

# Rules of Engineering Projects

Geoffrey A. Landis

April 2004 Analog

## Projects that fail...and succeed

From Murphy's Law (and its numerous corollaries) on through the Peter Principle, pundits have come up with hundreds of laws and rules to explain science and engineering. (Feynmann's rule: Never trust the data point furthest to the right<sup>1</sup>. Scotty's law: When the engines are overloaded and just can't take any more, they can always take just a wee bit more if the captain really needs them to.) Many of them are more humorous than realistic. (Despite sophisticated mathematical analysis, toast doesn't in fact always fall butter-side down. It's just that we don't remember the times when it doesn't make a mess.)

I've been working in science and engineering for a little over twenty-five years now. I've seen a few projects fly and a lot of projects fail. I've noticed that most projects end in a way that is neither spectacular nor even conclusive. If things failed by blowing up, at least you'd get some excitement, maybe even some fire and smoke. But most projects fail with a whisper, not a bang.

It may be true that in theory, even a failure is a result, but in the real world, most failures aren't obliging enough to be clear-cut and unambiguous—much more often, something doesn't work and you never really know why. In science fiction, an anomalous result in the lab means you've found some new science. In real physics, I've discovered, an anomalous result usually means an instrument error. (When in doubt, calibrate. Then calibrate again.)

The most typical project end, in fact, is a management failure—often one that happened years earlier. A project that has sputtered on for years after it's failed to achieve either a real success or a clear-cut failure finally just runs out of energy.

Still, I've seen a lot of projects fail simply from a lack of the combination of observation and experience that goes by the name of "common sense." So here are my observations on the laws of engineering, experimental science, and project management in the real world.

## Geoffrey's Laws of Engineering

1. The devil is in the details.
2. The best engineering material is paper. A design on paper always performs better than the one you actually have to build.
3. Starting projects is easy. *Finishing* projects is hard.

---

<sup>1</sup> Feynmann's observation: The experimenters certainly would have gotten a point even further to the right if they could have done so, right? So the last point is always right at the edge of what's possible—and hence is suspect.

4. In engineering, you can never ignore the laws of physics. In human endeavors, you can never ignore the laws of economics.
5. All failures are obvious in hindsight.
6. Failure is a milestone on the road to success.
7. Real advances are achieved by evolution, not revolution.
8. Design for the possibility that moving parts don't.
9. Every system designed to prevent a particular failure introduces a new mode of failure.
10. Never get rid of the old computer system until you have the new system up and running.
11. Engineering design consists of trade-offs. If you only see the advantages of your approach and not the disadvantages, you don't understand the problem.
12. The least important design constraint always ends up driving the design.
13. Program managers are the most conservative people in the world. They will never fly a new technology if it is possible to accomplish the mission without it.
14. Plagiarism is good engineering.
15. Progress is made when a project is invisible. With publicity comes management attention, and every engineering decision is subject to second-guessing.
16. To get where you're going, you have to start where you are.
17. Competition may be inefficient, but it's a lot more efficient than no competition.
18. Technology diffuses.
19. Technologies that succeed are built on the corpses of hundreds of ideas that sounded just as good, but failed.
20. Figures don't lie, but liars figure. Put your faith in numbers—but only when you've done the numbers yourself, and verified the assumptions.

## **Corollaries**

Some of these deserve a bit of elaboration, so here are a corresponding 20 corollaries to the laws of engineering projects.

1. Lots of designs look good on paper and you don't see the real problems until you get into the details.
2. Designs that have never been built in the real world don't have real-world problems. Just because somebody proposes a great design, and tells you how great it is, doesn't mean that it will be great in the real world. See rule #1!
3. Enough said.
4. If it can't make money, it won't fly. Advocates will always project the most optimistic economic scenario. You should view them with a jaundiced eye. Most economic projections are lies.
5. When something fails, everybody in the world will jump up and say that they knew right from the beginning it was a bad idea. (Same with successes for that matter.)
6. Since failures are milestones on the road to success, and all failures are obvious in hindsight, to succeed you have to accept that you are going to make "obvious" errors.

7. Everybody wants the breakthrough but nobody ever wants to accept slow, step-by-step progress. Don't expect the first design to achieve a breakthrough in performance. First get it to work, then work on improving it.
8. Every moving part is a potential point of failure. It's good engineering design to minimize, or eliminate, moving parts. (An automobile may indeed have a thousand different moving parts, but keep in mind that this is the end result of over a century of design evolution—and autos still break down.)
9. Enough said.
10. Mary's corollary: Keep the old computer for at least a year after you have the new one running, too.
11. This applies to other aspects of life as well.
12. If a design constraint isn't important to the design, dump it.
13. The best way to get a new technology flown is to make it look exactly like an old technology.
14. Real engineers never design something new if they can simply copy an existing design that works. Think of it as evolution in action.
15. And often third-guessing, fourth-guessing and fifth-guessing, too. To quote Bill Yerkes: "Work underground as long as possible—publicity triggers the corporate immune system."
16. It doesn't matter what we "should have" done twenty years ago. Where can we go from where we are now?
17. It's amazing how much improvement you get when everybody knows that they'll lose their jobs if they can't beat the other guy. Except in Congress, where competition is known as "wasteful duplication of effort."
18. An innovation that works in one design will find applications to everything else. And sometimes an innovation that doesn't work in one design will find application to somewhere else.
19. Anybody remember fluidic logic? Bubble memory? Cuprous-oxide solar cells? The Tesla bladeless turbine?
20. And remember that the devil is in the details.

### **Geoffrey's Three Laws of Robotics**

Robotics, too, is a form of engineering. Asimov, alas, got it wrong: his three laws of robotics, as it turns out, have nothing to do with the way real-world robots or artificial-intelligence systems behave.

To update the laws of robotics, I wrote down the three laws of robotics and computers the way they work in the engineering world.

**First Law:** A robot will do what it's instructed to do, no more and no less.

**Second Law:** The language in which instructions are given to the robot is designed to be convenient to the robot, not you.

**Third Law:** All consequences of the robot's actions are the responsibility of the programmer, not the robot. The robot doesn't know, or care, about consequences.

These three laws have certain corollaries.

*Corollaries to the first law:*

1. It doesn't care what you thought you said.
2. The way in which the robot interprets instructions is up to the robot, not up to you.

*Corollaries to the second law:*

1. Robots don't understand English.
2. If robots simulate understanding English, the English they understand is not the same as the English you speak.
3. If the language seems to be clear and straightforward to you, this is an illusion.
4. No matter how well-documented the language, some commands are always undocumented.
5. The undocumented commands always include "halt and catch fire."

*Corollaries to the third law:*

1. Causing harm to a human, or through inaction causing a human to come to harm, is of no concern to the robot.
2. It doesn't care about whether it causes harm to itself, either.
3. Those "keep out" zones are there for a reason.
4. Artificial intelligence isn't.

So there you have it. If you want to build some robots, or succeed in engineering—or in life—here are some simple rules to keep in mind.