

This document explains how Cougs in Space is performing project management based off of the Agile philosophy using GitHub's project management features.

Project Management

Based off of the Agile Philosophy

Revision: 1.0.2

Bradley Davis



Table of Contents

1 Agile.....	2
2 Backlog	3
2.1 GitHub Issues	3
2.1.1 Creating an Issue.....	3
2.1.2 Removing an Issue	5
2.1.3 Completing an issue	5
2.2 Best Practices	6
3 Sprints	7
3.1 Planning.....	7
3.2 Execution.....	7
3.2.1 Stand ups.....	7
3.3 Review	7
3.4 Retrospective	8
4 Metrics	9
4.1 Issue Points	9
4.2 Burndown chart	9
4.3 Team Velocity.....	9

1 Agile

Agile is a philosophy for managing projects and teams. The Agile manifesto list these values:

- *Individuals and Interactions over processes and tools*
- *Working Software over comprehensive documentation*
- *Customer Collaboration over contract negotiation*
- *Responding to Change over following a plan*

Cougs in Space is basing their project management off this and implementing the following Agile practices:

- Maintain a prioritized [backlog](#) of work ([GitHub issues](#)) with a single backlog manager (CTO)
- Breakdown work into manageable chunks ([Small issues](#) and Sprints)
- Structure your stories so they depend on each other as little as possible
- Try to use relative sizing to size up work ([Issue points](#)) rather than actual concrete amounts of time
- Have bi-weekly [standup meetings](#) so everyone knows what going on and how things are progressing
- [Fixed timescales](#) and variable requirements
- [Measure](#) the team's velocity and use it to estimate work

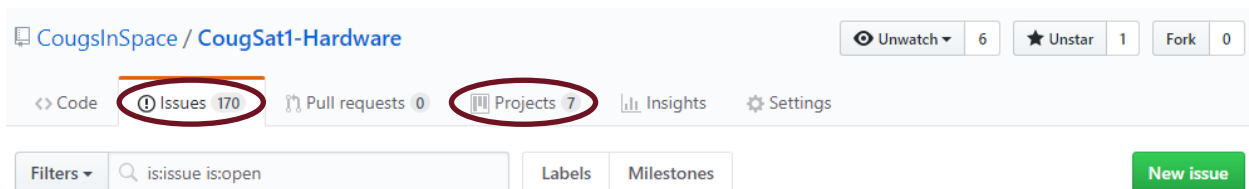
These practices are best explained in context in the following section.

2 Backlog

A backlog is an accumulation of uncompleted work that needs to be dealt with. Cougs in Space utilizes GitHub's issue features to create and organize this backlog. It is the backlog manager's duty to ensure the backlog is properly created and maintained. The current CTO is acting the role of backlog manager.

2.1 GitHub Issues

GitHub issues are part of GitHub's project management features. They are individual tasks that need to be completed that are relevant to its parent repository. The issues can be accessed by visiting the issues page in a repository, see below. Issues are also visible in the projects tab under the associated project.



2.1.1 Creating an Issue

The follow steps are followed when creating an issue. Start at #1 and read carefully and fully before advancing to the next step.

1. Is this actually an issue?
 - a. Does it reference work directly related to creating the satellite or ground station? [YES = Advance] *Consider resources repo, contact backlog manager*
 - b. Is it just a news / notice posting? [NO = Advance] *Consider Slack or website post*
 - c. Are they instructions on getting tools setup? [NO = Advance] *Consider creating a SOP document, contact backlog manager*
2. Open new issue editor (Green button in the appropriate repository, see image above)

The image shows a Jira issue creation form. The 'Title' field is at the top. Below it are tabs for 'Write' and 'Preview'. A rich text editor follows, with a placeholder 'leave a comment'. To the right of the editor are four dropdown menus: 'Assignees' (set to 'No one—assign yourself'), 'Labels' (set to 'None yet'), 'Projects' (set to 'None yet'), and 'Milestone' (set to 'No milestone'). At the bottom right is a green 'Submit new issue' button. A footer note says 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.' and another note says 'Styling with Markdown is supported'.

