# 

# X-AGILE INTERVIEW CHEAT SHEET

## Tell me about yourself?

* Talked about when you can to the US,
* Talk about when you started your IT career(e.g 2011) and that you have been in IT for over (XX) Years now.
* Talk about starting as a (E.G started as a business analyst in 2012), with this company (XXX), where you were doing (MENTION atleast 2 functions from your resume)
* Then talk about how working in the field you became expose to AGILE teams and was fascinated by how scrum masters were working with different teams , and you started taking trainings to become a scrum master, eventually landing your first scrum master job with this company (XXX AS ON YOUR RESUME). Mention at least one project u worked on with this company and how many years you worked there. Give a brief description of your team dynamics with this project (e.g : I was leading two teams , with some of devs distributed in india , Honkong and the UK
* Follow up with the other companies on your resume, mention one project with the companies if you can , talk about your duties facilitating all sprint ceremonies, helping PO gather requirements and refine backlog to yield maximum ROI for the business
* Talk about your last project with your most recent company , what you were doing, if the environment was SAFe mention that. If you helped them transition from waterfall to Agile mention that. Mention team dynamics, and how you as a scrum master focused more on the people than the processes so that you can create a collaborative environment where they can work at their maximum potential.
* Mention your degree from the University of XX, and If you have a certification mention.
* Talk about having great experience working as a scrum master for distributed and co-located teams at your previous jobs, and how you have worked with both waterfall and agile teams , and know how to get the best out of team members
* Talk about your work ethic, how you are detail/process oriented, a servant-leader who always make sure your teams are implementing scrum correctly and coaching them to be self-organize and self-managing , while always adhering to the values of scrum .
* Mention you are knowledgeable in other agile frameworks( XP, SAFe, Kanban, TDD, etc) not only scrum, and that you sometimes use some of their techniques to get the best out of your teams
* Lastly I belief in agility and also apply some of its features in my daily life and goals because it fosters room for continuous improvement and gets work done effectively.

**SCRUM IS YOUR TYPICAL ROLE AS A SCRUM MASTER**

* scrum master role is to help the teams internalize and leverage the scrum values to deliver valuable software whilst facilitating the scrum ceremonies
* Increase collaboration amongst team members so they can be self-organizing and self-managing themselves. Ensure the team is maximizing its capacity and delivering valuable tested working software every 2 weeks.
* The scrum master main tasks are to make the Scrum team understand how Scrum operates, to protect the Scrum Team from external interruptions and to remove impediments that hinder the Scrum Team to reach its maximum productivity.
* The scrum master is an enabler who helps the teams identify and remove impediments preventing them from completing their work.
* Scrum master coaches the scrum team to follow the scrum processes right and also to internalize the values of scrum.
* Scrum master is also a change agent, who helps to improve agility in an organization and also adds inputs to areas that might add value to the scrum process.
* A Scrum Master finds tactful solutions to new and unexpected problems.

**WHAT MAKES YOU A GOOD FIT FOR THE JOB?**

I think I will be a good fit for this job because :

* I have excellent interpersonal communication skills, which helps me facilitate collaboration between the team and external stakeholders.
* I am also an excellent schedule manager, who is able to multi task, and work effectively being the scrum master of multiple teams, both distributed and co-located
* I am a good servant leader , who understands the growth of me as a scrum master comes from the growth of my team . I strive to build trust amongst my team members , making them understand I am available to them . I don’t work for praises or pads on the back, but strive to create an environment where my team can work effectively and deliver valuable software with no impediments. My success stems from the success of my team so I always strive to promote a sense of community and a holistic approach to work within the team.
* I am great at conflict resolution, and also strive to create a trust environment with my teams , such that they adhere to the values of scrum , and are always transparent with their work so we can effectively deliver software and meet sprint goals.
* I am also a good Impediment removal, who actively seeks out, anticipates and work to remove impediments that prevent the team from doing work, and also protect and shield them from external work and outside distractions
* I am Passionate and devoted to the scrum process, and able to facilitate all scrum ceremonies making sure they run smoothly and within set time-boxes
* Also knowledgeable in other agile techniques like SAFe and KANBAN, and XP. and as plus I recently just got certified as a Scaled Advanced Scrum Master.
* MAKESURE YOU READ ABOUT THE COPMPANY YOU ARE INTERVIEWING FOR, UNDERSTAND THE JOB DESCRIPTION AND USE SOME OF THE ROLES AND RESPONSIBILITIES FROM THERE TO PASS ON AS SKILLS YOU HAVE

