Introduction

Throughout this reading, we will be discussing Sprint Retrospectives, their role in the development lifecycle, efficient processes for doing them, and ways to structure their activities.

What is a Sprint Retrospective and What Does It Do For the Team?

According to the Agile Manifesto, one of the 12 principles of agile software development is,

> "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."

A Sprint Retrospective is a dedicated meeting that takes place at the end of an iteration that is focused on evaluating and proposing changes to the development process, the team dynamic, and seeks to resolve related issues.

A Sprint Retrospective is a dedicated meeting that takes place at the end of an iteration that seeks to resolve related issues and is focused on evaluating and proposing changes to the development process and the team dynamic.

While the Scrum meetings and time during the sprints tend to focus on the product and the code development, sprint retrospectives give teams a time to focus on how the team accomplishes their tasks and interacts with each other.

Retrospectives are meant to:

- Identify and handle issues
- Improve team practices
- Identify and put plans in place for removing obstacles.

The Structure of A Retrospective

Agile Retrospectives recommends the following structure to follow during this process:

1. Set the stage

2. Gather data
3. Generate insights
4. Decide what to do
5. Close the retrospective

The remainder of this reading will break down what is meant by each of these steps and identify activities that can be done to provide concrete structure to each step.

Set The Stage

In order to have a productive retrospective, we need to begin by establishing some goals, as well as establish an atmosphere in which everyone feels both welcome and expected to contribute.

Everyone on the team should feel comfortable talking about both the things that are going well and the things that could be improved. This meeting must not feel like a witch hunt or involve people being personally attached. The conversation should be constructive and focused on ways to improve the team. When in doubt, refer to the Retrospective Prime Directive:

> "Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand."

--Norm Kerth, Project Retrospectives: A Handbook for Team Review

As you start the meeting, find a way to get everyone to start by saying something in only a couple words. Exercises might include the following (pick one or two max):

- Ask each team member what their goal is for the meeting.
- Ask for, in a word, what the team's biggest strength was this sprint.
- Ask for, in a word, what the team's biggest weakness was this sprint.

Next, provide an outline for the retrospective. Talk about the main things you are looking to address and what activities you will be doing in order to address them.

In your first sprint retrospective you should establish a Working Agreement that outlines your expectations for how team members will contribute to this meeting.

Activity: Working Agreements

A working agreement is "a set of behaviors that will support the team in having productive discussions" - *Agile*

Retrospectives.

It is much easier to point to a violation of this agreement than to tell someone they are doing something wrong, when no rules have really been defined.

Furthermore, your team is more likely to follow these rules than a list of behaviors we could tell you to do here, because you came up them yourselves and you agreed to them.

Coming Up With A Working Agreement

In order to develop a working agreement:

- Come up with a list of candidate working values that group members feel would lead to more productive discussions during this process. Examples:
  - "Everyone gets a voice",
  - "We are trying to find ways to improve, not trying to criticize",
  - "Solutions should be as simple as needed to resolve the problem".
- The ideal number of values is roughly three to five. Having more than seven is hard to remember and follow.
- If there are too many values, use the "finger method" voting.
  - Three fingers indicates "We should keep this",
  - Two for "I will go with the team"
  - or One for "I don't want this value".
- Resolution to the voting process works the same way as the estimation process, have the highs and lows discuss and try to come to an agreement in a few rounds of voting.

As you develop these values with your team, note that the values that come up are often clues as to the anxieties your team members have about the team dynamic.

Coming up with a working agreement takes a little time on the first retrospective, but is reusable and adaptable for all of the future ones.

Using the Working Agreement

Near the beginning of the retrospective, teams should review their working agreement. If some values need to be modified, deleted, or added, this is a good time to do it.

Working agreements help keep the entire team responsible for maintaining collaborative behavior, not only the retrospective leader.