3. **Create a title**, see image above:
 - a. Start the title with the name of the project (i.e. STRUCT, PMIC, IHU, AVIONICS, etc.)
 - b. Use a descriptive, unambiguous title
 - c. Consider appending a version number if appropriate (i.e. Update Schematic V2.1)
 - d. Use title case
4. **Add a comment**, see image above:
 - a. Required if the issue requires more information to explain it then just the title (normal people cannot read your mind)
 - b. Use common sense to decide what to write
5. **Assign someone(s)**, see image above:
 - a. If there is someone(s) in mind that should complete this task, add them
 - b. You may leave this empty and wait for it to be part of a sprint to assign it to someone
6. **Add labels**, see image above. *Refrain from creating new tags, ask the backlog manager*
 - a. **Difficulty:** (Required) Choose between beginner, intermediate, or advanced. Be accurate.
 - b. **Project:** (Required) Add one or more project tags. Hardware projects are based off the PCB or assembly. Software projects are based off the processor being written for.
 - c. **Task Type:** Choose between Design, Docs, Code, or CAD
 - d. **Category:** Choose between Tests, Refactoring, New Feature, Enhancement, or Bug Fix
 - e. **Priority:** Add Critical if the issue needs to be addressed first. It will be added to the next sprint. If required, ask the backlog manager to add it to the current sprint.
7. **Add it to a project**, see image above
 - a. CougSat-1: This project is located under organization

- b. **Relevant Projects:** These should mirror the project labels. Hardware projects are based off the PCB or assembly. Software projects are based off the processor being written for.
8. **Do not add it to a milestone**, see image above
 - a. **Milestones are sprints** which the backlog manager will create and add issues to it.
9. **Review the information**, deleting issues is impossible so be right the first time.
10. **Submit new issue**, see image above

2.1.2 Removing an Issue

If an issue needs to be removed for any reason, see below,

- A duplicate issue was created
- Issue references something that is no longer being developed
- Accidental creation of an issue


A certain process needs to be followed as GitHub disallows any issue from being deleted. That process follows:


1. **Add the Abandoned label**
2. **Add a comment** why the issue is being abandoned
3. **Comment and close**

2.1.3 Completing an issue

The preferred way of completing an issue is to create a pull request of the changes and list in the description which issues are completed, see below. The keyword *Fixes* is recognized by GitHub and will automatically close those issues when the pull request is closed.

EPS Mechanical Layout #101

 **Merged** WattsUp merged 5 commits into `master` from `eps` 21 days ago

 Conversation 1  Commits 5  Checks 0  Files changed 18



WattsUp commented 23 days ago • edited ▼

Member



Fixes #102

 Fixes #70

 Fixes #68

For issues that do not have files to upload to complete the issue, upload the files. Most issues (read all) will have files associated with them, please share these. If you really do not have any files to upload, in the issue, comment

explaining the results of the issue. Consider including how long did it take, were there any difficulties, any external links (discouraged except to a photo in our Google Drive), etc. Then hit comment and close.

2.2 Best Practices

These are the recommendation when creating an issue

- **Separate the issue** into as many little issues as possibly, stay reasonable
 - More little issues are easier to show progress and assign then one massive issue
- **Keep the difficulty sized appropriately**
 - If an issue is simple but takes a long time: make it beginner and split the issue up
 - If an issue is super complex: make it advanced and split the issue up
- **PCB issue flow**, most (read all) PCBs will require the following issues
 - Design block diagram Vx.x
 - Create schematic Vx.x
 - Review schematic Vx.x
 - Draw board outline Vx.x
 - Place components Vx.x
 - Board layout Vx.x
 - Finalize board (Silk screen and gerber files) Vx.x
 - Review PCB Vx.x
 - Assemble board Vx.x
 - Test circuits Vx.x
 - Create an individual issue for each circuit that needs testing
 - Integrate into EM Vx.x
 - Assembly flight ready board
 - No version number as we only do this once
 - Integrate into FU
 - No version number as we only do this once
- **Mechanical issue flow**, most (read all) mechanical components will require the following issues
 - Design the component
 - CAD the component
 - Fabricate the component
 - Assemble the component
 - Test the component
- **Software issue flow**, most (read all) software features will require the following issues
 - Design the feature
 - Implement the feature
 - Test the feature

3 Sprints

A sprint is a short duration deadline to complete a list of issues. Sprints range in duration from two weeks to a whole month. The life cycle of a sprint is explained below. With the timing of Cougs in Space's meetings, the start and stop of a sprint shall always occur on a Wednesday meeting. The planning of a sprint may fall on a day earlier in the week but never earlier than four days before the start of a sprint. Usually on the day of conclusion of a sprint, the Wednesday meeting will begin with the review and retrospective of the last sprint, and end with the kick-off of the next sprint.

3.1 Planning

The backlog manager reviews all of the issues currently in the backlog. They select preliminary issues and adds them to a sprint. This is done by creating a milestone in GitHub and adding issues to that milestone. The team leads, and the backlog manager review these issues and make changes if necessary. It is the team leads' roles to speak representatively of their team's abilities.