**WHAT PROJECTS HAVE YOU WORKED ON?**

* Name at least 2 projects
* With one of the projects from your most recent company.
* Emphasize your roles in this projects, how you were scheduling all sprint ceremonies, attending product discover and requirement gathering meetings and helping the PO maximize the values of the projects by constantly helping him prioritize and refine the backlog

**IMPEDIMENT / HOW YOU REMOVED IT:**

* An impediment I encountered with my last team was with a developer not able to conduct his test. The developer was done with a story , but during testing, his seed-box was not working, it was showing busy . So he came to me with the issue and I contacted another team that was responsible for seed-boxes and requested a new seed-box. Once he got the new seed-box he put it on the runway and it was running properly, he tested the code and deployed it to his local seed-box with no issues.
* THE ABOVE IS ONE EXAMPLE OF A DETAIL TECHNICAL IMPEDIMENT, YOU CAN ALSO TALK ABOUT THE ONE WE DISCUSSED IN CLASS ABOUT DEVS NEEDING CLARITY FROM PRODUCT OWNER WITH A PARTICULAR STORY, BUT PO WASN’T DEDICATED TO TEAM AND HARD TO REACH. TALK ABOUT REACHING HOW TO THE PO , TALKING TO HIM AND GETTING HIM TO THE TEAM TO ADDRESS THE ISSUE
* I always let my team know they don’t have to stay for more than 15 minutes with an impediment. As soon as they have one I encourage them to reach out to me so I can try to get a solution sooner.

**CONLFICT/HOW YOU RESOLVED IT:**

* An example of a conflict I experience with my team , was personality complex with two of my developers. They suddenly stopped talking especially during retro, I was monitoring them and this went on for a while.
* I had a one on one sit down with them , and one told me he felt his opinions and ideas didn’t matter , that everybody always listened to the other developer and always did things his way , they always ignored his position, so he decided being silent was better. So I told him this wasn’t the case and that in Agile we value everyone’s opinion the same and we are equal and we become successful if we work as a team with diverse opinions, and sharing new ideas.
* I brought the other developer together and spoke to them let them know everyone is equal in the team , no one is bigger than the other person, everyone’s opinion and ideas matter a lot. And apart from that we need to stay focus on our sprint goals and let go of personality conflicts and superiority complex. After speaking to them they improved a little bit but I discovered there was still some tension so, what I did was I invited the team out for happy hour so we could bond and grow with a team building experience. We drank, talk about family. Sports socialize and bonded. After that we got a little bit closer as a team , and the changes were visible in the next retro and daily scrums.

**DAY TO DAY?**

* Get to work at 9, have daily stand-up from 9:30-9:45. Then immediately after that I usually follow-up with some team members who raised some concerns during the standup.
* Depending what day of the week it is, with my previous team , we had backlog refinement every Tuesday from 1-2, and planning session Tuesday from 1-5pm.(max to not exceed 4 hours)
* On Wednesday which was the end of the sprint cycle , we have the sprint review and retrospective , where we sit and talk about what went well, what didn’t go well and come up with action items.
* So that is pretty much my day to day as a scrum master, some days I just had the standup, then follow up with team throughout the day if they had any impediments they need me to remove or escalate. So it just depends what day of the week because some other days I have backlog refinement, some days review and retro.
* I also usually have to attend meetings , and also check and reply emails.
* Make use of metrics, access reports, just to always constantly check if team is on track, reach out to team members that are not doing maximum effort, try to resolve any impediments.
* Also send sprint reports to managers and PO if requested weekly
* Attending weekly scrum of scrums meetings (SOS), if you working in a company with multiple scrum masters, just to share knowledge , exchange ideas, keep track of dependency stories, and find new ways of making your teams perform better.
* Also attending monthly CoPs(Communities of practice) to foster learning, understanding agility and keep up to date with industry trends and standards

