Dr. Way's Tips for Software Engineers

An incomplete collection of rules-of-thumb for the software engineer gleaned from life experience.

Be good to the gatekeeper

The office manager or project secretary holds the key to making your life easier at work. Remembering that this person is often the hardest working and least-appreciated member of the team, and recognizing the hard work and offering appreciation will go a long way.

Always have an answer

When asked "how long will it take", never say "I'm not sure." Say something, anything... "three weeks, assuming everything goes well" has always worked for me.

Help the team

If a fellow engineer or programmer is stuck trying to fix a resistant bug in their code, offer an extra set of eyes. If they are moving next weekend, see if they could use help carrying boxes.

Adjust your thinking

Unlike the familiar and comforting 3-4 month semester cycle in the world of academia, the world of software engineering has no such real pattern. Even though there are releases of new versions of software periodically, even frequently if you work in an Agile environment, in reality there is a continuing, day-to-day, sameness that takes some getting used to for many people. Prepare yourself mentally to expect very few definite finishing points and you will do yourself a huge favor.

Become a programming language polyglot

Learn as many languages as you can: C, C++, Java, Perl, Php, Html, Basic, Visual Basic, shell scripts, etc. The more you know, the more valuable you will be both as a programmer and eventually as a manager. Spend a couple minutes once in awhile trying out something new.

Be an ethical thief

The Internet is an amazingly rich resource for ideas and examples, offered up for free and judicious use by you! Before "re-inventing the wheel," spend a little time seeing what solutions already exist for the problem you are trying to solve. It will save you an enormous amount of time and energy in the long run, even if the example code you find is mostly junk. Be sure to play nicely, and give credit (and payment) when it is due.

Communicate meaningfully

A quick email reporting your latest progress to the project leader and your fellow engineers can be the glue that holds the project together. If everybody has at least a general idea of what everybody else is working on, the team can function better and the result will be better for all. Resist the urge to pile on in email flame-wars. I have never seen any progress come of it.

Take a walk after lunch

No matter what else is going on, taking a walk right after lunch will benefit you and the overall project immeasurably. If you run or bike or workout, that's good, too. Remaining in good health will carry you through the days when you can't stand the current task you've been assigned.

Test early, test often

From the first line of code you write, constantly test it. Look at it. Judge it. Write a small test program. The old saying "A stitch in time saves nine" applies directly to software development and engineering. A small mistake now, left unrepaired, can lead to major headaches later. By constantly keeping your code in good shape you'll put yourself in the best possible position. Test -driven development? Even better.

You will eventually like it

Even if you hate what you are doing right now (and this will probably happen at some point), I have discovered that if you work on it long enough (6 months to a year) you will start to enjoy it a little, and then more and more. Eventually, when the project ends and a new one begins, you will look back with some nostalgia about how much you liked working on that project you hated, and especially how much you will miss the people you worked with.

Specify, but not yet

A well-written specification is a wonderful thing that can lead to very solid software. But it is hard to get motivated to write it down first. So, first create a little prototype. Write a little program. Add to it. Show it off. Then jot down some ideas about it. Program some more. Write some more. Eventually you'll get to the point where you can envision most of the finished product from the little prototypes you have created, and it'll be much easier to write it all down. Be Agile.

Accept that you will carry the load

There will be times when you are the expert, and where some of your fellow engineers are little more than programming dinosaurs. This will happen, and there is nothing you can do to avoid it. Forge ahead, carry the load. One day you might discover that you are now the dinosaur and that new person is the whiz-kid.

Bring your lunch

A bag lunch is cheaper and healthier. You're going to make a lot of money as a software engineer. Why waste it on junk food? And why not be around longer to enjoy it?

Practice hiding your goofing-off

Get good at appearing instantly busy. You know how hard you have been working all morning, and how you just need 5 minutes to read CNN or check Facebook. You are actually more productive if you DO take little mental-health breaks like that. But getting caught doing it by the wrong person can be bad. So, always have your work open and ready. Position your monitor so it isn't visible from your office door (it’s called a Slacker Shield). But try not to be too paranoid. Everybody (with a life) does it.

Seek out the wise person

Somebody on your engineering team knows just about everything there is to know about just about everything (at least about your project... usually more). Befriend them. Listen when they speak. Watch what they do. Absorb. Let them rub off on you. And don't pretend to be this person yourself. You aren't. Yet.

Suggest, then duck

Don't be afraid to suggest a new idea or approach. And don't be surprised when you are shot down. You many have many very good ideas which may receive resistance from those who have been around longer. Don't be discouraged. If you suggest something every few months, at some point an old-timer will "come up with" the same idea. That's just as good. Remember, the goal is to improve the product or project, not to stroke your own ego. Your good deeds ultimately will be recognized.

Have a life

During crunch-times you may need to put in long hours, but doing it on a regular basis will burn you out. Instead, make it a point to leave at the same time every day and have some non-computer activities and interests that you pursue. Don't look back in 20 years and wish you had gone to more movies, read more books or spent more time with your family.