Project: Personal Sprint Reflection

1 Purpose

The Personal Sprint Reflection's purpose is to assess and reflect on both what you achieved individually and what the team accomplished overall during the sprint. This information, along with the information the team provides, will be the basis for your individual sprint grade. It is important that you provide accurate, honest, candid, and complete information. Your specific comments about the team are private to the teaching team, and will be generalized for any performance considerations. You should highlight issues/concerns **AND** good things/pleasant surprises. Both are important to this process.

Your submission will contain two sections: one based on your role (see Section 2 or 3) and one for the team (see Section 4). It should be one (1) to (3) pages long.

Here is what we will be looking at:

1. How well did the team's development process/workflow work?

   How did they adapt the models presented in class to optimize their development/capability to deliver? How did they divide responsibilities? How did they proceed from defining requirements to design, testing, and development? How did they test their code to assure its quality?

2. How good is the code?

   How well is the team doing producing clear, concise, extensible/maintainable designs and code? How much change to the design or code base occurred because of requirement changes or feedback from delivery?

3. For the scrum master, were they effective?

   Did they keep the process under control and moving along? How did this influence the responsiveness of the team and its effectiveness in meeting their goals?

4. Effective use of technology for development.

   How did the team (and you) use technology to aid them `Jira`, `GitHub`, `Sonarqube`, *et. al.*? How did they adapt or adjust the technology and how do these changes help the process achieve its goals?

5. Teaming

   Did everyone contribute? Did everyone make a difference?

6. What the areas needing improvement?

7. Accomplishments.

   What did they individually and their team as a whole accomplish?

8. Were they able to tie in course concepts from lectures, readings, or their own experiences outside of class? This applies to both the process and the code.

2 Reflection for the Developer Role

Your objectives:

1. Break-down the sprint goal and the product backlog into a sprint backlog.
2. Retire items on the sprint backlog, meeting quality and schedule commitments.
3. Ensure the *team* is successful as well as the sprint delivers a *Done Increment*. Team success includes ensuring everyone contributes, everyone is challenged, and everyone grows. While it is important that deliveries are made, we will view deliveries that involve everyone as more valuable than deliveries that come from individual heroics.

The major points you should assess are:

1. Your contribution to the team's direction.

   We want to get a sense of how you contributed to the the sprint backlog. How did you influence the backlog? Did you raise issues, influenced the discussion, and why (or why not) your thoughts were adopted. Did you contribute to others' ideas.

2. Your specific technical contributions.

   We want to get a sense of what you contributed technically—meaning both what you did and how well you did it. You can pull `Jira` tickets, `Github` commits, `Sonarqube` metrics, or anything else you think relevant. If you worked on something as a pair, identify that partner. There are a lot of ways you can go when you talk about some bit of work. Here's a rather lengthy list of ways you might go to highlight the value, challenges, or whatnot you encountered, overcame or delivered.

   Complexity Did the tasks tend toward being complex or straightforward—remember, not everything needs to be hard. Why was it complex or straightforward? Did it seem this way from the start or did you realize it as you went forward? What influences occurred that changed your mind? How did you solve the problem?

   For example, what made something hard might have been a requirements issue—the requirements weren't good somehow and you had to work to figure this out and then get it right. Or the requirements might have changed and you had to do a massive refactoring of the design.

   Type of Work Was the task feature development, infrastructure, or something else?

   Quality Is the code or documentation a good job for testing, structure, documentation, and style? Is it understandable and maintainable? Did you find a general solution or is it a bit of a hack?

   Why should anyone believe the code was functionally tested sufficiently? Which structural test criteria were used and what level of satisfaction was attained?

3. Your contribution to the team.

   Did you participate in team activities such as standups or team meetings. If you couldn't attend, please be upfront about why. We recognize that you have other pulls on your time such as other classes, jobs, and job searches.

   How active were you in communicating with others? Were you more solitary in getting your work done or more engaged with others? Not everything needs a group hug, of course. Some things do; some don't.

   Did you receive mentoring or help from someone else? Did you provide mentoring or help to someone else?

3 Reflection for the Scrum Master Role

Where the *developer* role is perhaps familiar, the *scrum master* role may be new. This role is to facilitate all things related to the team: sprint planning, identifying people who are stuck and then helping them find ways to become unstuck, checking in with people and keeping them generally on task, interfacing with product team, devising initiatives to optimize the team or for future development. You are the guide, not the boss.