**IF A TEAM MEMBER IS NOT PULLING THEIR WEIGHT:**

* If a team member is not pulling their weight , I try to sit down and have a one on one with them and find out what is wrong. Maybe it’s because the process he or she is using or wrong estimation of the story, or maybe the issue is external and has nothing to do with the project.
* He is not pulling their wait because of wrong estimation , maybe he thought it was a 3 point estimation but it turn out it was an 8 point estimation, so next time I need to make sure we estimate well. Or maybe he needs pairing with another team member who has the skillset for completing that story
* If he didn’t have much understanding of the story, I schedule him to sit down with the P.O so he can clarify the story with him.
* So if it because of poor estimation we have to start estimating well so all members pull their weight, and if its lack of understanding of the story, we just need to let P.O clarify the story more and I also encourage more grooming sessions for refine stories.
* Or maybe he is not pulling the weight because of an external, personal issue, so I try to talk with him to find out what is going on and advise maybe he can request time off to resolve his issue , so that he come back and be more focus on the job because , him not pulling the weight drags the team back and in agile, we are more successful when everyone does their part , and we move as a team towards completing sprint goals.

**How can you explain scrum to someone who knows nothing:**

* Ok let’s look at waterfall for building a website. You will build the entire website , with all its functionality( like the homepage, the log in page, the career page, and every other pages pertaining to the website) and deploy it for the customers to use and the development might take months , and the company is not making money during this phase, they have to wait for the whole complete project to be completed and deployed before the company starts making money. But with Agile you can start by building the homepage and send it out to the customers to use , while you working on the other pages, and here the company is still making some money , while customers are getting familiar with the website and giving valuable product feedback that helps the team know if they are on the right track , and also gives the business and idea of what add-ons on the website will increase the business ROI. Take an example like facebook it started with just friend requests , but now you can make video calls, send money and also shop the marketplace
* Scrum emphasizes a team -based collaborative development approach organized around short sprints that let the team produce incremental and releasable versions of the software. They do this by adhering to some basic scrum values that makes the team become self-organized and high performing. And unlike waterfall where the development cycle follows top down directives from professional managers budgeted over a fix length of time, with agile , we go through a series of scrum ceremonies , with self-organizing teams of developers releasing potentially shippable software in short sprints
* Explain the scrum ceremonies briefly ( backlog grooming, sprint planning, sprint, daily standup, sprint review and retro).
* The first value is commitment, when we go through the sprint planning and have picked out the stories to work on during that iteration or sprint cycle, it’s important that team members commit to their stories they have picked and strive to complete them during the sprint, and try to avoid spill overs with stories going back to the backlog for the next sprint.
* Also another important value is openness. I always tell my team this is the most important value , because transparency is a very big deal in agile. Why??? because it creates visibility and shows roadblocks/impediments that may hinder the progress of the project. So being transparent means if you have any roadblock or impediment it is good you mention it early so you get help sooner before it becomes an issue preventing the team from achieving the sprint goal
* And also focus, meaning the team has to focus only on the sprint backlog items which they have decided to work .
* Respect, which is one of the scrum values, respect amongst team members is important , value everyone’s opinion , creates high performing collaborative teams.
* And lastly courage, to complete the sprint , even if we have challenges, and courage as the scrum master to enable and serve the team to follow the scrum framework and agile manifesto.
* And thus thru this values , complex software development is made easier encompassing all the pillars of the agile manifesto that values:

1. Individuals and interactions over processes and tools (scrum promotes effective interaction and communication between team members so that they deliver valuable product)
2. Customer collaboration over contract negotiation (the scrum framework is designed to promote collaboration , team members collaborate with each other to find the best way to develop the software instead of following a plan for top managers who do not actually take part in the development)
3. Responding to change over following a plan (the complex nature of software development and the constant evolution of technology frequently foils elaborate up-front planning. So agile methods accept that projects evolve and improve with time and so create room for changes to be made, and scrum permits developers equip to handle changes. )
4. Working software over comprehensive documentation. (Scrum requires a working potentially shippable product increment as the primary result at the end of each sprint. I understand that the increment may not include enough functionality for the business to decide on the shipment, but it is still important the team strive to complete the sprint goal , with potentially shippable product

**What is Velocity:**

* It is the average number of story points a team completes each sprint, it is a predictable pace at which the team can successful get work done within the particular sprint. It is usually used to determine:
* How many user stories are going to be included in a given sprint.
* How many sprints do we need to complete a set of user stories
* Calculated using normalized technique or from historical data based on previous sprints (avg 3-4 sprints)
* Historical data ( calculate the total number of completed story points within the last 3 sprints , divided by the number of sprints (3)).

