Preventing most-probable errors in planning

Robert M. Poston,
Software Standards Editor

Promotion, schedule, cost, product quality — all are vital concerns of professionals on a software project. This seven-part series on error-prevention will move through the software life cycle, showing how to prevent the most-probable errors in each phase. And, as this series progresses, we'll see how good old standards can help every step of the way.

Identifying most-probable errors

You hear a lot about fault-removal programs these days. These are commercial versions of the long-established military verification and validation programs. In both fault-removal and verification and validation, the aim is early error detection.

That is not good enough any more because we now know how to anticipate and block errors in all phases of the software life cycle. Unlike early error detection, prevention focuses on not letting errors happen in the first place. However, we do not know how to prevent all errors in all phases, so fault-removal techniques are still needed.

What is the key to software error-prevention? Tell professionals what a good job is. Most of the time if they simply know what a good job is, they'll do a good job and avoid many mistakes they would otherwise make in floundering for direction.

Because our industry has experienced so much rapid technological change, we have not had breathing time to establish what "good" is. Seldom could managers tell people what a good software development plan or requirements/design specification is, let alone what good programs or tests are.

Today, we do have industry-level, consensus-based IEEE software standards that establish what "good" is in many areas. These standards provide a starting point. With this baseline, we can measure deviations from or errors relative to it. Once we can measure deviations from something, we can improve it. If we can systematically improve the products of the software development cycle by stopping common errors, we can systematically improve the process of developing software.

Vilfredo Pareto (1848-1923), an Italian economist and sociologist, showed that about 80 percent of discovered problems in manufacturing could be attributed to about 20 percent of possible causes. Programmers today may not recognize the term "Pareto analysis," but most will tell you that the 80/20 rule does hold for software.

Preventing errors in project planning

If we could classify errors by types or categories and collect this problem data for two or more projects, we could predict the probable statistical distribution of errors on the next project. We would know which errors occurred most often according to Pareto analysis. This is seldom done, however, because it costs too much and takes too long. Besides, who wants to dwell on their errors in detail?

But, just suppose we did get all our most-probable errors neatly classified according to Pareto analysis over the term of two projects. Errors result from incorrect actions (sometimes inactions), and they can be observed usually only for a moment if you are paying careful attention. Incorrect or omitted actions usually go by unnoticed or are forgotten until they show up as faults in the final work product. Classifying them has not proved very useful.

At this point, the most-probable errors idea may sound a little shaky, but there are some underlying principles that can be extremely helpful when applied as follows:

Count as errors only those that are manifested as frequent faults. Make a list of those errors for each work product you are producing. All should be observable defects. Keep the list hierarchically organized and short: no more than seven errors. Provide the list and a standard to each project team member before work begins on a product.

Every member of the project team now has a description of what is to be produced (the standard) and a list of errors that are most likely to crop up when doing this job. It is heartening to see how good a job people will do when they know what they are supposed to produce and when they know what pitfalls to look out for while producing it.

This a far cry from the more usual directives managers issue. "Go produce X. You figure out what it is supposed to look like." Often implied, sometimes
stated: "An engineer should have no trouble with that."

Starting the most-probable errors list.
To be statistically correct, we should derive our list from a Pareto analysis. But most managers can't wait for that. They need help on this project.

"Given no statistical history, how can I come up with the most-probable errors for my list?" the project manager asks. For this project you'll have to compile your own most-probable errors list, use it as a benchmark on this project, and modify it as you collect statistics from other projects.

If we look at error information from testing and defect-removal activities, we see three categories that always appear at the highest level of classification: missing, wrong, and extra code. These classifications can be used effectively to separate types of errors that occur in documents, too. We'll partition our most-probable errors list for project-management plans into missing, wrong, and extra information categories (see Figure 1).

In each category, we'll limit the list to no more than seven errors. It is easy to remember seven or fewer items, and if we keep the list pared to the most-critical errors, we won't even have to refer to it after a while.

 Missing information. Fill in the missing-information category first. Gather two or three project-management plans from managers in your organization. These plans represent as-is project-planning information. IEEE Std 1058-1987, a standard for software project-management plans, provides a list of information that should appear in a project-management plan. Does your as-is information agree with the should-be information in the standard?

 Beware! There may be solid reasons why certain information has been omitted from the plans you are examining. Has the legal department restricted information content? Are politics at work? Is part of the project being funded from another source?

 Wrong information. Now work on the list of wrong information. In the wrong category is information that is present but incorrect.

 Standards won't help here. This is where the school of hard knocks provides direction. In most organizations, though, a couple of helpful reports may circulate around the office. Usually they are titled "Lessons Learned" or "Post Mortem." They describe what went wrong and what went right on previous projects. Look for these reports. Sometimes they are tedious to read, but they can be valuable.

Extra information. Errors of extra information should be listed next. Authors of plans often include extra information thinking it will add clarity to descriptions or justification for actions specified in the plan. Readers normally find this extra information confusing. "This is just a lot of noise!" they may complain.

 However, information that is extra in a project-management plan may be needed in other documents on the project. This means we need standards to tell us where information belongs. The IEEE family of software engineering standards tells us where to put information and where to find it.

 If you know, by being familiar with these standards, that a piece of information in the project-management plan really belongs in the requirements specification, you have identified extra information in the project-management plan.

 A word of caution here: Occasionally, a project-management plan is intended to be more than just a plan. For example, it might be directed at educating a customer about the product to be built. When this happens, extra information may be justifiable. This special-purpose information should not be listed as an error.

 Go public with the list. Now that you've got your list, don't keep it to yourself. Walk it around the building. Let the quality-assurance people and project managers comment on it. But remember the rules: No more than seven errors in a category, and each error must be observable as a defect in the product.

