

Seven Mistakes Made All the Time by Most Software Development Organizations

by Randy Charles Morin

In the tens plus years that make up my career, I've worked for nearly twenty companies. That's likely not something to brag about, but it is the truth. This experience has given me a unique perspective. I know what is normal and what is not normal in the software development industry. I also have a pretty good idea of what works and what doesn't.

Beyond anything else, what I have gained in this experience is a lot of funny stories of what development organizations do, that they don't realize are blatant mistakes. What you have to remember is that development managers are often good developers, rather than good managers. This leads to some very poor managerial decisions, or rather mistakes.

I worked for a company named Bell Sygma (now CGI). Like every company, they required that I start working for them ASAP. They needed me on to start Monday. I've heard that 20 times. I showed up on Monday and there was my new desk. Fortunately, I brought my laptop, because my desktop computer had not arrived yet.

Mistake #1 – Software developer arrives before his computer.

This is not the only time that I've joined a company and they didn't have a computer for me. But quite often the computer arrives in the first week. On a couple of times, it took as much as a month and more. This was going to be one of those month and more scenarios.

I met my co-workers. That was fun. One of the co-workers became one of my better friends. We spent a lot of time at the local J.J. Muggs sharing beer, especially at happy hour.

One of the first days at work, he told me that he bullshitted his way through the interview. Our boss had asked him what MFC was. He said that it was Microsoft Foundation Class. He was hired. But he actually didn't know how to program with MFC. He only knew what the acronym stood for.

Mistake #2 – Software developer is not properly qualified.

Funny thing is that everybody makes this mistake. In my few years in this industry, I have never once had someone check the references I list on my resume. And I have had my fair share of jobs and interviews.

This project called for 40 developers. Forty were hired over the first months. It turns out that several of the developers were lacking real qualifications. But as each developer joined our crew, they also joined a crew of developers with no computers, no network and no software. For more than a month, I remained the only developer with a computer, my personal laptop.

Neither was there a specification. Nobody really had much of an idea of what we were going to create. We were located in Toronto, Canada. The only manager of the handful of managers working the project who had any idea what we were about to create was located 500km away in Montreal, Canada. He didn't write much down on paper, rather on the occasion that he was in town, he would go around to all the developers with his laptop showing them the prototype that was developed in OS/2.

Mistake #3 – Lack of a specification to guide the developers.

This is a standard mistake. Except the rare development organization, the specification is never written. On many occasions, I have even been handed the specification after 90% of the development was complete. The specification was a bunch of screen shots of the nearly complete application and tying them together in a document with a bit of textual explanation.

Finally after a few months, the computers arrived and everybody spent the week, configuring their computer, installing their software and people were generally happy. Then only a week or two after the computers were ready, the managers moved the entire staff into a new building. So we spent a month plus without computer, then we got the computers and only a couple weeks later we were moving to a new building.

The I.T. staff began configuring the computers to run on the new network in the new building. This was somewhat unnecessary as the developers were capable of doing this on their own.

At first, it wasn't really that apparent, but eventually the developers notice that the network stability was proportional to the amount of time the I.T. staff spent working on our computers. That is, the more time the I.T. staff worked, the less stable the network.

Mistake #4 – Network problems.

On one occasion, the color printer stopped working. Fortunately, the office clerk didn't notice. The developers realized that if they called the I.T. staff to fix the printer, that the result would be disastrous. So, the developers hid the color printer, hoping that nobody would say anything if they couldn't find the printer.

After a couple weeks, the office clerk inquired as where the color printer had gone. The developers revealed the hidden printer. The clerk called the I.T. staff. All the printers on the entire floor stopped working shortly after the I.T. staff arrived. A few hours later, the entire network was down. That was very frustrating.

Another problem was also brewing. The managers had purchased 10 diskettes for the entire development staff. The diskette hungry developers quickly consumed them. We needed more. Although management was prodded for weeks on end, a new supply of diskettes didn't arrive for several months. The developers had to purchase their own diskettes in order to save their data onto a removing media.