Throughout the meeting, it is the responsibility of any team member to mention when one of the values is being compromised. This is easiest to do by pointing at the value on the written agreement if it is written down. This will remind the teammate that they are violating something that they agreed to.

Gather Data

At the end of the sprint, each team member has their own idea of what happened throughout.

The point of this phase is to use concrete evidence as well as documentation of the team members subjective experiences in order to create a shared understanding of what happened during the sprint. In order to improve, we need everyone on the same page.

Start with concrete facts:

- What features or stories were attempted?
- Which were completed?
- How many story points were done?
- How many pull requests got accepted?
- How many pull requests got rejected?

Try to refer to team documents such as the Trello or Sprint Check Ins in order to get this information.

The Burndown Chart

A burndown chart is a powerful tool for keeping track of the team's velocity (the amount of work completed each sprint) throughout the project.

Essentially, it plots the amount of story points remaining in the project over time (usually between Sprint Retrospectives), see Figure 1.

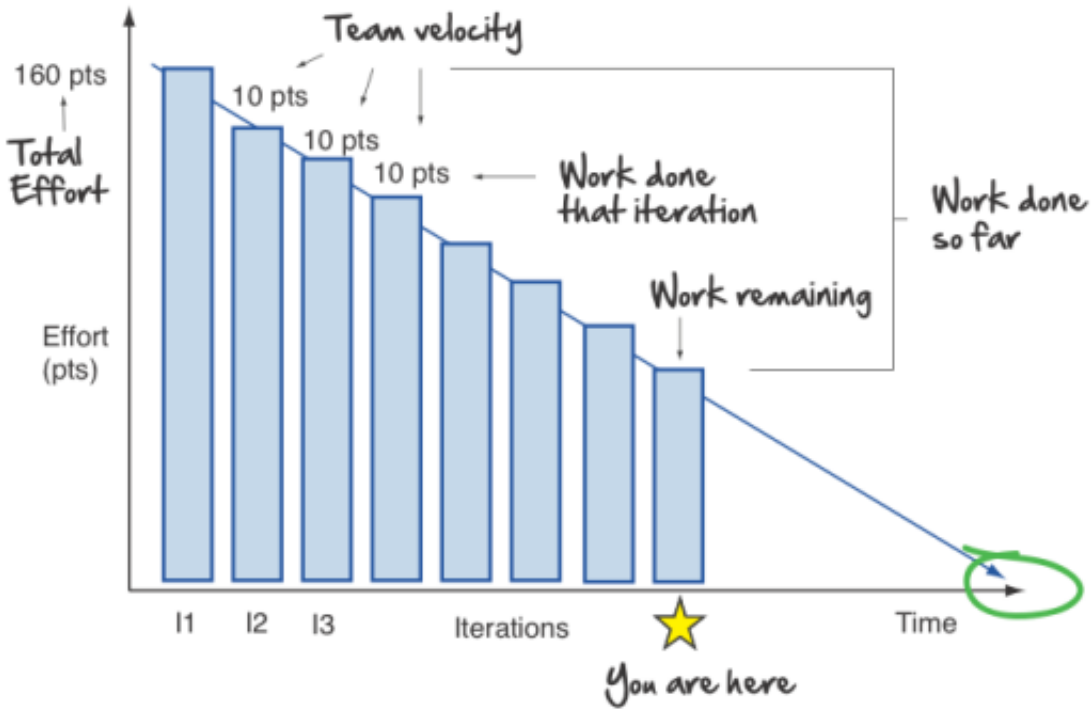

Figure 1: Ideal Burndown Chart (source: The Agile Samurai by Jonathan Rasmusson)

The burndown chart allows the team to quickly see how much work is done each iteration, how much work is remaining in the project, how much of the project have been completed, and the expected completion date of the project.

A more realistic burndown chart can be seen in Figure 2 where the team velocity varies from week to week for a variety of reasons.

