Ken Schwaber once flew on a plane next to some CEO. The CEO recognized him as “that Scrum guy” and asked him many questions about it. After Ken had finished, the CEO said something like, “So with this ‘Scrum’ thing, I’ll get the results I want in less than a month rather than 18 months!?” Ken replied, “No, you’ll get the results you *don’t want* in 30 days or less. Then, you adapt so you start getting the results you want in the next 30 days.”

There’s a disconnect in software development if you think you can get what you want without being disappointed. How do you know what you want? Maybe you want outcomes like, “Increase revenue by 15% by the end of Quarter 3” but are unsure of how to obtain this outcome. You might *guess* at how you could develop your product to yield more revenue, but it’s only a guess most times. Unless you test the assumptions informing your guess, your guess will live a long, dangerous life. Unless you examine the impact of your actions often, you run the risk of those actions being futile. Unless you learn from what you see, your success hinges on your market being knowable ahead of time. If this latter thought is true, you do not need Scrum. This talk is for all those who find significant aspects of their product and market change often and unpredictably. Prepare to be disappointed.

How much disappointment can you handle? How much is too much and how much is ok? When does disappointment absolutely wreck you and when does it strike you as curious, interesting, informative, even exciting? I’m thinking of Alexander Fleming’s penicillin discovery. He wasn’t searching for antibiotics, in fact he was conducting an unrelated experiment. I wonder what would have happened if he saw the weird growth on his petri dishes and yelled, “My project has failed! I’m terrible at this! I’m throwing this all away and starting again. Maybe, I can still make my deadline!”

It seems like a key to getting “the right amount of disappointment” is knowing how much and what types failure you can tolerate. Is it tolerable to fail to deliver a feature at the end of 18 months? 8 months? 8 weeks? 8 days? 8 hours?

Disappointment is related to failure of expectation, performance, outcomes, etc. If the extent and type of failure can be minimized, engineered, or otherwise chosen it falls with less heft. A blow to the head becomes a playful tap on the forehead. Ok, what might this look like in real life.

Product: Residential Mortgage Application  
Arbitrary Time Frame: 18 months

Assumption: People will want to initiate and complete a home mortgage application completely online and without the interaction of a loan officer

A crack team of software developers, product managers, stakeholders and an agile coach get to work. The agile coach is new and only focuses on helping the team get better at delivering items on their backlog rather than examine the impact of the changes being delivered. At the end of 18 months, the product is rolled out. Crickets. Hardly anyone is using it, yet overall mortgage applications have not slowed. **HUGE DISAPPOINTMENT!!!** No one is laughing and several folks are worried about their jobs. The top brass along with the product owner think maybe it’s a marketing problem or a user experience problem. Since they haven’t been showing their application to users until now, they don’t have good mechanisms in place to assess usability. They decide to hire a design agency to assess usability and survey their customers. In the meantime, they up their marketing spend. Tens of thousands of dollars are sunk into billboards, radio, youtube even park bench advertisements but application rates still remain low. Soo disappointing!!!

Three more months go by.

The design agency releases their results, and they’re VERY disappointing. There appear to be no major problems with the user experience. Customers who converted say their experience was quite pleasant and very easy from an application design perspective. Moreover, conversion rates are quite high! What could be going wrong?!

The embattled product owner chances to meets a colleague from a competing bank at a popular lunch spot. She bemoans her troubles in the nearly 2-year project. “I’m so disappointed, I could quit” she laments. Her colleague muses, “I wonder why this product isn’t working? Hmm, our application rates are through the roof.” Discouraged, the PO pays for lunch and walks back to her office. Maybe from inspiration or desperation, she stops a random pedestrian and asks, “Would you apply for a mortgage completely online? I mean, would you want to do the whole application yourself without having to talk to a loan officer?” Quizzically, the stranger replies, “Oh heavens no! I’d be too afraid of making a mistake that would last 30 years!” The PO is taken aback; the fundamental premise of their product initiative has just been roundly rejected. She thanks the stranger and continues walking. From somewhere in her disappointment, an ember of curiosity is glowing. She pulls her notebook from her purse and finds another random person to ask.

