Explain the "Planning Poker" technique?

Planning Poker is an agile estimating and planning technique that is consensus based. To start a poker planning session, the product owner or customer reads an agile user story or describes a feature to the estimators.

Each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100, which is the sequence we recommend. The values represent the number of story points, ideal days, or other units in which the team estimates.

The estimators discuss the feature, asking questions of the product owner as needed. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time.

If all estimators selected the same value, that becomes the estimate. If not, the estimators discuss their estimates. The high and low estimators should especially share their reasons. After further discussion, each estimator reselects an estimate card, and all cards are again revealed at the same time.

The poker planning process is repeated until consensus is achieved or until the estimators decide that agile estimating and planning of a particular item needs to be deferred until additional information can be acquired.

15. Agile and Scrum certifications are hot in the market and organizations are expecting the candidates to hold one or more out of it. Name a few certifications. (aware of them :)

- Certified Scrum Master (CSM):
- Advanced CSM:
- Certified Scrum Professional (CSP-SM)
- Professional Scrum Master (PSM)
- SAFe 4.0 Scrum Master (SSM)