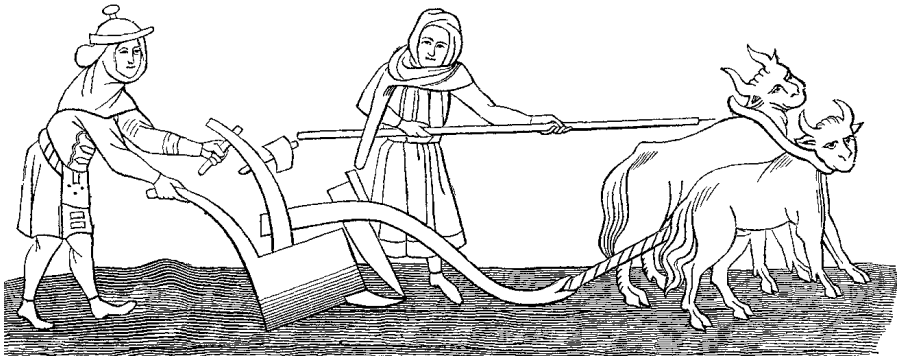
— 4 —

How to Reach Perfection

"Improvement usually means doing something that we have never done before." —Shigeo Shingo

In ancient times, in the days before the horse harness was invented, oxen were especially useful animals because you could hook a cart to their horns. In some countries, even today, you can see an ox pulling a cart down the road, tied by the horns.

Farm workers used a long stick, a prick, to poke and control the oxen. It didn't feel good, and sometimes the ox would rebel by kicking the prick, which did nothing but hurt more.



Whatever goal the ox may have had, kicking against the pricks did not help to achieve it. From this comes the old saying, "to kick against the pricks," meaning "to do something utterly useless and painful."

Einstein said something similar with different words. He said, "The definition of insanity is to keep doing the same thing but expecting a different result." If something isn't working, try something different.

The same thing is true in chess. FICS, the free internet chess server, keeps statistics of people's ratings over time. One person played 60,000 chess games and ended up a worse player, with a lower rating. It would seem logical that to get better at chess, you should play many games, but that isn't sufficient.

To improve at chess, Yasser Seirawan suggests you should read books on chess, use arithmetic to count pieces on the board, take responsibility for blunders, and use reasoning. Grandmaster Lev Alburt suggests practicing a lot of tactics. Jeremy Silman suggests studying positional chess. If continual playing isn't working, try

different things until you improve. Of course, some people don't want to improve, and are happy enough just playing. There's nothing wrong with enjoying the game of chess and not trying to improve.

Now it's time to look at this principle from the point of view of programming. If you want to improve, you're going to need to change something. Maybe try what Joel Spolsky says, and have a one-click build. Or listen to Uncle Bob Martin, who says that each function should be clear and tell a story. Experiment with the advice of the Pragmatic Programmer, and leave no "broken windows" in your code. Maybe try out pair programming.

This is the golden rule of programming, the straight brick road to perfection: whenever a mistake happens, ask yourself how to avoid that mistake in the future. Always ask yourself this.

Zero Quality Control: programmers are not the only ones interested in improving the quality of their product. ZQC is a Japanese manufacturing system for zero-bug quality control. ZQC realizes that people make mistakes, and catches them at the moment they are being made. When a mistake is found, it doesn't mean people are stupid. It means they need to find a way to avoid the mistake in the future.

In programming, you can do the same thing. After you write a line of code with high potential for error, double-check it with your eyes to make sure you didn't make a mistake. If you need to manage resources, write the destructor before you write the constructor to make sure the destructor isn't forgotten! Write code in small enough chunks that they can be tested easily and visually inspected for correctness. Make your code easy to read.

When you make a mistake, don't despair. Remember that some people delete entire customer databases by accident. Don't delete a customer database, but if you do, focus on improvement. Franz Liszt once told a young pianist, "Have patience with yourself, your future is ahead of you. Rome was not built in a day." Don't let the negative voice in your head tear you down.

If you stop looking for ways to improve, you will stop improving. If you know a bug is there, fix it. Don't leave it for later, when it will be harder to fix. Be the Jean Valjean of fixing bugs.