Agile planning & Execution

Surya. D

Planning to be agile:

lol: I love deadlines, I like the whooshing sound they make as they fly by
-Douglas adams.

Plan iteratively:

Don't decide everything at the point when you know the least.

- Planning everything at the beginning will lead to missed deadlines
- Iterative planning allows for course corrections and more accurate estimates.

Agile roles and Need for training:

Product manager vs product owner

Product Manager	Product Owner
Businessperson who manages the budget	Visionary who leads the team in a series of experiments designed to achieve the sprint goal
Manager who is mostly focused on operational aspects of the business	Conduit between the stakeholders and the team, translating between business and technical goals

Project manager vs scrum master

Project Manager	Scrum Master
Task master who keeps everyone marching to a fixed plan	Coach that keeps the team focused on the current sprint
Documents impediments as project risks	Eliminates impediments while buffering team from interruptions

Development team vs scrum team

Development Team	Scrum Team
Consists only of developers	Cross-functional team that includes developers, testers, security, business analysts, operations, and more

You cannot just replace these roles without training.
The mindset should start from the top.

Kanban and Agile planning tools

Agile planning tools: Tools will not make you Agile
Tools can support your Agile process

Zenhub: Plug-in to Github.
→ It provides a kanban board & project management reporting
→ Customizable & integrated with Github
→ Useful where you are in a project based on Github.
→ Gives a quick overview of user stories, sprint planning, status, people etc.

Default zenhub pipelines:

New issues → Icebox (cold storage) → Product backlog → Sprint backlog → In progress → Review/QA → done

→ kanban board is a way of tracking plan items needed to be done, items in process & completed items

USER STORY

A user story represents a small piece of business value that the team can deliver in an iteration.

User story → what, why, who, etc. → A brief description of need and business value

→ Any assumptions / details
→ Definition of 'done' → Acceptance criteria

Story descriptor : (1)

User stories document a persona requesting a function to achieve a goal.

As a < some role >
I need < some function >
so that < some benefit > ⇒ Value out of it; may help in prioritizing

→ List any assumptions (2)
→ Document any details that may help the developer.

Acceptance Criteria : Definition of done (3)

→ It a critical to document This
→ Use : Gherkin syntax ⇒ Helps everyone to understand.

Given < some precondition >
When < some event happens >
then < some outcome >

Sample story :

Assumptions (1- Sensor or Available speed range)

Title : Maintain safe following distance using Adaptive cruise control

As a driver (→ This is the role)

I want the vehicle to automatically adjust its speed to maintain safe distance from vehicle ahead

so that I can drive more comfortably and safely without constantly adjusting the cruise control

Acceptance Criteria :

1- ~~Speed adjustment~~

Given ACC is active and a vehicle is detected ahead within the set distance
when the leading vehicle slows down
Then the system should reduce the vehicle's speed to maintain the selected gap

INVEST: Stories should follow this

I - Independent
N - Negotiable (for priority, and stuff)
V - Valuable
E - Estimable
S - Small (so it can be done in a sprint)
T - Testable

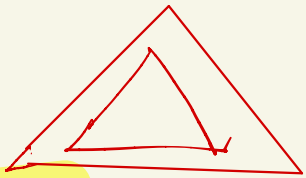
Epics: A big idea
→ User story that is bigger than a single sprint

Epic gets broken down to smaller stories

→ Backlog items tend to start as epics when they are lower priority and less defined
→ for sprint planning, epics should be broken down into smaller stories.

Story points: (Velocity)
→ A metric used to estimate the difficulty of implementing a given user story.
→ An abstract measure of overall effort

what does a story point measure?



Tradeoff btw what you know and what you don't

→ Story points acknowledge that humans are bad at estimating time to completion.

→ Instead, story points use relative T-shirt sizes (S, M, L, XL) mapped to

→ Most tools use Fibonacci numbers (1, 2, 3, 5, 8, 13, 21)

→ Agree on what "medium" is and evaluate accordingly.

→ Relative to the team does.

Story size:

→ A story should be small enough to be coded and tested within a single sprint iteration - ideally, just a few days

→ Large stories should be broken down into smaller stories.