Mistake #5 – Not supplying the developers with the tools they require.

I've seen this on countless occasions, at nearly every development organization. Diskettes are the most ridiculous of examples, as they are cheap and very available. But failure to supply developers with the tools they require is very common. Tools include servers, software, phones, computers, desks and more. Some of the 40 developers at Bell Sygma did not get phones until 6 months after they started working there and a telecommunication company owned Bell Sygma.

The best way to keep a development organization happy is to supply them with the tools they need to get their jobs done. Developers thrive on their work. As long as they are not distracted with problems, like a lack of diskettes, then the developers should be able to keep themselves motivated.

With 40 developers on staff, it was also apparent that we were going to require some team software to allow the developers to work together. The management team had ordered an expense source control product, but the I.T. were having a difficult time installing the product. The management team decided that they would request a demo copy of Source Safe to see if it were any more difficult to install.

Source Safe was easy to install and configure and within a few hours, we had a small portion of the team using the product very successfully. The team continued to use the product, even though it was only a demo version and not the real thing. After the 30-day demo period had expired, the developers continued using the product.

Mistake #6 – Software piracy.

The management team knew that continuing to use the product beyond the 30-day evaluation period was piracy. They didn't care and continued to use the product without purchasing it. They also knew that the development team was also aware of this piracy. What this communicates to the development organization is that piracy is acceptable. That ethics should not stand in the path of progress.

Things degraded from there. The development team began experiencing ethic problems. Backstabbing each other, lying to management, pirating code and much more became the prevailing paths to success in the organization. Management had to discipline some developers that stepped well beyond the boundaries of what is considered ethical. Management had no one to blame but themselves as they started the team down this unethical path.

Software piracy runs rotten in most development organizations. I don't think I can remember any organization that didn't pirate any software. I don't believe that piracy is acceptable, but I have to live with it to keep a job.

On the other hand, you cannot be too rigid about following rules. Some rules of the land are to prevent unethical behavior. But others are simply unreasonable. For instance, Bell Sygma at the time had a coding standard that dictated that all filenames were to follow a corporate standard. The standard suggest that all filenames be exactly 8 characters in length followed by a file extension. The first four characters were reserved for the

organization and product code. The next two for the project and the last two could be used liberally.

Of the forty developers, three of us worked on the client project. That meant that all of our filenames had to start with a particular 6 character prefix and that we only had two characters to uniquely identify our files. This standard would have made it difficult to develop our MFC client.

Mistake #7 – Unreasonable standards.

It's important to create organizational standards. And it's important that people follow the standards so that integration of components is made easy. But organizational standards and processes that yield little to zero benefit should be removed. These zero benefit standards are a barrier to success. A good question to ask is "Why does this standard or process exist?" and "What is the benefit of this standard?" Quite often and in the case mentioned, the answer was that this standard worked well in the past, but that there was no known benefit of continuing to follow the standard.

It has been suggested that it is sometimes difficult to tell what is an unreasonable standard that should not be followed and what is a reasonable standard that if not followed portrays unethical behavior. I completely disagree. But I won't convince everyone of this and I won't try.

Eventually after 6 months of frustration at Bell Sygma, I moved onto a new position with a new company. A few months later, the project was scaled back because there wasn't any visible progress. Eventually, the project was dropped completely. Around the same time, the company was purchased by CGI.

I figure that salaries alone were at least \$2 million for those first 6 months. Overall the project must have cost several million more and with no benefit. Everybody makes mistakes. All mistakes have consequences. Some mistakes have consequences more terrible than those outlined in this article and some mistakes have consequences less terrible. But hopefully you can avoid the seven I outlined.

About the Author

Randy Charles Morin is the Chief Architect of SportMarkets Development from Toronto, Canada and lives with his wife, Bernadette and two kids, Adelaine and Brayden in Brampton, Canada. He is the author of the KBCafe.com website [<http://www.kbcafe.com>], many programming books and many articles.