**ESTIMATION TECHNIQUE:**

* Estimation is a method for measuring how long it will take to complete a user story, and we use to do estimation in story points , using a conversion of (1point = 1 day’s work)
* Stories are estimated relative to the smallest story. Hence the team needs to identify their smallest story which is assigned a 1 and thus every other story is estimated relative to that one.
* We use the modified Fibonacci sequence (1,2,3, 5,8,13,20, 40, 60, 100) for estimation , and every team member (from BA’s, developers, to testers) all take part in the estimation process.
* The technique I used with my previous teams was planning poker. I usually give each team member a deck of cards containing the Fibonacci numbers, the P.0 usually reads the smallest story that may get a 1, each team member selects a card representing their s story points estimate for that user story and the whole team turns over the card at the same time. If they all have the same number , then that’s the estimate for that user story. The process is done for all user stories, and if some developers estimates vary greatly from the average, they explain their estimates, the team discusses and clarifies any doubts and estimation is done again till the team converges on one number for the user story. Estimation is done for user stories from the product backlog until we have enough story points to meet the teams capacity for that sprint.

**what is agile:**

Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

**What is Scrum:**

scrum is an agile framework leveraging a continuous iterative and incremental approach to developing complex software.

**Agile Manifesto:**

Individuals and interactions over processes and tools

Customer collaboration over contract negotiation

Responding to change over following a plan

Working software over comprehensive documentation

**Backlog grooming?**

* At my last job, we always had backlog grooming on Tuesday. Knowing the P.O is the content authority of the backlog I always pre-ready for the meeting by sending him an email over the weekend, to outline clear goals of his expectations and scope of discussion that will arise from the meeting. During backlog grooming, we are either creating new stories , or refining existing stories per business needs. I also assist the P.O in prioritizing the stories, so we talk over stories and depending on customer’s needs rank the stories according to order of priority that will help achieve business goals. I like using the MoSCoW prioritization technique because it’s easier for P.Os coming from a business background to comprehend. With the technique we rank stories that are a must half first, followed by should half, then Could have , and bottom stories that might not be necessary for the product, are rank at the bottom. This way we always have stories at the top of the backlog that will bring the most immediate value increment the business wants that can increase their ROI. I also was ensure we have shovel-ready stories for at least 3 sprints in advance just in case the PO is absent , or takes vacation, this way my I and my teams are not caught off guard , and we always know what to work on .

**SPRINT PLANNING:**

* During my previous sprint planning , we sit down with the team and the P.O outlines the iteration goal. Here the P.O gives us what they business wants , and then the development comes up with the how. i.e the tools, skillset they are going to use to turn the wants into some working software .
* Solidification: We start by looking at the prioritized stories of the product backlog. Once everybody has a clear understanding of the stories , especially the ones of high priority and what is needed for each story, we then move to estimation.
* With estimation with my previous team , I usually use the planning poker technique with a modified Fibonacci deck of cards to estimate the stories. Since our sprints were 2 weeks, we usually use deck of cards with 1,2, 3, 5, 8 and 13.
* During estimation I share the poker cards to all the developers since they are the technical ones actually doing the work.
* So the P.O picks a story reads it , the team discusses briefly on the story, then vote on a count to 3 , showing their card representing the story point they think the story should get. If someone shows a 1 , it means they have a clear understanding of the story and know the procedure and technical requirement it will take to complete they think it will take a day’s work. So they directly saying it is not complex and easy to achieve.
* If someone says the story is an 13, it means they are indirectly saying it is complex and difficult or they don’t understand what goes into completing it.
* So what I do is after the vote I let the person with smallest story point explain to the large story point why they considered it a 1 and what go into completing the story, and why it is not complex as they might have thought. After they have discussed on the story , exchange ideas and level-set, I always do a re-vote with the team until they all have the same story-point estimate for that particular story.
* After that s done we move to the next story until all the prioritized stories are estimated.
* After the estimates, we move to calculating the team’s capacity (that is the availability of the team ) to work on that sprint. We always plan for an 80% capacity , and keep 20% aside for unforeseen or inject stories or any other events such as meetings that may occur during the sprint .
* When calculating the velocity to determine how much work the team can work on for that sprint , with my previous team we use to do a conversion of 1 story point = 1 day of work, so for the 2 weeks sprint( i.e 10 day’s work) each team member gets 8 points (factoring the 2 points for the 20% buffer). So if the developers are taking no days off , their velocity every 2 weeks will be 8, and if they take days off , we just subtract the days from their capacity.
* So after knowing the team capacity for that sprint , each team member then picks a story based on their availability. So they then commit to working on stories with a maximum of 8 points per sprint.
* So after the capacity planning we add all teams members availability and know the velocity of the team. So we commit to stories for that sprint summing up to the velocity. With time after 2 or 3 iteration we have a working average of what the team’s velocity is, and don’t have to calculate it every time.
* Moving forward, after the capacity, we just go ahead and commit to the stories the team picked , making sure everyone has the sprint goal in mind and understands the definition of done of each story, and making sure they can deliver within 2 weeks as planned. Sometimes I also conduct a vote of confidence by a show of hand to see if everyone understands what is needed and to clear any concerns.