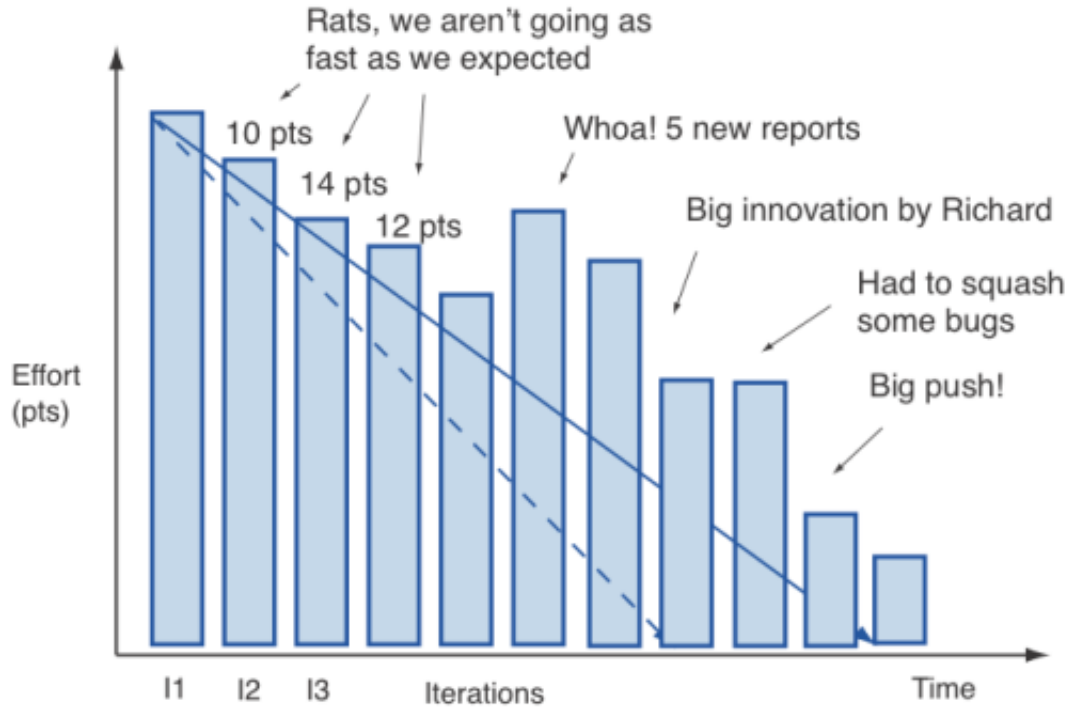

Figure 2: Realistic Burndown Chart (source: The Agile Samurai by Jonathan Rasmusson)

Even so, by comparing points in the burndown chart, we can easily see the pattern of team velocity and use that to make future conditions.

Sometimes we might see that the number of remaining story points went up, because new features were added to the backlog, or a large number of hard bugs were found. These, along with any other strange changes to the team's velocity would be important things to discuss in the sprint retrospectives.

Feelings

While data is important, when dealing with team improvement, people's subjective feelings are just as crucial for making decisions for improvement and correction.

During the data gathering phase, we are not just establishing what happened, but also finding out how people perceived what happened, and getting our team members to open up about their experience throughout the sprint.

Agile Retrospectives states that:

> Creating a structured way for people to talk about feelings makes it more comfortable to raise topics that have an emotional charge. When people avoid emotional content, it doesn't just go away; it goes underground and saps energy and motivation. Or the emotion may come out in a flare of anger, and a

> flame war won't help your retrospective.

Activity: Mad, Sad, Glad

In this activity from *Agile Retrospectives*:

> "Individuals use colored sticky notes to describe times during the project where they were mad, sad, or glad."

Have one color of sticky note for each emotion.

Have teammates take a few minutes to write down the events on the various notes. Use 1 note per specific event.

When done writing, cluster the notes on the whiteboard by how related they are to each other.

If there are any clusters, come up with a name for them and circle them.

