10 ways to avoid mistakes during project development

Version 1.0
February 22, 2010

By Alan Norton

You may have heard the phrase "Be proactive, not reactive." It's certainly appropriate when discussing how to best deal with mistakes. The idea, of course, is to prevent mistakes before they occur. But how exactly do you do that?" In this article, I will detail 10 real-world ways you can preempt mistakes during project development.

For the sake of simplicity, I've lumped all errors into one category: *mistakes*. The items listed here are specific to developers, but other IT roles can also benefit from many of them.

1 Learn from other's mistakes

Find experienced peers who are willing to share their mistakes and then learn from them. I am somewhat biased, but the editors, writers, and members of TechRepublic seem willing to honestly share their mistakes -- even if somewhat embarrassing at times.

2 Do your research first

No matter how much you know, you'll encounter new challenges on an almost daily basis. Each challenge usually requires you to learn something new. Before you tackle a problem or task, do your homework. The trial-and-error method of learning may have been necessary and acceptable years ago. But with the resources available on the Internet today, there is little excuse for mistakes made because you didn't do the proper research in advance.

3 Have a plan

You can't know how to get to your destination without a roadmap. In project development, that roadmap is known as a project plan. Whether done formally or informally, you need to know how to get where you are going. Days or even weeks of programming time can be lost if the wrong path is taken. When done the right way, a project plan will help keep you from straying off course.

4 Follow standards and use templates

There are good reasons why experienced professionals took the time to create and publish industry and company standards. Standards detail best practices and procedures learned over years of trial and error.
Templates such as predefined forms can be useful since most of the work is already done in a standard format. A standard EULA approved by your legal counsel is another good example of a template that can come in handy if you are developing application software. A mistake in a legal document can be an expensive exercise -- one best avoided.

5 Communicate and coordinate with others

If you are part of a team, it's essential to communicate with other team members to avoid redundancies and to coordinate your work with theirs. Emails, instant messages, project status reports, and teleconferences are all ways to communicate and coordinate with others on the team. Unfortunately, each of these is far from perfect. You can spend the better part of a day reading and writing emails, participating in conference calls, and instant messaging with your peers. But it is a necessary part of the development process.

The perfect tool for communicating and coordinating with others in a team environment has yet to be developed. One of the better tools developed to share code is revision control software. Your project may also benefit from the use of a communication plan that ensures everyone involved -- including customers and stakeholders -- is kept apprised of key developments. (The TechRepublic downloads library has a free communication plan template.)

6 Allow enough time

It was at Hughes Aircraft Company that I first heard the phrase "You want it bad - you got it bad." It didn't happen very often, but when it did it was almost always made in reference to a part from a vendor that was badly needed, rushed through production, and upon arrival, failed testing. Failure to allow enough time for each phase of the project can lead to missed requirements, inadequate analysis, poor design, rushed programming, insufficient testing, and incomplete documentation. The result can be a system that doesn't meet expectations and fails in one or more key areas.

Estimating the time needed to accomplish each phase of a project is difficult. I achieved the best results when I sat down with my supervisor and determined the time allotted for each major task in the project plan. I was overly optimistic in my estimates. He was much more realistic in his estimates, and he turned out to be right. As a rule of thumb, doubling my initial estimates came close to the actual time required. That information was useful for developing project plan timelines.

You may need to develop a similar rule of thumb until you can more accurately estimate completion dates. Ideally, you want to complete each phase of the project on time, and the best way to do that is estimate them correctly up front. Here are a few tips on creating realistic schedules.

7 Reuse proven code

If you're an experienced developer, you should have built up a large code base over the years. Go blue and recycle this code whenever possible. You will likely have to modify the code to fit the new requirements, but proven core code is a good foundation to build on. Not only will you reduce the risk of introducing new bugs, but you will eliminate the time wasted creating similar code and the subsequent testing required.
Share your code with others so they can reuse parts of it. Proven code can be shared via plug-ins or libraries. Good external sources of code are available on the Internet that can be legally used for free or for a small fee.

8 Use checklists

Before a commercial plane trip, the pilot and co-pilot are busy walking through a long, detailed checklist. Checklists can be used during various phases of the project development process. They are particularly useful when working with large systems and when a single person is responsible for multiple tasks.

For example, a list detailing the steps required for system turn-on will help avoid accomplishing tasks out of order and prevent errors of omission. It is all too easy for developers to overlook important items like system access when they are busy doing final testing and documentation.

For more on the virtues of using checklists, see Leverage checklists to improve efficiency and client satisfaction.

9 Test, test, test... and carefully review your work

There is a healthy level of paranoia about delivering error-free work. Test as much as possible as early as possible. Errors in the code are typically more expensive to correct when found near the end of the development process. The last thing you need when facing a critical release date is to find a bug that should have been found months ago.

Careful and thorough testing will allow you to find those mistakes before your users can. Double- and triple-check your work. Develop test data and a plan to test common calendar-based events like EOM processing and annual reporting. All functionality and every single possible scenario should be thoroughly tested. And, yes, this is also a good place to use a checklist.

10 Test again with a third party

Find at least one experienced person who can be dedicated to the beta testing. They will undoubtedly use the system in ways you never dreamed of and find bugs you missed.

Don't overlook or rush this final quality assurance task. It's typically your last chance to get it right. Once a bad piece of software is released or a system with a critical bug is turned on, a company's image can be tarnished for years to come.

The final word

One of the most important lessons I learned very early in my career is that a mistake isn't a mistake until someone else knows about it. I was but a young inexperienced pup when I accidentally deleted some system files on a Tandem PC. It could have been a disaster. But I had enough sense and problem-solving skills to identify and copy over the missing files from another Tandem PC.

I have never told anyone until now about my near disaster. This may be obvious, but you should keep unseen mistakes to yourself. There is almost never anything to be gained by telling others you have done something *really stupid.* It can negatively affect your image and possibly damage your career.

I hope these proactive tips will help you avoid making an embarrassing mistake that becomes known to your boss, peers, and users. If you find yourself in that unenviable situation, you might want to read Calvin Sun's article 10 things you should do if you make a big mistake.

I have always learned the most from my mistakes -- but I prefer not making them in the first place.

Additional resources

- Mini-glossary: Project management terms you should know
- Build a foundation for project success with this definition template
- Project planning template for project managers
- Top project management resources: Best of Tom Mochal
- 10 things you should do to successfully manage your workplan
- 10 techniques for gathering requirements
- 10 tips for meeting IT project deadlines
- 10 things you should do near the end of a project