 Preventing the errors. For most software professionals, error-prevention is a new idea. Ease into it. Try it. Watch it work for you.

 Initially, simply provide your most-probable errors list to each project manager. Any reasonable person who is warned of potential problems will try to avoid them. Errors should decrease right away.

 Ready to go further with error-prevention? Let's take the sample list in Figure 1 and see how we can prevent some of the errors on that list.

 Missing information. Errors of missing information can be blocked in two ways: By using a checklist and by reviewing information thoroughly. IEEE Std 1058-1987 provides a good checklist in the form of a plan outline or table of contents. It helps you remember to cover all information categories. If necessary, you can add to this checklist or change it to suit your own needs. If this checklist or table of contents is available in your computer system, you can call up a copy and conveniently insert data into appropriate categories. The use of computerized
standard outline templates is now commonplace. This template concept can be extended further to include Gantt or PERT networks. A standard software project network encompassing all activities and intertask dependencies can be kept on-line. New project managers can start from this standard network and delete nonapplicable tasks and intertask dependencies until they come up with their individualized plans.

All the checklists in the world cannot prevent all errors of missing information — we still need reviews. Reviews should be conducted by experienced project managers and people responsible for doing the work on a project. Quite often, a project plan is developed largely by a project manager without input from those who will implement the plan. In these instances, reviews are even more necessary.

The first error of missing information on the sample list is “missing every- thing.” The error of having no written project plan at all occurs frequently. Especially on small projects, managers often think they can keep track of all the pieces of the project in their heads. Whether or not managers can depend on their memories is irrelevant. How will the rest of the project team coordinate its efforts if the manager gets sick or leaves? Worse yet, if the manager has made a mistake, how can it be identified or corrected? Prevent this error by writing a project-management plan.

Number 2 is missing task descriptions for building test-support software. Test-support software includes stubs, drivers, simulators, databases, and testing tools. Most plans list an activity called testing, but that usually amounts to test execution. And getting ready to test can be time-consuming. Include in your checklist the activity of building test-support software. It should go in the project-plan activity list. In the table of contents of Std 1058, it goes under the heading “work packages.” And be sure the people who are going to do the testing agree with the activity description before you finalize the plan.

Third on the sample list is the schedule for reviews. Descriptions of requirements, design, and code reviews regularly appear in project-management plans. But schedule charts rarely show time to prepare for or to conduct these reviews. Small blocks of time in the network schedule templates should be allocated to preparing for and conducting reviews.

Often missing is number 4, a description of a user’s manual. Not only is the description omitted, but the time to produce the manual is not scheduled. In all project plans, there should be a list of deliverables. One way to make sure deliverables are remembered is to provide a comprehensive general list to project planners. Planners are more likely to consider all essential deliverables when they are reminded of many possibilities. Deliverables identified as unnecessary can be easily deleted from the list.

Now you have the idea of how to combat errors of missing information. Let’s move to the wrong-information category.

**Wrong information.** Hours of time can be taken up in a review of a project-management plan by people concentrating on misspelled words, grammatical mistakes, and sentence-structure problems. Indeed, these are errors and should be corrected, but take care of them quickly. Mark them for correction and return them to the author.

---

**Errors of wrong content are not easily remedied — and can lead to project cancellation if not corrected.**

---

Errors of wrong content, on the other hand, are not so easily remedied — and can be so serious that they cause project or product cancellation! These errors should be given the utmost attention.

There are three well-known ways to prevent errors of wrong information in a project-management plan: Assign experienced managers to the planning team, insist that they stick to proven planning techniques (often implemented with tools), and hold reviews with the people who are going to implement the plan.

Move back to the sample list now. Look at the first error in the wrong-information category, schedule estimates.

Occasionally, we see a software project that finishes on time, but this is the exception rather than the rule. No matter what we profess, generally we do not keep good metric information from project to project. So, our best bet is to bring in project managers who have completed similar projects in the organization recently. Draw on their corporate memory about scheduling, and ask them to augment the entire most-probable errors list with their knowledge and experience. Now the actual estimating can begin.

Estimating should be performed using two proven techniques: model-based and work-breakdown techniques. Model-based techniques give us reasonableness checks. They prevent wild estimates. Work-breakdown structure techniques let us look at a total project as a network of very small tasks. The probability of making errors in estimating the schedule for a small task is much lower than when estimating the schedule for a total project.

The people who will implement the plan will find the small tasks from the work-breakdown structure easy to review. They will be able to quickly identify errors and keep them out of the estimate.

Errors in cost and resource estimates can be prevented using the same approach and techniques.

“Poor organization descriptions” is wrong-information error 3. Most project-management plans include simple organization descriptions, but they don’t say who will be responsible for delivering what. What group X will deliver to group Y is often incorrectly stated. If this information is omitted, we have an error of missing information.

But, whether an error of wrong or missing information, the prevention is the same. In the work-breakdown structure, define for each activity inputs, outputs, and who in the organization is responsible for doing the work. That nails down individual responsibility and input-to-output flow for all activities.

Don’t forget reviews here. Your standard checklist should be used. We begin to see how to apply experience, techniques, tools and reviews to prevent errors of wrong information.

**Extra information.** Errors of extra information do not affect a project as severely as errors of missing or wrong information, but they do confuse people and slow down reading. Compare the standard checklist to your project plan. If information appears in the plan but not on the checklist, it is probably extra. Unlike errors of missing or wrong information, errors of extra information need not be corrected.

In the sample list, we see that product and tool details are common extra-information errors. The standard tells us they belong in other documents.

The sample list in Figure 1 is pretty high-level. You’ll want to add more details to your list.