Making Rules

written by CD | OCTOBER 15, 2017

My first early childhood education exercise was to go to a pre-school where a teacher was having particular trouble with three very disruptive boys she could not get under control. She hoped just to get a day of relief by having me take them for the day. I approached the boys and told them that I hoped to be a teacher some day and that they could help me. They were surprised to hear that from me. I asked them to tell me what made a good teacher and a bad teacher. They said they didn't like their teacher because she had all these stupid rules. I acted very surprised and asked for an example. They said, like no playing ball inside the classroom. In shock, I asked why on earth would a teacher tell them that? Playing ball is fun!!! I probed further to get them to tell me why a teacher might have such a stupid rule. After some thought, they suggested the ball might break something. I said, oh wow, yeah, that might make some sense. One by one, I had them give me rationale as to why any teacher would have any of the rules that their teacher was imposing on them. They came up with some great reasons.

The next week I heard back from the teacher who wasn't sure if she should ask what I had said to the boys because they had somehow become the best behaved children in her class. Later I learned that one approach to establishing rules that students would adhere to is to have them come up with their own rules. It would take longer to arrive at a good set than it would if the teacher dictated them; however, the students observed would consistently adhere to the rules they came up with far more than those a teacher dictated.

Core Tenant of Agile Software Development Philosophy

When interviewing engineering manager candidates, I like to ask them: "If you had to choose just one tenant of Agile Software Development Methodology to be "religious" about, what would it be?" I get some interesting responses, but the one I tell them I am religious about is to not be religious about anything. Every situation is slightly

different and the most effective process is adapted to the situation. More on that in another post. My next favorite answer is to have Retrospectives where the team analyzes how the last sprint went and determines what to repeat, what to improve and what to not do again. In essence coming up with their own rules for how to best development software.