I Break Stuff for a Living

You know the best part about building a bridge? Finding out what makes it collapse.

This is the kind of thing I used to think about all the time when I was 10. At the end of every school day, the bus would drop me off about half-a-mile from my dad's house. To get to our neighborhood, I would walk across a wooden footbridge that was built over a dried-up riverbed.

That rickety thing must have been older than anyone on my street. It was so old, in fact, that water hadn't run under it for years. Kids would play in the riverbed, kicking soccer balls and chasing dogs into the brush. I'd sit on a rock and wonder what it would take to make the bridge fall down.

Eventually, curiosity and a hot summer day got the best of me. I won't go into too much detail. Let's just say it involved a few bicycles, some rope, and a lot of buckets. The affair ended with my dad telling me to go to my room. I was grounded for three weeks.

As soon as I was allowed to leave the house, he walked me down the street with a bundle of lumber. My dad was an architect, and that day he helped me figure out what made the bridge finally crack. Then, we fixed it.

In middle school, I kept looking for other bridges to break. Every time a math teacher began reading from the textbook and drawing diagrams on the board, I'd slip into a daydream. It was impossible to focus in the classroom. But during lunch hour, I'd read about airplanes and space shuttles, flying hundreds of miles an hour through sky and space. How did anyone come up with this stuff? How could anyone be sure that they were safe?

I imagined suiting up crash test dummies for a supersonic test flight. I wondered: Could NASA scientists be grounded, too?

One day in high school, I noticed my trigonometry teacher working at a notebook computer. His screen had two windows open. Both had black backgrounds and were filled with line after line of intense-looking words.

"It's a computer program," he explained. "I just figured out why it's broken."

"Who broke it?" I asked, without thinking.

"Well, I broke it," he responded. Looking at my perplexed expression, he added, "I mean, if you think about it, anything you're building from scratch is broken until it works, right?"

After that conversation, I started staying after school to help my teacher break his program. It was supposed to read 300 homework assignments that our class had completed on the computer, grade them, and then show him the lowest grades. If he could get this thing to work, he could spend less time filling out grades and more time helping the students who weren't doing as well.

The problem was that his program couldn't understand a lot of the answers it was reading. It had to do with the way some students chose to type out fractions and math symbols. Different students typed out their answers in different ways, but the program only spoke one kind of math, I suppose.

My teacher had made a bunch of fixes to the program, and now he was thinking of other ways that students could surprise the computer. Every time he broke the program, he could figure out a way to teach it a new trick.

I learned that breaking computer programs was the only way to figure out whether or not they would work for every possible condition. Testing my teacher's programs was a lot like dragging buckets filled with sand onto an old, worn-out bridge.

The question, though, was who would be weird enough to act like my 10 year-old self when doing a high school math homework assignment. After a couple of weeks, we realized we were testing for things that would never happen, and stopped finding ways to break the program.

I started learning to code, taking classes online to become a software engineer. The next year, I found the perfect job, doing "Quality Assurance" work for a tech company downtown. I've been doing the same job, in different ways, ever since.

Working on the Q.A. team is kind of like waking up every day and finding new ways to break stuff. I talk to the engineers to see what they're trying to build. This week, it's an interactive web page that lets students see different pieces of a movie by jumping to different parts of the world on a map online.

Once they've got a prototype up and running, I create a fake person—a "test user"—on the computer. Instead of trying to break the map a hundred different ways myself, I turn my test user into its own program. The test user can try those hundred different things in just a few seconds, showing us what's broken, and helping the team decide what to fix. Whenever we update the program, everything must be tested by Q.A. to be sure the new version won't break.

Every once in a while, I write a test and find a bug that would be really difficult to fix. Sometimes, I break the system in a way that's so clever there's no point in making a fix. The team will tell me that no sane person would go to that much effort to break the system, so the bug will probably never cause us trouble.

I keep waiting for them to tell me to go to my room.