# 2021

Changes in RED+: Tim Storer’s walkthrough

1

a) A retrospective focuses on the **process** of the work that the team has done, i.e., how the team worked together or separately to solve the issues they experienced and implement the features agreed with the customer. Retrospectives are about finding problems in the workflow/software process and thinking of ways to resolve them. In contrast, sprint review meetings are about communicating with the customer to decide what features and fixes for the **product** to focus on in the **next sprint**, i.e., the *what*, not the *how*.

b) The problems:

1. Not all team members were **present**.
   1. Make sure to find a time when everyone is **available**, including Robert.
2. Sticky notes were given to everyone **in turn**, and only one for each.
   1. Give *multiple* sticky notes to everyone at the start to allow for dynamic addition to the board (because of diversity of speed and amount of issues team members can think of)
3. Encouragement of short retrospectives (**10 minutes**) reduces team’s motivation to bring up issues, therefore damaging morale
   1. Encourage everyone to speak, don’t try to “keep it [the retrospective] short”, make sure everyone is happy with what was discussed before ending the retrospective. / Explicitly set a specific time, e.g., 60/90/120 minutes.
4. Some issues are repeated, highlighting lack of engagement with what the other teammates are adding to the board
   1. Encourage diversity of issues raised, possibly including how to achieve the issues
5. Unequal work is given to team members when assigning overdue tickets to them (because of the different estimated times)
   1. Try to re-estimate their completion times given current context, analyse why they weren’t completed, redistribute the tickets according to time, e.g., give *Configurable notifications* to multiple people
6. A teammate’s work was discussed without them being present because they did not want any “**embarrassment**”.
   1. Encourage talking about teammates’ contributions and sharing any **concerns and praises**.
7. The topic of the retrospective is chosen by the leader
   1. Topic should be chosen after all the issues have been raised, don’t constrain people’s contributions. Remind everyone to bring up issues about everything.
8. Retrospective format chosen by one person not elected to do so.
   1. Encourage discussion between all teammates of best format. OR explicitly elect a teammate to act as Scrum master.
9. Retrospective is stopped after data gathering step is complete.
   1. Continue analysis of the issues brought up.
10. Overloading team – additional work on “Not committing broken code” and overdue issues on top of next sprint issues.

c) The “5 Why’s” method:

1. Why was there a delay in the project schedule?
   1. Because there were 3 tickets overdue.
2. Why?
   1. Because they took too long to do.
3. Why?
   1. Because the tickets have a very long estimated completion time.
4. Why?
   1. Because they are too general.
5. Why?
   1. Because they were agreed as issues by the team/customer.
   2. Because they weren’t broken down into chunks that could be delegated better.

2.

a) INVEST:

* Independent: Not very dependent on any other user stories, it includes the only other story in itself. Maybe not included, but depends on first?
* Negotiable: Rather negotiable, as it identifies the user as a patient with asthma. Not negotiable – too much detail.
* Valuable: There is very little motivation in the story, only “monitoring” mentioned as a valued outcome by the user. Good justification.
* Estimable: Since there is an over-abundance of features listed, it is **impossible** to estimate it.
* Small: There are way too many features listed, resulting in a **huge** user story.
* Testable: The workflow is described pretty well, which means that a very large test could implement the features mentioned; however, it would be beneficial to separate this into multiple small tests. Quite difficult to test automatic entry into app from meter reading.

b) High priority user stories:

1. As a clinician, I want to review a patient’s record so that I can understand their medical history better.
2. As a clinician, I want to receive a notification as soon as a patient records a peak flow below 300 for 2 days in a row so that I can react quickly and treat the patient who might be having problems.
3. As a patient, I want to have the option to record whether the reading was taken before or after medication and exercise so that I would give more contextual info to my clinician and not cause unnecessary worry.
4. As a user, I want to access information easily and quickly so that I don’t waste time on the application but spend it more productively (e.g., resting as a patient/treating other patients as a clinician).
5. As a patient, I want my reading to be entered into the app automatically so that I don’t waste time and make mistakes while entering my reading manually.
6. As a patient, I want to be able to add an optional note to the reading so that I explain to my doctor the context of the reading.

c) Personas:

[Include: needs and desires, what they are trying to achieve]

1. Sarah Smith, 35, clinician specialising in asthma. Recognises benefits of technology and wants to implement app to treat patients better with everyone’s individual situation.
2. Pete Park, 24, asthma patient who has been having difficulties lately without knowing why. Degree in Computing Science helps willingness to try out new technology.

d) MOSCOW:

* Must have features:
  + 0. (Connecting app to meter). Without this feature, the app would be too similar to a paper-and-pen system used beforehand, which would make the development of such an app not worth it.
  + 2. (Notification). Without this, clinicians would miss possibly life-threatening situations to the patient.
* Should have features:
  + 3. (Context). Very valuable information, but not needed for the bare-bones skeleton of the functionality.
  + 4. (Usability of UI). Important for being able to use the app quickly and effectively, but the app is functional without being the friendliest of UIs.
  + 5. (Automatic reading).
* Could have features
* Would want features:
  + 6. (Optional note). The other context story is possibly enough context for a clinician, no need to add more.

Be more willing to put features in Could have/Would want based on 25-50% budget

3.

a) Architectural pattern – shows reusable solutions to whole system architectural problems. Design pattern – addresses design problems at the scale of class interactions.

b) **Pipe and filter** architectural pattern because:

* search/query **filters** would be easy to implement as **Filter** components;
* different time/resources can be allocated to different filters/searches based on how difficult they are.

c) I would choose these specific tailoring aspects:

* *Pull* data flow because the data does not change drastically enough to keep a continuous flow of all repositories and their changes. This way, filters can be more effective because they use output from other filters more without worrying about the original data source.
* Concurrent pipeline? Researchers can see some repositories immediately to start analysis on them.
* Reorderable filters to allow researchers to customise them as required.
* Branching filters to allow more potential analysis of the repositories by the researchers.

d) NO. Remember – **prepare pen(cil) & paper to draw** !!!

e) The GitLab data **source** could be **added as another**, parallel source going through the same filters? Alternative: generalise source class.

f) No because:

* redundant behaviour since filters are already plug-in
* results are meant to be consistent between researchers;
* would be in danger of exhibiting the inner platform effect, better to use it for extension of pipe and filter.

4.

a). 14 distinct. All “Given”, “When”, and “Then” parts of sentences translate as steps, as well as “And” steps, which are synonyms to their previous step.

b) Scenario: Recurring reminder for individual  
Given a RemindMeBot in the channel General,  
And the date is 22nd January 2021,  
When I say “Remind **Kerry** every Monday to attend the standup”,  
Then RemindMeBot responds “I will remind **Kerry** every Monday to attend the standup beginning 2021-01-25”

c) Maintenance improvements:

1. Since all scenarios start with the channel General and 22nd January as the date, avoid repetition by putting these steps in a **Background**
2. Scenario outlines more specific
3. **Example tables**
4. Steps should be shorter
5. **Parametrise steps**? General “When I say remind me next \_\_\_ to \_\_\_”
6. Step parameters

d) Disagree (either way) because:

* coverage could be **indicative** of how well scenarios/features are tested

# 2020