3.2 Execution

The team leads will assign issues to individuals and they are responsible for completing those issues. This period occupies most of the sprint duration

3.2.1 Stand ups

A stand up is a meeting between all of those involved to review the progress on the sprints. These are typically everyday but as Cougs in Space is a club, these will take place every meeting. The questions that are answered in a standup are:

1. What has progressed?
2. What obstacles are impeding my progress?

The backlog manager will usually present metrics to quantifiably answer the first question and then each team will qualitatively answer the both questions.

3.3 Review

Once the duration of the sprint has elapsed, the review discusses what was completed. This is usually aided with a demo, photographs, etc.

If the sprint was achieved, the completion of the sprint shall be celebrated, usually aided with sugar.

From the [metrics](#), the sprint leader will be named and given a trophy. The sprint leader is the individual who completed the most issue points in the sprint. This is a coveted position and should be attempted to be achieved by everyone.

3.4 Retrospective

The sprint is concluded by a reflection of ways to improve the sprint process. The questions that will be answered in the retrospective are:

1. What should the team stop doing?
2. What should the team start doing?
3. What should the team continue doing?

4 Metrics

These are the various metrics that can be procured from our project management system

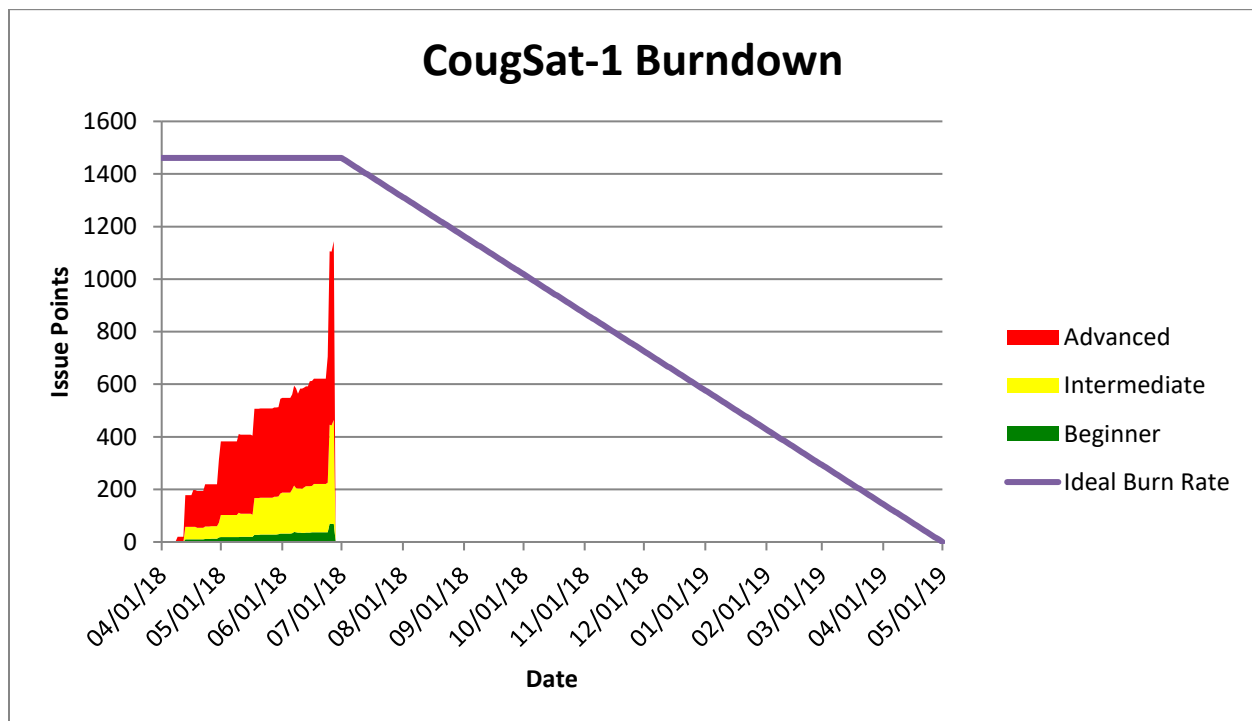
4.1 Issue Points

Each issue is assigned a number of issue points, (Agile usually calls it story points), based off of the difficulty, see below. Every member's contributions are tracked in number of issue points they complete.

- Beginner: 1 point
- Intermediate: 4 points
- Advanced: 20 points

4.2 Burndown chart

A burndown chart communicates the issues for a project, milestone, or the whole satellite with respect to time, see below. Whenever an issue is opened, the number of issue points increases, when an issue is closed, it decreases. The ideal burn rate is a line connecting where and when we started and where and when we are going. If the actual issue count is above this line, the project is behind schedule. If below, the project is ahead of schedule.



4.3 Team Velocity

From math on how many issue points are being completed per unit time, a team's velocity can be procured. This metric is useful for planning future sprints.