# Scenario Background

* Five weeks ago, the team started focusing on their currently-highest impact technical debt: bugs that take a long time to diagnose.
* Team decided that the key measure is 80th percentile number of days to diagnose per bug (counting work days only). Success is to get that below 2 days. It started at 11 days.
* The team has chosen to categorize bugs by what makes them take so long.
* The current category is bugs that cross between code from several teams. It is hard to get all the expertise needed from multiple teams. Bugs bounce back and forth, or someone spends time figuring out code written by another team.
* The last 3 weeks, the team tried a number of experiments to improve cross-team tracking, prioritization, and communication.
* So far, there has been no change in the target measure for this class of bugs, which sits at 13 days.
* Team may or may not know why.

# Scenario Goal

* Team is sharing results of this line of inquiry.
* Leader wants to improve team’s ability to own tech debt.

# Leader

## Beliefs

* Their key measure is a trailing indicator. It is the right outcome measure, but they probably need an additional, predictive measure.
* The problem is clearly bugs that bounce between teams and then sit there with no action for days.

## Remember Your Stance!

* Ownership is more important than the team finding the right solution immediately.
* The team has more info than you, always.
* Help the team realize what it knows and what are unknowns.
* Learning happens when there are unexpected outcomes. You don’t want everything to happen as planned or hoped!
* Mentoring is OK. If the team is stuck, co-create at least 3 possible solutions.
* Challenging is OK. If the team has a blind spot, point it out and ask them how they will correct it or correct for it.
* Consulting is NOT OK. Do not make decisions for them or provide direction.
* Judging is NOT OK. Let the team experience the natural consequences of actions, not your praise / shaming.

# Team

## Beliefs

* You’ve tried improving tracking and communication all that you can. It’s just not fixing it.
* Diagnosing these bugs is hard. It requires deep knowledge, because there often isn’t a lot of information to go by, the information is arcane, and only a few people understand the code well.
* Key individuals are often occupied, often on other bugs. They lose time thrashing between tasks, and bugs block on them.
* This is particularly bad when you need a key person from another team.
* People are all trying to help the greater good; people are happy to help another team.
* You may or may not be sure of what to try next.

## Remember Your Stance

* You own this. You probably know more than the leader.
* Don’t try to convince; just try to show what is true.
* If you need a resource, ask for it. But labor is not a resource – your labor is your own and you don’t have to ask for it.