Story point anti-pattern: (Don't do)

→ Equating a story point to wall-clock time → Don't overdo

→ Agree with the team on checking the relative time & efforts

Building the product Backlog: (Ranked list of all unimplement stories)

→ A product backlog contains all the unimplemented stories not yet in the sprint

→ Stories are ranked in order of importance and/or business value.

→ Stories are more detailed at top, and relatively less detailed at bottom (but needs to be addressed)

Backlog Refinement (Order the product backlog and make the stories sprint ready)

→ keep the product backlog ranked by priority so that the important stories are always on top.

→ Break large stories down into smaller stories.

→ Make sure that stories near the top of the backlog are groomed and complete

Who should attend?

→ Product owner.
→ Scrum master.
→ Development team (opt)
  → lead developer / architect

What is the goal?

→ Ranking the stories in order of importance.
→ Make sure the story contains enough info for a developer to start working on it.

New issue triage

The user stories keep flowing in especially being agile.

→ Start with new issue triage.

→ Goal: At the end of backlog refinement the new issues column is empty

→ Take stories from new issue and
* Move them into the product backlog if they will be worked on soon
* Move them into icebox if they are good idea but not now.
* Reject them if they are not where you want to go.

→ Product owner sorts the product backlog in order of importance

→ The team may provide estimates and other technical info.

→ Large vague items are split and clarified

→ Goal is to make the stories "sprint ready"

Labels -

→ Help visualize the work (Bug, documentation, duplicate, enhancement, good first issue, helpwanted, invalid, question, wontfix etc)

↓ Add

Technical debt: (any story that does not add perceived value to the stakeholder)

→ Technical debt is anything you need to do that doesn't involve creating a new feature (Key don't see)

→ Technical debt builds up when you take shortcuts but may also occur naturally.

(some developers claims its enhancement so check accordingly 'lol')

eg

→ code refractor.
→ setup and maintenance of environments
→ Changing technologies like databases.
→ Updating vulnerable libraries.

Backlog refinement tips

→ Refine backlog every sprint to ensure priorities are correct
→ Have at least 2 sprints worth of stories groomed
→ More time you spend refining the backlog, the easier sprint planning will be

Goal is to make backlog ready for sprint planning meeting

In backlog refinement meeting:

→ Triage new issues
→ Make stories sprint ready
→ Create new labels (Enhancement, Help wanted, Technical debt, etc)
→ Add labels to stories.

Sprint planning

→ The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved.

→ This is accomplished by producing a sprint backlog.

Who should attend?

- Product owner
- Scrum master
- Dev team (Dev, test, BA → cross functional teams)

→ Each sprint should have a clearly defined business goal.

→ The product owner describes the goal and product backlog items supporting it.

→ It's important for the whole team to understand why they are building the increment

Mechanics of sprint planning:

The dev team:

→ Takes stories from the top of product backlog and assigns them to the sprint backlog.

→ Assigns story points and labels. (collaborative effort)

→ Ensures each story contains enough info for a developer to start working on it

→ Stops adding stories when you hit the team's velocity.

Team Velocity:

→ Number of story points a team can complete in a single sprint

→ This will change over time as the team gets better at estimating and better at executing

→ The velocity is unique to the team because the story point assignment is unique to the team

Create a sprint milestone:-

→ Create a sprint milestone to start the sprint
→ the milestone title should be short.
→ The description should document the milestone goal.
→ The duration should be 2 weeks in general.

→ It is the product owner's responsibility to present the sprint goal
→ It is the development team's responsibility to create a sprint plan
→ Sprint plan is created by moving stories from the product backlog into the sprint backlog until the team's velocity is reached.

Summary and Highlights

Congratulations! You have completed this lesson. At this point in the course, you know:

- A user story documents a person requesting a function to achieve a goal.
- Using a template helps ensure that stories are complete.
- Defining "done" helps minimize misunderstandings.
- Use the INVEST acronym to remember the qualities of a good user story: independent, negotiable, valuable, estimable, small, and testable.
- Epics can be used to capture big ideas.
- Story points are a metric used to estimate the difficulty of implementing a given user story.
- Story points are relative, like T-Shirt sizes.
- You must agree on what "average" means.
- You should never equate story points with wall-clock time.
- A product backlog is a ranked list of all unimplemented stories.
- Stories high in the ranking should have more detail than those that are lower.
- Create stories using the "As a", "I need", "So that" template to ensure everyone understands who it benefits and the business value it provides.