In this project, the person serving as scrum master is also a developer when possible. That is, your first responsibility is scrum master. If you have time, you become a developer.

Your core objectives are:

1. Keeping the team on track. The scrum master works with the product owner and the team to decide on a sprint commitment and to help the team deliver on its goals. It's an important role as you are essential to getting the momentum going and staying on course. It will take real commitment for everyone to start working in a pattern where progress is made by culture (meaning, when no one is looking, people do what's needed and do it correctly).
2. Facilitate the sprint's lifecycle, beginning with sprint planning, during which the team estimates each item in the backlog (can be with the product owner), often using story points. See below if story points aren't familiar to you.
3. Remove impediments for the team.
4. Report on progress. Keeping the team informed whether it is ahead or behind on the sprint commitment.
5. Watch for scope creep.
6. Keep the stakeholders informed.
7. Contribute as a developer—to the extent possible given these responsibilities.
8. Not be *the boss.* It is NOT the job of the scrum master to create work and assign work. These are the team's responsibility. The scrum master is responsible for insuring everyone is contributing to the backlog and people are signing up for tasks. The ideal situation is where team members are grabbing tasks. The scrum master needs to insure folks are taking work and retiring work, removing obstacles as necessary. Now, if you have a meeting and people are divvying up work and one person happens to be doing Jira, fine. But what shouldn't happen is the scrum master acts as the boss and ASSIGNS work. The scrum master should keep the team focused on the sprint goals and raise issues/questions about the work being taken. The team owns work assignments, not the scrum master.

Here are the major points you should address as scrum master.

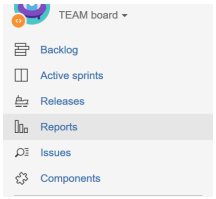
1. Discuss the Sprint Planning process. What were the sprint goals? Did the team settle on the sprint goals quickly or easily, or was there discord or diffusion about planning the sprint. How did you facilitate getting to a decision. Did this affect the technical backlog—and if so, how?
2. Discuss how communications flowed with the Product Owner (TA or professor) during the sprint.
3. Discuss how team communications flowed during the sprint. How many meetings were held and what was the purpose for them? Did you conduct stand-ups of some sort? How did decisions get made, communicated, and recorded?

4. Discuss obstacles the team faced in achieving the sprint goal. How did the team identify them? How did they get addressed? Were they resolved? What role did you play in any of these steps?

5. Discuss the changes were handled.

6. Discuss the team's progress and how to monitored progress.

   Use Jira reports to support your claims. Jira reports are available via the report icon on the left side of the project page. There are three charts that may be helpful.



   (a) *Burndown Chart*: tracks the total work remaining and project the likelihood of achieving the sprint goal. This helps your team manage its progress and respond accordingly.

   (b) *Sprint Report*: helps you understand the work completed or pushed back to the backlog in each sprint. This helps you determine if your team is overcommitting or if there is excessive scope creep.

   (c) There is a third chart: *Cumulative Flow Diagram*: shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.

   Don't merely include the charts. You must discuss them.

Sidebar on Story Points

A Story Point estimates relative difficulty of a backlog item, and usually indicates how big or small a given item is. The base unit is often determined by considering a reference item, and then scaling from that. It is not supposed to be equivalent to time (e.g. hours), but some do match time to points. Because it's relative, the relation of story points to work is specific to a team. Early on, you should not think about story points as fine-grained estimated. For example, 4 versus 3 likely isn't much different, except that something about the backlog item that gets a 4 seems just a bit harder than a 3 to you. A 3 versus a 5 or a 6, now that indicates there's some separation for how hard you think the backlog item is. Early on, the trick is to make qualitative estimates up front and then reflect during the Sprint Retrospective on how well the estimations worked. That is, once you completed the sprint, consider how hard the backlog item actually was and compare it to the original estimate. You should also compare it to the other backlog items the team completed and note the adjusted values. Use these revised numbers for the next Sprint Planning estimations. When someone now estimates 3 story points, you can ask them, how does this backlog item compare to the last 3 story point item. Also remember, story point estimations will vary among team-mates. So one person's 3 may be another person's 5, at least early on. Remember, everyone is learning her own scale and learning everyone else's scale. Over time, people will normalize their estimates—unfortunately this usually takes more time than we have for this project. In some sense, this is a form of *reinforcement learning*, which you may know from a Machine Learning class.

