***RW-Project Management***

*Note: This is subject to, and encouraged, to change. However, it will be followed exactly in the first increment. It works for Lockheed, so it might work for us.*

**General Flow**

1. MVP Identification
2. Increment Planning
3. Sprint Planning
4. Task Completion
5. Sprint Wrap-Up

**Detailed Flow**

1. MVP Identification
   1. **Purpose:** 
      1. Clearly identify what specific Minimal Viable Product is sought to be produced.
   2. **Goal:**
      1. Produce names of promising MVP(s) that are both specific in nature and relatively easy to generate CAD models of.
   3. **Notes:**
      1. In order to pass this gate, a project must have a fully detailed High-Level design on paper including supporting calculations and, if applicable, weight, size, and cost estimates.
2. Increment Planning
   1. **Purpose:** 
      1. Develop the groundwork of the projects to be addressed over the next three months.
   2. **Goal:** 
      1. Specify features of desired MVPs and produce comprehensive list of stories, story scores, and story success (exit) criteria for each said feature.
   3. **Procedure:**
      1. Define every Feature for the selected MVP.
      2. Pick out desired number of Features to tackle during this increment.
      3. Define Stories for each Feature addressed in this Increment.
         1. Stories should be:
            1. Independent
            2. Negotiable
            3. Valuable
            4. Estimable
            5. Small
            6. Testable
         2. Story Examples
            1. User Stories

Card: ‘As a <user role>, I want <activity> so that <value of even doing action>

Conversation: ‘Yeah, the robot needs to lift heavy things too’

Success Criteria:

The phone is silver

The robot lifts heavy things

* + - * 1. Enabler Stories

Test <entities> in <location> because <purpose>

Do <action> for <entity> so that <purpose>

* + 1. Define Story exit criteria
       1. Exit criteria = Success Criteria
       2. When a story’s tasking is completed, the story is only closed out if the exit criteria is satisfied.
          1. Test that <criteria>
          2. Demonstrate that <this happens>
          3. Verify that when <a role> does <some action> they get <this result>
          4. Given <a context> when <this event occurs> then <this happens>
       3. Stop writing criteria when there is enough to size a story, testing will be too convoluted, or there have already been 2-3 revisions.
    2. Define Story relative points
       1. Each story needs to be scored with a 1,2,3,5,8,13 based on:
          1. Volume
          2. Complexity
          3. Knowledge
          4. Uncertainty
       2. Calibrate this scale by defining a small story that will take about a half-day to develop and half day to test/validate as a “1”.
          1. Base every other score off of this, with common sense applying as a multiplier (i.e. a 5-point assignment is 5x more complex than a 1-point assignment.
    3. Establish Program Board
       1. Stories[rows] X Sprints[columns]
          1. Features fall into rectangles, as do any dependency markers.
          2. Dependents are tied together with a red string
  1. **Notes:**
     1. Increments (8-14 weeks) are 12 weeks long
     2. Sprints (1-4 weeks) are 2 weeks long
     3. Each MVP has one or more Features, of which have one or more Stories [Big Steps to Feature], of which have one or more tasks [Little Steps to Feature]
     4. The last sprint of each increment is used to plan for the next increment

1. Sprint Planning
   1. **Purpose:** 
      1. Develop the plan of the immediate work to be done over the next 2 weeks.
   2. **Goal:** 
      1. Specify features of desired MVPs and produce comprehensive list of stories, story scores, and story success (exit) criteria for each said feature.
   3. **Procedure:**
      1. Establish Real Capacity
         1. Capacity = (full time workers\*8 – each day off)
      2. Clarify Stories
         1. Story descriptions made at PI planning are likely jagged and made with little information. It’s important to clarify their exact specs.
      3. Break Stories into Tasks
         1. Each story is broken into a task, and each task is assigned an hour estimate.
         2. As a rule of thumb, but by no means a rule, a story point is roughly 8 hours of task work.
            1. However, tasks should be judged by time commitments, whereas stories should be scored by complexity. This presents a foil to the rule of thumb.
      4. Process Continues While There’s Capacity
         1. The velocity should be used as a rough gauge for how many points to take on this sprint.
         2. More importantly, however, the real capacity should be roughly the same as those presented in the story points.
      5. Build Sprint Goals
         1. Address risks and present possible mitigations (1-5)
         2. Take a team-wide confidence vote on finishing this sprint’s goals (1-5)
   4. **Notes:**
      1. *If the past sprint was a building sprint, this sprint must integrate and test altogether next sprint.*
      2. 2 hours at the beginning of the sprint is spent planning for the sprint.
      3. Every day, each team member gives their past 24 and next 24 accomplishments and work plans.
      4. ‘Velocity’ is the average number of points closed from each past sprint.
         1. If nothing changes to the work force, and if the team is planning correctly, the velocity should be constant.
         2. If the velocity is high, there is either many people on the team or a very good team of individuals.
2. Task Completion
   1. **Purpose:** 
      1. Eliminate actual engineering tasks in pursuit of producing final product.
   2. **Goal:** 
      1. Swarm over each defined task to complete as completely as possible and close out each task as fast as possible to bring about a product as quickly as possible.
   3. **Procedure**
      1. Move, task by task, duties from the (none) column, to (in progress), to finally complete.
         1. Once all of the tasks for a story are completed, the exit criteria is reviewed and if it allows, the story is closed out
      2. It is important to track how much time you spend, and have it consistent with the chart.
      3. As you complete duties daily, update the burn down line
         1. The ideal line is simply the number of hours on the first day over the course of the sprint, to equal 0 at the end.
   4. **Notes:**
      1. You cannot re-estimate stories, but you can add new ones
      2. The backlog contains everything, including those unnecessary but helpful.
         1. These can be picked up at any time on the side, or, if important enough, tasked in the next sprint
3. Sprint Wrap-Up
   1. **Purpose:**
      1. Analyze predicted performance of sprint, actual performance, and what can be dropped or gained to make the next sprint better.
   2. **Goal:**
      1. Eliminate nefarious sprint occurrences, gain efficiency-boosting tactics, and all-around ensure that sprints are on the best path for producing well-engineered products as soon as possible.
   3. **Procedure**
      1. Analyze burn-down line, anything that may have not gotten done in the last sprint/now needs to get done (i.e. failed test)
      2. Explicitly address:
         1. What went well
         2. What went poorly
         3. What will happen differently next time