After a very long lunch break, she shows her findings to her Scrum team. She has found that, of the 50 people she’s talked to, only 5 of them would unreservedly use an application system like theirs. The rest say some kind of expert interaction is desirable. A developer interjects, “Well, what if we designed a feedback system where an applicant could tell us what they want to know about their application and what would make the application process better for them? We could do that in just a couple weeks.” The PO gives the greenlight.

In their next Sprint Review, the impact of the feedback form does not seem promising; application rates are still quite low. The team is a little disappointed, but one particular point of feedback peaks their curiosity. A few users said that they really enjoyed getting help via the feedback form and wished they could have a real-time conversation with an expert to answer their questions. The product owner wonders aloud, “Could we ensure they get feedback from our expert loan officers within 5 minutes of the applicants question?” An intrepid developer replies, “I think so. We’d have to scrap the email feedback process in favor of a chat system that notifies loan officers. Would that work for you?” The PO was a little disappointed that the email feedback form was going the way of the Dodo, but interested to know more about why some liked the loan officer feedback. “Make it so.”  
  
Fast forward another 3 months and 6 Sprint Reviews.

Application rates have tripled, average customer reviews have moved from 2.8 to 4.5, and even conversion rates have increased modestly. What changed? The team exposed themselves to customer feedback and disappointment. Before, they had guarded their assumptions from being tested and themselves from being disappointed. Their disappointments were gathering, like a river behind a damn. Unlike wine, assumptions do not age well.

“He who fails the most (without dying) wins.” – Seth Godin.

If you want to optimize learning, plan to fail half the time.

The second noble truth of Buddhism: Suffering is due to attachments and **expectations**, to grasping and clinging. Your inability to avoid change may make you angry, sad, and frustrated. It can be hard to let go of the false belief that the only way to achieve happiness again is to regain what's been lost.

Become familiar with disappointment. Become friends with disappointment. Carpe Demon.

Reducing the need for attachment may be done by reducing the fear of losing the thing we’re attached to. If disappointment is akin to suffering, then we reduce our suffering by making less fearful the thing we stand to lose. Smaller exposures to reality, tests with smaller scopes, less exposure to risk. This is what the Sprint (and really every Scrum event) represents – a moment to be transparent, inspect, be disappointed, and adapt. The key is that scope of each inspection is small. We haven’t had enough time to build up too much attachment to the things we’ve been doing. This makes the suffering of disappointment much less in the moment, but actually *more* over time. This is actually favorable since small disappointments can more easily be transformed into learning opportunities than catastrophic miscalculations.

# What

Disappointment is when our expectations do not match reality AND when the cost of being wrong is too severe for us to find any humor is the reasons why.

Insanity is doing the same thing over and over again and **expecting** different results.

# Who

Those who need and want the outcome.

Those who build the product development strategy.

Those who build the product to satisfy those who need and want it.

Those who guide, counsel, teach, coach, lead, and share the burden of making it real.

You will always be disappointed when you **expect** others to act as you would.

# Where

The intersection of ourselves (or the things we create) and *gemba*, the *real place*, production, conversation, performance review, an Erlenmeyer flask, the stage when curtain rises, when you pull your rip cord, taste the sauce, check the soufflé, tell the joke, choose to be vulnerable, pitch the idea, show the new functionality…

If you **expect** the map to describe the journey you should also **expect** disappointment. Scrum has no map.

# When

Disappointment occurs when expectations meet reality and a breakdown is the result. More specifically, when we *inspect* the interaction of our expectations upon reality and find the results are not what we want.

When has anything happened when you **expect** it to?

# Why

The cost of being wrong outweighs or is perceived to outweigh the value of learning to be had.

Inspect and Adapt is the Law of the Land. Until **expectations** are shed, one can’t inspect with clear eyes, nor adapt to the unexpected.

For me, it’s *attachment to expectations* that needs to be shed. Expectations are fine e.g. “I expect that adding this feature will produce this effect.” The expectation is worked up into a hypothesis and a test (working software deployed to production) is conducted. This is the primary measure of progress. So long as the attachment to expected results is tempered, the expectation itself becomes a guide. It’s intuition guided by knowledge and fueled by curiosity. This leads the Scrum team into unknown territory. Empirical knowledge becomes the map you follow, Scrum is the vehicle you take, and the Scrum team’s behavior (if aligned with the Scrum values) is the fuel that pushes the vehicle forward.