If a nonprofessional makes an error, the employer cleans up the mess. But when a professional makes a mistake, he cleans up the mess.

Apologies are necessary, but insufficient. You cannot simply keepmaking the same errors over and over. As you mature in your profession, your error rate should rapidly decrease towards the asymptote of zero. It won’t ever get to zero, but it is your responsibility to get as close as possible to it.

Every single line of code that you write should be tested. Period.

To design your code to be easy to test. And the best way to do that is to write your tests first, before you write the code that passes them.

The employer’s problems are my problems, and need to understand what those problems are and work toward the best solutions.

Good code should be extendable. Maintainable. It should lend itself to modification. It should read like prose.

Deadline is coming, I collapsed from exhaustion.

As developers, we’re going to be asked/told/conned into writing twice the code in half the time if we’re not careful.

Creative output depends on creative input.

Do not incorporate hope into your estimates!

There is no way to rush. You can’t make yourself code faster. You can’t make yourself solve problems faster. If you try, you’ll just slow yourself down and make a mess that slows everyone else down, too.

The tests should be understandable by the stakeholders and business people, and should be run frequently.

We require disciplines that drive us to collaborate.

Nothing can bring a young software developer to high performance quicker than his own drive, and effective mentoring by his seniors.

When seeing a messy function: the first thought is: It needs to be cleaned. And then, the second thought is: Do not touching it! Why? Because you know that if you touch it you risk breaking it; and if you break it, it becomes yours.

Patients do not pay doctors to practice sutures.

The only way I know of to effectively eliminate communication errors between programmers and stakeholders is to write automated acceptance tests.They are completely unambiguous, and they cannot get out of sync with the application. They are the perfect requirements document.

Professionals fear messes far more than they fear blind alleys.And they are always on the lookout for growing messes, and they clean them as soon as they are recognized.

There is no sadder sight than a team of software developers fruitlessly slogging through an ever-deepening bog.

Professionals don’t make commitments unless they know they can achieve them.

When professionals make commitments, they provide hard numbers.Rather, they provide probabilistic estimates that describe the expected completion time and the likely variance.

Professional developers work with the other members of their team to achieve consensus on the estimates that are given to management.

I determined that I was going to enjoy my career by doing it well, not by doing it stupidly.

Sleepless nights won’t help you get done any faster. Sitting and fretting won’t help either.

Instead, slow down. Think the problem through. Plot a course to the best possible outcome, and then drive towards that outcome at a reasonable and steady pace.

Avoid creating surprises. Nothing makes people more angry and less rational than surprises.

The trick to handling pressure is to avoid it when you can, and weather it when you can’t. You avoid it by managing commitments, following your disciplines, and keeping clean. You weather it by staying calm, communicating, following your disciplines, and getting help.

It is unprofessional to be a loner or a recluse on a team.

Programming is all about working with people. We need to work with our business, and we need to work with each other.

If the business decides that one project is higher priority than another, it should be able to reallocate resources quickly. It is the project owner’s responsibility to make the case for his project.

A craftsman is someone who works quickly, but without rushing, who provides reasonable estimates and meets commitments. A craftsman knows when to say no, but tries hard to say yes. A craftsman is a professional.

School can teach the theory of computer programming. But school does not, and cannot teach the discipline, practice, and skill of being a craftsman.