**Daily standup:**

* When I joined (NAME YOUR LAST COMPANY HERE), my first meeting with the team was coming up with a working agreement on the norms of our daily scrums ( such as , everyone standing, one person speaks at a time, everyone shows up on time, every day at the same place,, and keep it within 15 minutes).
* I always scheduled the meeting for 9:30 am , but always strive to be at the room 15 minutes prior to view the burndown chart and get the task board ready for the session. During the meeting each of the 6 developers explained in brief the story task they were working on . They give an update what they did yesterday, what they will work on today, and if they had any impediments or roadblocks I wrote it down, to follow up with them for resolution .
* I was encourage the team to speak in brief about their stories so we can keep the meeting within 15 minutes, and sometimes when developers were going in detailed I intervene and let them comment with a note on the parking lot for a meet-after to go over their concerns. Each team member usually spoke for about 2 minutes, and to conclude the standup ,we always look at the burndown chart together, to see if our workflow is going as planned and making progress as a team and are on track to deliver all the stories we committed to for that sprint. At the end of the meeting I always conduct a vote of confidence by a show of hand to see the overall optimism of the team to meet sprint goals.

**SPRINT REVIEW:**

* This is session we actually showcase all the stories the team worked on during the sprint
* The session was usually two hours maximum
* The Product owner usually ran this meeting , and the P.O uses the acceptance criteria to mark the stories as done , and once the stories are marked as done , the stakeholders accepts the stories or recommend changes on the stories , which goes to be work on for the next sprint. Stories are then deployed for staging and wait release based on the release schedule of that company
* The development teams demos the product during this meeting.

**Retrospective:**

* This is where we sit down with the team members , and do the inspect and adapt ( one of the pillars of scrum ), we sit down and inspect how the sprint cycle went, and come up with new solutions moving forward . i.e. we adapt to changes and look for better ways to collaborate and make future sprints to be more effective.
* During the retro, we talked about what went well, what went wrong and brainstorm on possible solutions. To save time I always gather some pre ready data from the team using ideaboadz. So a day before the retro I send a link to the team to gather feedback to use during the retro session: I use a start , stop , and continue format to gather the information.
  + - * 1. Continue column: things we should continue doing , things that went well in the sprint. Keep doing it
        2. Stop column: things we should stop doing that hinders sprint progress. We stop doing them
        3. Start column: This are possible solutions or action we have to start doing to overcome all impediments or any difficulties we faced during the previous sprint. Follow up with team members to make sure we implement this action items in the future sprint. we vote together as a team on the action items and agree to implement highest one going forward.

**MANAGE SCOPE CREEP:**

* The best way to manage scope creep is by openness and transparency. Like if they are any changes in the scope and you might not meet the deadline, Send an email to the team the P.O and stakeholders to let them know you might not finish the work on time. Being transparent well ahead of time, makes the stakeholders trust you more and doesn’t make your team look bad coming up short at the end. Because looking at the burndown chart you will know if you are on track to complete the work in time or not.
* If the PO is the one trying to increase SCOPE with additional stories while we already sprinting , I make sure we talk together as a team , see if we have capacity to incorporate the new scope, if not advise the PO to do a trade-off with a story in the backlog of similar estimation points, that way team is still within capacity and on track to meet sprint goals