Once finished, discuss the following questions:

- What stands out for you as you look at these notes?
- What is unexpected about these cards?
- What was difficult about this task?
- What parts felt positive?
- What patterns do you see in the clusters? What do those patterns mean for the team?
- What does this suggest as next steps?

More information about the Mad Sad Glad Activity:

https://miro.com/templates/mad-sad-glad-retrospective/

Moving to the Next Phase

Once you have gathered both feelings and objective data, discuss with your team any patterns, changes since the last retrospective, or surprises that any of you have noticed.

This is the beginning of the next phase, generating insights into your team process.

Generate Insights

Agile Retrospectives states that:

> Now is the time to ask "Why?" and begin thinking about what to do differently. When generating insights, the team considers the data to identify strengths and issues from the previous iteration.
>
> Lead the team to examine the conditions, interactions, and patterns that contributed to their success. Investigate breakdowns and deficiencies. Look for risks and unexpected events or outcomes.
>
> It is easy for people to jump to solutions once problems emerge. First solutions may be correct, but often they're not. The work of this phase is to consider additional possibilities, look at causes and effects, and think about them analytically. It's also a time for the team to think together.
>
> These insights help the team see how to work more effectively - which is the ultimate goal of any retrospective.
>
> Generating insights allows the team to step back, see the big picture, and delve into root causes.
>
> When you skip generating insights, your team may not understand how events, behaviors, and circumstances affect their ability to develop software. Time spent generating insights helps ensure that when your team plans an improvement, it's one that will make a positive difference.

Activity For Generating Insights: Five Whys

Over the course of the previous steps, we created a shared understanding of what happened during the sprint.

This activity is meant to get at the underlying reasons why those events occurred in order to expose things the team might want to address.

The process is simple, bring up an event that occurred to the team. Ask why it happened.

When the team comes up with an answer, ask why the answer happened.

Repeat the process until about 5 why's have been asked for a particular event.

For example:

```
"We didn't finish the "Play against a friend story that was in our sprint goal?"

"Why?"

"It turned out to be a lot more work than we expected"

"Why?"

"We didn't get to the networking experiment in Sprint 0"

"Why?"

"We had to complete the basic Unity tutorials first"

"Why?"

"There's a lot more to learn about Unity than we initially expected"
```

Because of this in the next stage, the group might decide to refocus on making sure everyone is grasping the basics of Unity and that a networking experiment is assigned to everyone in the next sprint.

More information on the "5 whys" Activity:

https://kanbanize.com/lean-management/improvement/5-whys-analysis-tool/

Decide What To Do

At this point you should have a variety of ideas for how to improve your team processes in the next sprint.

During this phase, we should prioritize these ideas by how likely they are to make meaningful change, and pick 1-3 ideas that are most important to our team.

Too many initiatives can overwhelm your team, and so it is important to only pick a few here.

It might be useful to use the "finger voting" method provided previously to have your team come to a consensus on initiatives.

Activity: SMART Goals

Establishing SMART goals is a common business method of stating goals in a way that makes them easily understood and acted upon.

SMART is an acronym that stands for:

- Specific - Says exactly what must be done to accomplish this goal.
- Measurable - How are we going to tell if we met the goal?
- Actionable/Accomplishable - Can we complete this goal with what we know now?
- Relevant - How does this goal align with our larger goals? Why is doing this important?
- Time-Bound - What is the time frame for accomplishing this goal?

Describe your 1-3 initiatives as SMART goals, with a written description and a plan for how to achieve them, and measure their progress.

For more information on SMART goals and plenty of examples:

https://www.ucop.edu/local-human-resources/_files/performance-appraisal/How%20to%20write%20SMART%20Goals%20v2.pdf

Close The Retrospective

Letting the retrospective go on forever can make people become bored and antsy. Once a small set of actionable items has been created, it is time to begin wrapping up the retrospective.