4 Reflection on the Team

Assessing work, teaming, and the process is an important part of creating and maintaining high performance teams. Assessing the project means being critical, which means identifying the good, the average, and the bad objectively, fairly, and without emotion. Remember the point of this critique is achieve or maintain a high caliber team.

The good	It should be acknowledged so it can be reinforced. Good can be pleasant surprises, mentoring, jobs done exceptionally well, extra effort, *etc.*
Average	Average or simply solid work is just that, solid work. It is valued and appreciated. But, this is what is *expected.* Don't feel the need to overblow how you acknowledge the *expected.*
The bad	If work product and respect for the Scrum Values[1][2] of an individual are not meeting expectations, it is your obligation to the team to address it.[3] This does not mean vent. The team needs to acknowledge the issue so it can be addressed. Bad can be unpleasant surprises, non-communicative team-mates, non-contributing team-mates, or team-mates who produce sub-par results. In sum, they are net drains on the team. Keep in mind being contrary is not necessarily bad. Providing alternate views is important. However, at some point everyone needs to get on board and get behind the team. Beyond this point, being contrary or intransigent (especially a *told you so* attitude) is not helpful.

You should rate and comment on your team-mates' work. You should also rate your own work to provide a sense of calibration.

Your input is a serious statement and does affect grading. For this process to work, you must be honest, candid, and objective. Your ratings and comments are **confidential** to the teaching team. You must rate and comment on each teammate.

You should create a table with a column for each person's name, rating, and comment. The rating should use a scale from 0 to 10, where

0 means *Bad/Non-existent*,

5 means *Average/Solid*,

10 means *Way above and beyond.*

Here's an example:

Name	Rating	Comments
Groucho	7	Very good work. Witty beyond expectations. The main backlog item was he shot an elephant wearing his pajamas. Groucho did it in line with expectations although we're not sure how the elephant got into Groucho's pajamas. One thing that could be improved is his treatment of Mrs. Dumont; it could be more respectful.
Chico	3	When Chico did work on the backlog, he did solid work. For example, his piano solo hit on all the notes. But, Chico did not meet all his sprint expectations. It seems he spent too much time playing cards. He also got argumentative about what to do when he reached the viaduct. While I understand he was frustrated, he wouldn't let this go. He kept questioning why the viaduct way past the point of being reasonable.

Harpo	8	Amazing work. Took on some of Chico's missed backlog items without saying a word boo about it. He did communicate when he did it and tried to connect with Chico before assuming his work. Added a nice touch with the horn; it was an inciteful idea. The black hat was a stylish touch. Solid flute playing. Destroying the piano was not helpful as it enabled Chico. But, using it as a harp made the best of the situation.
Zeppo	5	Straight down the middle. Did what was expected. Solid contribution.
Minnie (me)	6, maybe 7	I served as scrum master. We did hit on all the objectives and I believe overall I did a solid job. I believe I did a little above and beyond because it was difficult to keep this crew focused. Keeping things moving was a consistent challenge. While projects are always changing, there were times where the problems required extra effort. For example, there was a major logistics problem removing the tusks from the elephant Groucho shot. Groucho couldn't figure out how to make it work. Neither did any of the others nor did our IT group. I searched on google to find leads on how to do this. I did see a firm from Alabama who claimed it could do it. So, I spent a lot of time arranging the connection and then worked with that firm to find an answer (I was the developer as Groucho had moved on to another backlog item). The solution was to move the elephant to Alabama where the tusks are looser.

If this all seems ridiculous, these are bits taken from the Marx Brothers, who made comedies back in the black and white days of film. Groucho, Chico, and Harpo were the funny brothers, and Zeppo played the straight man. Minnie Marx was their mother and manager. These bits were taken "Animal Crackers" (`https://youtu.be/gPSAu8xfmhk`and "The Cocoanuts" (`http://www.metacafe.com/watch/4493024/the_cocoanuts_why_a_duck/`).