**A GOOD USER STORY:**

* Can be delivered within 1 iteration
* Should fit within a 4x6 index card
* Should have conversation showing what is going to be done and how it looks at the end
* Should have a defined tracking and performance criteria
* Independent. The story should be self-contained, not dependent on another user story.
* Negotiable, until becoming part of the sprint backlog, the user story should be able to be changed or rewritten.
* valuable, (each user story has to create value for the end user or the customer)
* estimable, (easy to estimate by the scrum team)
* small, (small so it can be easy to plan , task and prioritize with certainty, and able to completed within 1 iteration )
* testable (has to be testable, per definition of done by the consumer)
* Has a definition of done

KANBAN BOARD COLUMNS:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| backlog | In progress | Code review | QA | Done |
|  |  |  |  |  |
|  |  |  |  |  |

SDLC

1. Planning and requirement analysis
2. Design project architecture
3. Development and programming
4. Testing
5. Deployment, track and monitor

**HOW TO SPLIT LARGE STORIES:**

* Simple/Complex Business rule variation

S.P.I.D.R splitting technique

* **SPIKES:** Make a large story smaller by pulling out a spike, which is research activity to to gain clarity and know more about the story.The research might lead to knowledge on how to split the story or making the tasks for the story manageable
* **PATHS:** Consider the paths through a story and split each path into its own story. You can do this by drawing a flowchart of what happens in the story from concept to delivery, and each path can be its own story.
* **INTERFACES:** If the story goes through many different interfaces which makes it longer to develop, split it maybe by browser type, version or hardware type
* **DATA:** Look for ways to split the story base on the types of data that must be supported for the story. ask questions can the first story support valid data , and later add ons support invalid data?
* **RULES:** Sometimes stories are too large because of the business rules or technology standards that must be supported for the story. Consider relaxing support for this rules in the initial stories , and you can add the support in later stories

**WHAT HAPPENS TO PRODUCT AFTER SPRINT REVIEW( HOW OFTEN DO WE DEPLOY)**

* Once a story is worked and marked as completed and tested it has to be deployed.
* Based on release schedule , it is either staged and ready for deployment. We release every month , based on the update we were working on. At U.S Bank we released quarterly

**Benefits agile?**

* that organizations are capable of significantly reducing the overall **risk** associated with software development.
* in particular, agile development accelerates the delivery of initial business value, and through a process of continuous planning and feedback, is able to ensure that **value** is continuing to be maximized throughout the development process.
* As a result of this iterative planning and feedback loop, teams are able to continuously align the delivered software with desired business needs, easily **adapting** to changing requirements throughout the process
* Lastly, with waterfall, the client doesn’t get as much feedback on the build…i.e You are not engaging your stakeholders as much as with agile, so agile is beneficial because there is constant communication between the build team and stakeholders, with stakeholders having an authentic relationship of trust and confidence with the teams. Transparent communication means we are able to constantly refine and reprioritize the product backlog to align the requirements with market needs.

**BURDNDOWN CHART:**

The Sprint Burndown Chart makes the work of the Team visible. It is a graphic representation of the rate at which work is completed and how much work remains to be done.

* X-axis represents time (the sprint days)
* Y-axis represents the amount of work left to be done, measured in either story (points) or hours (tasks)
* It is use to monitor project progress and track performance of the team.
* The goal is to have all the forecasted work completed by the end of the sprint.

**HOW DO YOU MANAGE THE TEAM:**

* As a scrum master, I am servant leader to my team making sure they have everything necessary to deliver valuable software. I am no managing per se like in the traditional waterfall model.
* Instead my goal is to foster collaboration within the team , facilitate scrum events and get work done without dictating to the team or assigning work to them .

**Definition of Ready:**

it is a working agreement between the team and product owner on what readiness means. It is an input criteria to plan a story in a sprint. It is a way for a product owner to indicate an item in the product backlog as ready to work in sprint. It is the accountability of product owner that there is a definition of ready defined. Scrum team can refuse to take an item into sprint. The team makes explicit and visible the criteria (generally based on the INVEST matrix) that a user story must meet prior to being accepted into the upcoming SPRINT.

|  |  |
| --- | --- |
| Definition of DONE | Acceptance Criteria |
| * The definition of done is a checklist of things that need to be completed for any story to be considered done. * The definition of done is agreed by the team prior to starting work. It covers what the team feels is necessary to consider any story done. * Includes: Acceptance criteria, code review, Testing | * Acceptance criteria are story specific requirements that must be met for the story to be completed. They are a technique for adding functional detail to user stories. Acceptance criteria are often added during backlog refinement or during the sprint planning meeting. |

**Managing technical debt**

* Be vigilant, resolve impediments as soon as possible
* Avoid using shortcuts in development , and make sure devs are committing to stories and tasks they have the skillset equip to do.
* Use simple design and refactor relentlessly
* Unchecked technical debt makes software more expensive to modify than reimplement.

