3 Military Aviation

Lessons for Agile Projects

I've served as a military officer and Air Force pilot for the last 14 years. Over the past three years I've also worked with some of Seattle's smartest software engineers as an Agile project manager. These two worlds appear to be on opposite ends of the spectrum, but I believe they are far more similar than you may realize. Below are three lessons I've learned from military aviation which will benefit your Agile software project.

1. Slow Down

The phrase "Slow is smooth, smooth is fast" applies to military aviation and software development. Think about this training technique taught to all Air Force pilots. The first thing a pilot should do when they have an engine fire notification is…nothing…for 60 seconds! Why? Imagine a two-engine aircraft with one engine on fire and the wrong engine is shut down. You are now in a single engine aircraft, and your only working engine is on fire.

Situations *can* get worse and slowing down to ensure you are doing the right thing is better than doing the wrong thing fast. This applies to Agile projects where everything can feel like a crisis. The next time you are faced with impending doom, ask yourself how important immediate action is? Are you bypassing your normal safeguards for a production defect? The risk you are accepting might be worth it, but it should be a well thought out decision and not a quick reaction based on emotion.

Key Takeaway: Take a moment before doing something which turns a bad situation into a crisis.

2. Humans make Mistakes

Your Software developers are the best in the industry and are experts in their trade. I understand this mentality as a former Air Force pilot with over 3,000 hours of flight experience. Here is a shocking fact, every pilot I know, at some point in their career, has forgotten to put the landing gear down before landing. The point is that most pilots don't land without the gear because we created processes to ensure we catch those mistakes. We adhere to checklists which are run before each landing, we have a pilot and a co-pilot verbally confirm each step, and we have internal systems which will warn us if we get too low without the gear. These countermeasures have dramatically reduced the amount of human error in aviation.

Does your Agile project have any countermeasures in place for human error? If the answer is no, or you don't know, then acknowledge mistakes are happening and ensure every crisis results in a process to ensure it doesn't happen again. Taking the time to do a full root cause analysis will save you time and money in the long run. A great exercise to prevent future mistakes is a Failure Mode Effects Array (FMEA)which can give you a great starting point for countermeasures.

Key Takeaway: Humans will make mistakes and you should build processes to help reduce those errors.

3. Plans are useless…

One of my favorite quotes is from President Eisenhower on preparing for battle "Plans are useless, but *planning is indispensable."* I've been handed a 14-day flight mission, with 20 stops in austere locations, and given less than 24 hours to plan. I also knew without a doubt that the plan would change in the first 7 days. Does this mean we shouldn't plan…NO! You must go through the planning process because the more you plan the faster you can adapt to changes. The key distinction is how you plan. I use the below steps as a guide to help me structure planning and it can be applied to military aviation and software development projects.

- *Create a high-level overview to identify critical dependencies and choke points.*
- *Develop a plan for those critical dependencies and choke points.*
- *Ensure everyone on the team is aware of the big picture plan.*
- *Create a list of parking lot issues which need to be addressed but could be tackled later.*
- *Finally, create a detail plan the initial portion (first two legs of the mission / first sprint).*

The rest of the details we would constantly adjusting as we went. I've seen Agile projects struggle on both ends of this spectrum. They either plan too much and waste effort on details which will change, or they don't plan at all (because they are Agile) so nobody knows what is going on. A good rule of thumb is a 10-week high-level plan and a two-week detailed plan which is updated weekly. Regardless of how you set up your planning cadence, remember the process of planning is important, even if the plan doesn't survive first contact.

Key Takeaway: Continually assess your planning cadence to make sure you aren't too far on either end of the planning spectrum.