

In Praise of Bad Programmers:

A tale showing the difference between individual and team skills

Bad Harold

Harold was a bad programmer—I mean a **really** bad programmer—the kind who owed it to himself and all around him to find another profession. But Harold was a nice guy and he was a lifer; he'd been with the company for a long, long time. He had been a low-level programmer forever, he never got promoted, he got minimum pay raises each year, and he got moved around a lot. But nobody wanted to fire him. So every time a project started up and needed headcount the manager who had Harold on his team at the time took the opportunity to unload him onto the next person unfortunate enough to have to supervise Harold. One time that person was me.

The Keen Team

The scene was in the 1980s when longevity with a company meant more than it does today. The team I managed was very keen. Probably the defining characteristic of this team was that we, all of us, wanted very much to become a better team and we worked very hard at being one. We had well-defined and practiced processes, we developed and implemented requirements and design modeling approaches,

we applied and refined a comprehensive inspections methodology that inspected requirements, designs, code, test plans, test cases, **everything**. We even invented an innovative new testing method. And when we became more effective, we worked hard at becoming even more effective. Then Harold joined the team.

Discipline and Success

Long before Harold came on board, the team had made a pact: we would allow no bad work, not from anyone, not the chief designer, not the test coordinator, not the manager. Not even Harold. We agreed we would provide all the processes, templates, support, resources, training, reviews, feedback, and discipline necessary for every team member to be 100% successful. We even defined the metrics that measured this success. So when Harold joined us, we laid this system out before him and explained that following these processes was a condition of being a member of the team. This discipline applied to everyone, even Harold.

A Program gets Redesigned

As his first task, Harold was given one of the easier programs to write. The package he received prior to coding

contained good reviewed requirements, an excellent program design complete with graphics, a proven process for developing test conditions, and a schedule for the ultimate inspection of his code. While he was coding, other project issues arose and we couldn't keep to the review schedule. When Harold was finally able to bring his code for inspection he had already unit tested it. This was in conflict with the normal and expected process order, but it wasn't Harold's fault; and the program worked. Well, at least according to Harold it did. However, in the code review we saw right away that the program he wrote did not follow the reviewed and approved design at all. Harold had not submitted (and defended) an alternative program design to the one created by the team's systems designers; he had just written what he wanted to write, the way he wanted to write it. Accordingly, his code failed the inspection immediately. Harold was shocked: "*...but it passed all the tests!...*" he said. In fact, he complained loudly and to anyone who would listen. His complaints reached all the way up to the Director of the installation *protesting*: "*...they're making me rewrite something that already works!...*" Fortunately, his griping was

ignored, the Director backed the team, and Harold had to rewrite the program to the original design. When the code finally came back for inspection, it matched the design, worked well, and passed inspection easily with relatively few comments and issues.

A Better Team, a Better System

As a team, we noticed an interesting thing: **because** Harold was a low-achieving programmer, we had become a better team: we had defined and enforced our process more tightly, we had set up and captured metrics more consistently, we had learned to provide better counseling and support to the programmers (especially Harold). Most importantly, we had adopted a team goal and ethic: we would (a) never accept bad work and (b) never simply discard a weaker member of the team who could not perform—we would do whatever was necessary to allow that person to produce good work and to meet the team's standards. Provided team members supported these goals, stepped up to the challenge, availed themselves of the team's resources, and produced viable products, they could be a member of our team and the team would take care of them professionally.

This higher ethic made us a better team.

The Physics of Systems

Modern physics is learning that the behavior of systems may not be a direct or predictable function of the behavior of the parts of the system. Not all cogs in a watch have to be big cogs. Sometimes more optimal behavior in a system can be obtained from less optimal parts. Sometimes the best teams are not made up of all superheroes and when a team asserts itself to overcome some limitation, even with its own personnel, it can become better than it would have been without that limitation.

This is not to say that we should intentionally seek to employ bad programmers. But teams are not disconnected parts like nuts and bolts in a bucket; they are systems. The dynamics of an effective team are more complex and subtle than they sometimes appear and low-performing people may have an effect on a project other than through the products they produceⁱ.

Individual vs Inept; Creative vs Crummy

Sometimes it seems that software process is implemented with the intention of squeezing out any trace of individuality and creativity—to make building systems a **mindless** process. In this team, we were careful not to do this; but we were also careful not to simply allow anyone to change anything, anytime, without consideration of the value and

return. We certainly never allowed anyone to make changes simply out of laziness, misunderstanding, or incompetence and we never let anyone fail.

Harold Redeemed

When Harold had his code approved, he (to the surprise of everyone on the team) remarked “..you know, *this design is much better than the one I did...*” In fact, from that point onward, Harold became a believer in our approach and (to the surprise of everyone in the installation) a vocal evangelist for our team and our processes. He even made a significant contribution to the team's processes and tools—with approval of course.

The team had defined the highest level of inspection acceptance as an “AS-IS”, meaning no errors of any sort, no minor functional issues, no design questions, no documentation inaccuracies, not even spelling or grammar mistakes. AS-IS was **perfection!**

In nearly 500 inspections we conducted during the project lifecycle, only one person on the team ever got an AS-IS on any work product ever submitted for inspection.

That was Harold.

ⁱ See Harvey, Jerry B., **How Come Every Time I Get Stabbed in the Back, My Fingerprints are on the Knife? And Other Meditations on Management** Jossey-Bass 1999