**Benefits of continues integration:**

* Continuous integration is process whereby developers and testers validate the newly written code frequently.
* Devs check in their code to the mainline branch at least once a day, or multiple times a day, every check in triggers a build and all unit tests are run.
* If tests fails, notification is sent to all devs and new code is rolled back automatically.
* Tests help as safety nets to capture any bugs that were inadvertently introduced.
* The advantage is that bugs are caught as soon as they are introduced, and the compatibility of everyone codes is tested.
* You always have the best working version of the code.

**Continuous Delivery;**

* CD adds to CI, one step further in that after the build and automated unit tests are ran successfully, you automatically or manually deploy the application to a test stage or production environment.
* CD and CI are the recipes for implementing successful DevOps in an organization.

**WHAT IS DAILY STANDUP INTENDED TO DO:**

* Prepare the team for the day’s collaboration
* Help the team see if they will meet the sprint goal
* Get an update on the progress on the project and what the team plans to work on in the next 24 hours.
* Find and remove any impediments that may slow the team down from working and thus meeting their sprint goals.

**CHALLENGES WORKING WITH DISTRIBUTED(C0-LOCATED) TEAMS:**

* **Communication:** Distributed teams are limited in their communication channels, scheduling a ceremony such as the daily standup is kind of difficult, you really can enforced su best practices. One part of effective communication is nonverbal cues, reading and understanding them , and using digital tools like webex to facilitate communication , though it helps to an extent, distributed teams are stilled disadvantaged by . Also the accent.
* **Access:** 
  + A critical issue for teams distributed across time zones is having access to each other with regards to core hours and communication tools. Core hours can be tricky—the greater the time differential, the more difficult the effort, and aligning work and making sure everyone understands shared accountabilities becomes even difficult.
  + Lack of proximity with co workers
  + Especially not getting respond to a critical bug or defect , assigned to a distributed team member because of time zones differences, we might take an extra day to get an impediment resolved and thus not working at optimum velocity
  + Offshore teams that cannot establish overlapping hours will likely have a lower average velocity. By way of example, just the act of taking on additional tasks becomes a one-day event
  + Ensuring teams have the communication tools to ensure DSU goes well and everyone understand work progress
  + Establishing core hours
* **Culture/team spirit/personalities:**
  + When the people we collaborate with are from different cultures, things can get worse. We tend to attribute things that go wrong, miscommunications and delays, on cultural differences. We don’t understand how people from the other culture think and what drives their behavior. And if we don’t understand the people we need to perform with, it’s hard to collaborate and bring results
  + People with whom we share social settings also share similar expectations, experiences and perspectives .Shared work identity brings identity to the team
  + Its subsequently hard to build team spirit, and that physical team bonding experience is absent

**DIFFERENCES BETWEEN SCRUM and SAFE:**

|  |  |
| --- | --- |
| SCRUM | SAFe |
| -Scrum is mainly for smaller teams to organize and mage the work | -SAFe is for bigger multiple teams, for the whole organization |
| -Scrum is to the agile team | -SAFe is to the agile enterprise |
| -Scrum is only at the team management level | -SAFe focuses on portfolio management, Program management and team management |
| -Scrum doesn’t have  2day PIP to plan a certain work to be delivered in 3 months , with various iterations | -SAFe has 2 day PIP to forecast what work is in the pipeline for delivery  in the next 3 months, and align all teams so they have one common goal |

**DIFFERENCES BETWEEN SCRUM and KANBAN**

|  |  |
| --- | --- |
| **Scrum** | **Kanban** |
| 3 Set Roles : Product Owner, Scrum Master and Development team | No set roles are defined. might have a project manager |
| Cadence for delivery is regular fixed length sprints . (e.g 2 weeks) | Cadence for delivery is continuous on a needed basis |
| Measures production using velocity through sprints | Measures production by cycle time  (Amount of time it takes  to complete one full piece of project from beginning to end) |
| Uses a pull system where an entire batch of backlog items is pulled into a sprint backlog and worked on for the sprint | Uses a systematic workflow pull system, that allows team members to only pull new items once previous work is done |