Summary and Highlights

Congratulations! You have completed this lesson. At this point in the course, you know:

- It is the product owner's responsibility to maintain a groomed backlog
- Backlog refinement is used to order the product backlog and make stories sprint ready
- You start refinement by triaging new issues
- Large stories should be broken down until they are small enough to fit in a sprint
- The goal of backlog refinement is to get the backlog ready for the sprint planning meeting
- It is the product owner's responsibility to present the sprint goal
- It is the development team's responsibility to create a sprint plan
- A sprint plan is created by moving stories from the product backlog into the sprint backlog until the team's velocity is reached

EXECUTING THE PLAN

Executing the sprint.

→ Sprint is one iteration through design, code, test, deploy cycle.
→ 2 weeks in duration
→ Every sprint should have a goal.

Daily execution

→ Take the next highest priority item from the sprint backlog.
→ Assign it to yourself.
→ Move it in process.
→ No one should have more than one story assigned to them unless they are blocked.
→ When you are finished, move the story to Review/QA and open a pull request.
→ When the pull request is merged, move the story to the Done column.

Daily standup (/Daily scrum)

→ Occurs everyday at the same time & place
→ each team member briefly reports on their work.
→ Called a "stand-up" coz everyone should remain standing during the meeting to keep it short
→ Time boxed to 15 mins.
→ Not a project status meeting — all status should be tabled for later discussion

Who should attend?

→ Scrum master
→ Development team
→ Product owner (optional) (p.o should not speak unless necessary, it is for the team & not status reporting)

Daily standup Questions!

→ What did I accomplish the previous day?
→ What will I work on today?
→ What blockers (or) impediments are in my way?

Impediments & blockers:

→ Should be unblocked by scrum master
→ Developers that are blocked can work on next story

Tabled topics:

→ Topics raised during stand-up should be held until the meeting has ended
→ Scrum master should make a note of it. Sometimes called "parking lot".
→ Anyone interested in those topics can stay to discuss.

Completing the sprint

Using burndown charts

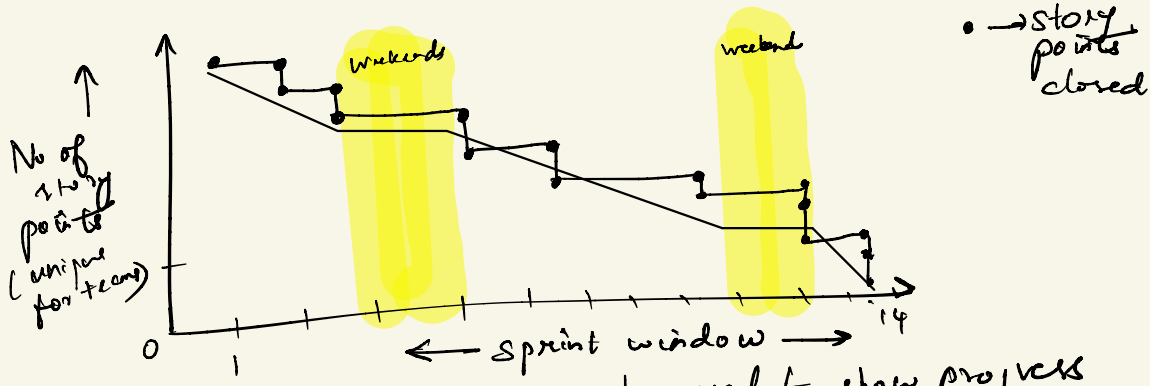
Milestones & burndowns :

→ Milestones can be created for anything in your project.
⇒ Sprint, beta drop, demo, release ....

→ Burndown charts can be used to measure your progress against a milestone

→ The measurement of story points completed (vs) story point remaining for a sprint

→ Over time the story points remaining should go down, hence the name : burndown.



→ " Burndown charts can be used to show progress towards any milestones, not just sprints.