Make sure that any key insights and actionable items are recorded, and make sure you thank everyone for their work both throughout the sprint and during this retrospective.

Just like everything else in Agile, we want to improve our retrospectives iteratively as well. Take a few minutes to discuss what went well, what could be done better, and what could be done differently in the next retrospective.

Retrospective Leader

A retrospective is led by a member of the development team, who helps guide the team through the process.

As the leader, the primary focus is on correctly following the process of the sprint retrospective and ensuring its goals are being met. While they are following along in the content of the meeting, they are meant to remain neutral in the discussions, even when they have strong opinions.

The leader of the retrospective will rotate in this class between the members of the development team every sprint, and should not repeat in the semester so that everyone gets a chance to do it.

The leader has the following primary responsibilities:

- Managing the Activities
- Managing Group Dynamics
- Managing Themselves
- Managing Time

Managing the Activities

The activities will not produce much results if the team is unsure how to accomplish the activity. It is the retrospective's leader to introduce the activity, monitor the activity, and lead the team in getting insights from doing the activity.

At the start of the activity, remind the team of the activity's purpose.

Next, give the instructions for the activity as they happen, not all at the beginning. Tell the team how to do the first step, have them do the first step, then tell them the second step.

Be aware of how much time each step is taking, as well as notice when the team has finished a step and is ready to move on.

Once the activity is over, lead the team in gathering insights. Ask questions like:

- What surprised you? What was challenging?
- What did this tell you about our project?
- How could this activity have been improved?

Just doing the activities without reflection will result in very limited results.

Managing Group Dynamics

One of the primary roles of the retrospective leader is encouraging participation from all members of the team. We want to make sure that "people who have something to say have a chance and people who have a lot to say don't dominate" - *Agile Retrospectives*. It is important to ask questions of quiet team members to encourage them to share their opinions, although sometimes we must be willing to accept a "I don't have anything to say" from shy members.

If there are people on your team who overwhelm the conversation every meeting, contact them before the meeting and make sure that you communicate that you need to hear from everyone on the team, and one person primarily talking can make other members retreat to the sidelines.

During the retrospective, if someone is overwhelming the conversation, hold your hand up, reiterate that we need to hear from everyone in a neutral tone, and ask questions of those who haven't had a fair share of

speaking time.

Other than participation, blaming and violation of working agreements are common retrospective problems.

During a retrospective, having blame being thrown around can start a cycle of confrontation that can completely derail a meeting. Look out for "you" statements like "You didn't create the pull requests properly" and encourage them to be turned into "I" statements like "I am having trouble merging these pull requests because of the lack of standardization". These can focus the team on issues to fix without it turning into a yelling match.

If yelling does start to occur, it is important to intervene immediately. Tell the person that the group wants to hear what they have to say but can't if they are shouting. Calling attention to the behavior is usually enough to stop it, but if it continues you may need to call for a break or get assistance from your instructor.

Violation of working agreements also need to be dealt with immediately or the working agreement will be seen as not enforced and lose any value. When violations happen, remind the team of the working agreement.

Managing Yourself

If you are feeling overwhelmed with managing all these things, feel free to call for a short break. Ask yourself "How did the group get here?" and "Where does the group need to go next?" in order to refocus your efforts. Over time, your comfort in dealing with these situations will grow, and this class is an opportunity to introduce yourself to them with low professional stakes.

Managing Time

Leading a retrospective is a difficult balance between paying attention to the time and managing the needs of the group at the same time. If something important is happening, it is best not to interrupt, but at the same time the goals of the retrospective need to be met. If you are running out of time, it might be best to put pins in some discussions and return to them at a later time, especially if they are getting too detailed.

Summary

Throughout this reading, we discussed Sprint Retrospectives, their role in the development lifecycle, efficient processes for doing them, and ways to structure their activities.

In a Sprint Retrospective we want to:

1. Set the stage

2. Gather data
3. Generate insights
4. Decide what to do
5. Close the retrospective