**DIFFERENCES BETWEEN WATERFALL and AGILE:**

|  |  |
| --- | --- |
| **AGILE** | **WATERFALL** |
| •Software development model is flexible and allows room for changes in the project requirements even after planning has been completed | •Waterfall software development is structured and often rigid and there is little or no chance to change the requirements once the project starts. |
| •Agile is good for software projects with changing and evolving requirements | •The waterfall model is good for projects with clearly defined requirements and without expected changes |
| •Testing is done concurrently with programming as the software is being built | •Testing is done at the end of the of the build phase, after all the software has been built |
| •They is continuous customer/user involvement with agile throughout the development cycle, as feedback is gathered at the end of each sprint and implemented into the build. | •Less customer/user involvement as feedback is only gathered at the beginning of development and at delivery. It makes it hard to implement customer feedback into the build. |
| •Agile follows an iterative development approach and so the planning, development , testing phases appear more than once. (Each Sprint) | •All project development phases (designing, development, testing ) completed once in each phase of the project.  • |

**CHALLENGES IMPLEMENTING SAFe:**

* Collaborating amongst teams is difficult, especially teams with dependency stories identified at PI Planning
* Aligning features with multiple teams, and some other teams might dysfunctional so it makes aligning work harder.
* Communication especially on scope change during iterations, and when you are working with off shore teams, it becomes more difficult.
* It is also hard to identify value streams and synchronize works to make sure value is aligned to business needs. Agile frameworks ensure that the development team is free to design their development plan. SAFe is implemented for many agile teams and result in lack of self-organized team. Sometimes it becomes difficult and inappropriate for teams to work on the same product whose deliverables are not synchronized and that occur as a challenge in SAFe.
* SAFe is anti-agile to an extent: SAFe involves waterfall methodology elements, does not permit testing unless the process is completed, and thus leveraging the benefits of CI, automated tests in scrum environment is lost. Integration here is hard and tedious as you are working in a large integration environment and thus high probability of bugs/ defects in the system.

**WHAT HAPPENS IF THE PO IS NOT AVAILABLE, HOW DO YOU ENSURE YOUR TEAM IS STILL WORKING ON TRACK AND KNOW ALL THE REQUIREMENTS:**

* Make sure you refine stories well during backlog grooming to ensure team has shovel ready work to be worked on for upcoming sprints (at least 3)
* Talk to your managers to appoint a proxy PO , who understands the business needs, the requirements and know how to groom, order and maximize the value of the work of the team so whatever stories they are pulling from sprint backlog to the product backlog , yields the most immediate ROI for the business.

**WHAT TECHNOLOGIES/TOOLS HAVE YOU USED:**

Tools:

* JIRA : Our focus is on Jira Software, used by software teams to ; plan , track and release software.
* AGILE CA (RALLY): Agile software like CA Agile Central helps organizations keep track of their business objectives and align around a centralized purpose for better business outcomes. Use it to tie strategy and daily work, track and manage delivery and leverage data to accurately measure performance.
* MS PROJECT, SHAREPOINT, TFS,VERSION ONE

Technology:

* JAVA
* C#
* ANGULA, NODE.JS
* SQL,PYTHON,R HADOOP

VERSION CONTROL:

* GIT
* GITBUCKET

QA Testing:

Some scripting languages used in automated testing are:

* VBScript
* javascript
* Perl
* python
* RubyUnix Shell Script

**DIFFERENCES BETWEEN EPICS, FEATURES & STORIES:**

* **Epics:**
  + A business case,a solution development initiative or a project, which is large enough to require analysis, a big goal/MVP the business is trying to achieve or create, and needs financial approval before implementation.
  + Implementation occurs over multiple Program increments
* **Features:**
  + A unit of work that is greater than a single sprint. They are usually decomposed into stories during story refinement sessions.
  + Implementation occurs over multiple sprints/iterations
* **Stories:** 
  + Stories are short descriptions of a small piece of desired functionality, written in the user’s language. Agile Teams implement small, vertical slices of system functionality
  + They are sized so that they can be completed in a single sprint

**DO YOU HAVE ANY QUESTIONS FOR US?**

* Potentially if I get hired, what Kind of teams will I will I be working with , are they co-located and off-shore
* What technologies are you guys using and what is the project all about
* What agile tool are you using for keeping track of projects