→ Used to forecast the team's probability of achieving the sprint goal.

→ Shows progress

Sprint Review : Demonstration of features implemented during sprint

→ Live demonstration of implemented stories
→ Product owner determines if the stories are done based on acceptance criteria.
→ Done stories are closed.

Who should attend?

- Product owner
- Scrum master
- Dev team
- Stakeholders
- Customers (optional)

→ Feedback gets converted into new product backlog stories
→ This is where iterative development allows the creation of products that couldn't have been specified up-front in a plan driven approach.

Rejected stories :-

→ Stories that are not considered done, add a label to indicate this & close them.
→ Write a new story with new acceptance criteria
→ This will keep the velocity more accurate

Sprint retrospective

→ A meeting to reflect on the retrospective — "A time to reflect on how the sprint went is"

→ Measure the health of the process

→ The dev. team must feel comfortable to speak freely.

→ Who: Scrum team, Development team
Product owner is optional but ideally don't invite so dev team can open up freely

→ 3 questions answered: 1. What went well? (keep doing)
2. What did not go well? (Stop doing)
3. What should we change for next sprint?

This is scrum master's role.

→ This is critical for maintaining a healthy team

→ S.M ensure that changes are made as a result of feedback

→ Goal is to improve for next sprint

Measuring Success:

Using measurements effectively: Measurements & metrics

→ You can't improve what you can't measure
→ High performing teams use metrics to continually improve.
- They take baselines and set goals and measure against them.

Beware of Vanity metrics

→ It is important to be sure that the metrics you use are actionable

Work on Actionable metrics:

→ It is important to take a baseline before trying to measure change

- Use A/B split-test to deploy new features to 50% of users
- Measure behavior of those who saw the feature and those who did not
- Do more of what produced the desired behavior

Top 4 actionable metrics:

→ Mean Lead Time: How long does it take from idea to production?
→ Release frequency: How often can you deliver changes?
→ change failure Rate: How often do changes fail?
→ Mean time to Recovery (MTTR): How quickly can you recover from failure?

Examples

→ Reduce time to market for new features
→ Increase overall availability of the product
→ Reduce the time it takes to deploy a SW release.
→ Increase the % of defects detected in testing before production release.
→ Provide performance & user feedback to the team in a more timely manner.

End of sprint activities :-

→ Move stories from done to closed
→ close the current milestone
→ Create a new sprint milestone.
→ Adjust unfinished work.

Handling untouched stories :-

→ Stories not worked on can be moved to the top of the product backlog.
→ Resist the urge to move them to the next sprint
→ Remember to un-assign them from sprint milestone.

Handling unfinished stories :

→ Do not move unfinished stories into the next sprint!
→ Give the developer credit for the work they did.
→ This will keep your velocity more accurate
→ Adjust the story points accordingly
→ Adjust the descriptions & story points of the unfinished story, label it unfinished, and move it to done
→ Write a new story for the remaining work.
→ Assign remaining story points & move it to the next sprint

Ready for the next sprint

- All stories assigned to the current sprint are closed
- All unfinished stories are reassigned
- The sprint milestone is closed
- A new sprint milestone is created.

(This will reflect the velocity of the sprint)

Agile Anti-patterns & Health check

You will find

- No real product owner (have a proper visionary P.O)
- Teams are too large (Don't have teams, approx 10)
- Teams are not dedicated coz of multiple projects.
- Teams are too geographically distributed.
- Teams are siloed (dependency on anyone outside your team should be avoided)
- Teams are not self-managing.

Scrum health check

- The accountabilities of everyone are identified & enacted
- Work is organized in consecutive sprints of 2–4 weeks (or) fewer
- There is an ordered product backlog.
- There is a sprint backlog with a visualization of remaining work
- At sprint planning a forecast, a sprint backlog & a sprint goal are created

→ The result of the daily scrum is work being re-planned for the next day.

→ No later than by the end of the sprint, a Done increment is created.

→ Stakeholders offer feedback as a result of inspecting the increment at the sprint review.

→ Product backlog is updated as a result of the sprint review.

→ PO, dev team, S-M align on the work process for their next sprint at the retrospective.