Going From Good To Great: Becoming **A Better Software Developer**

WHITEPAPER

ZEPHYR
Real-Time Test Management

Table of contents:

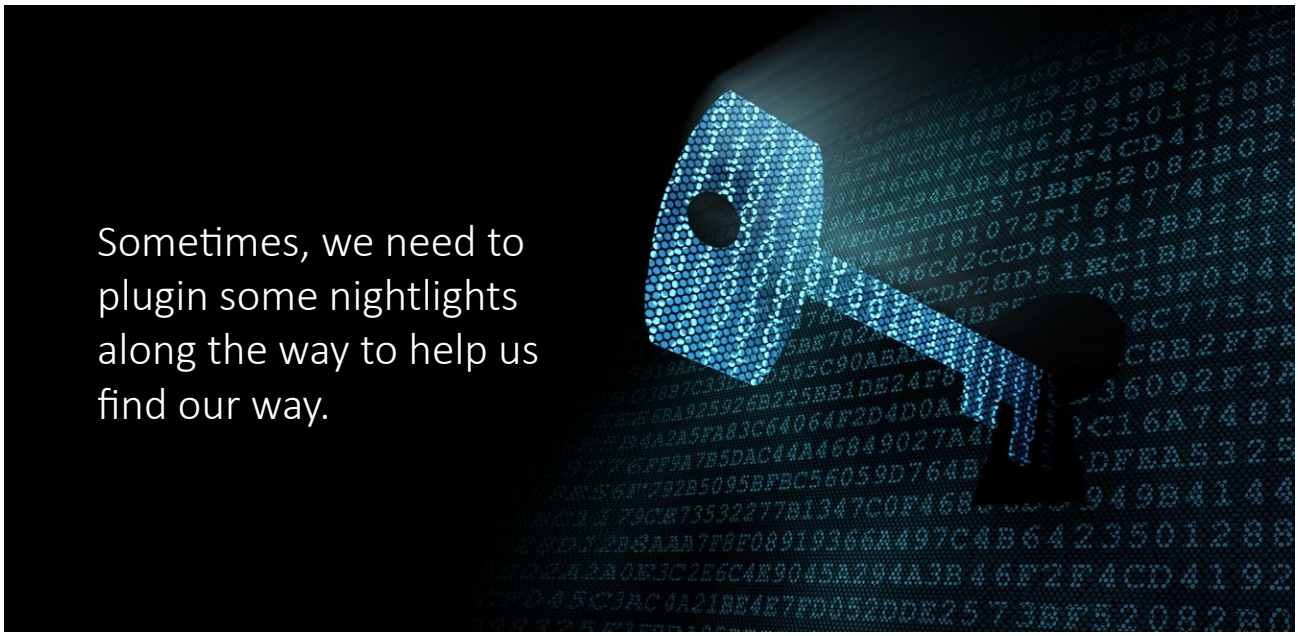
Becoming **a Better Software Developer**

It is really easy to get stuck in your career and to feel like you are either not making any progress or not making as much progress as you would like.

In my home, my office is at the far end of the house. Often, when I stay up late to work on a project, I have to try and find my way back to the master bedroom in the dark. I blindly grope my way through the darkness tripping over toys and running into furniture on my way to my destination.

Finally, after many such nights, it occurred to me to plug some nightlights in to light my path. Now, I can see where I am going and not only can I get there much faster, but I get less bumps and bruises on the way.

It isn't all that uncommon for us to progress along our careers as software developers in the same way that I was hopelessly trying to find my way back to my room in complete darkness.



In this article, I am going to give you four night lights you can use to help see your way in the dark and make real progress with your career.

One of the biggest mistakes I've made in the past and one I frequently see many developers and non-developers make is not sticking with what is working. We tend to get bored doing the same thing over and over again, even if that thing is our best chance of success.

Bringing it back to programming terms, it is likely that you have skills in a particular technology that you've developed to a point of being really good, but those skills need just a little bit more work to be great.

It is tempting to want to move on and learn some new technology or turn from the success you've had in the past and journey down a new road.

I often have to force myself to honestly evaluate what seeds in my garden are growing the best and to focus on cultivating those flowers rather than always planting new ones and letting my best plants wither.

Take some time and honestly evaluate your skills and determine what skills and processes are working best for you. You have a much greater chance of making a breakthrough in one of these areas than you do of starting over down another path and making a breakthrough there.

If you can identify those areas you already excel at and the skills that are bringing you the most success, it is like lighting the path to your future. No need to backtrack when you are already on the right path. Just recognize that path and go forward with it.

I'm not going to lie; this is often a painful and boring process. It is much more fun to try something new, but if advancement is what you seek, sustained and focused effort is required.

There will be time for learning new technologies and developing new skills, but make sure you focus on what is working first, if you want to have the best chance of making the most forward progress in your career.

Improving problem solving skills

Software development wouldn't exist without problems that need to be solved. The very point of the software we write is to solve some real world problems. If it weren't for problems, our jobs wouldn't exist.

So, it follows to assume that to become a better software developer, to move from good to great, we have to enhance our ability to solve problems.

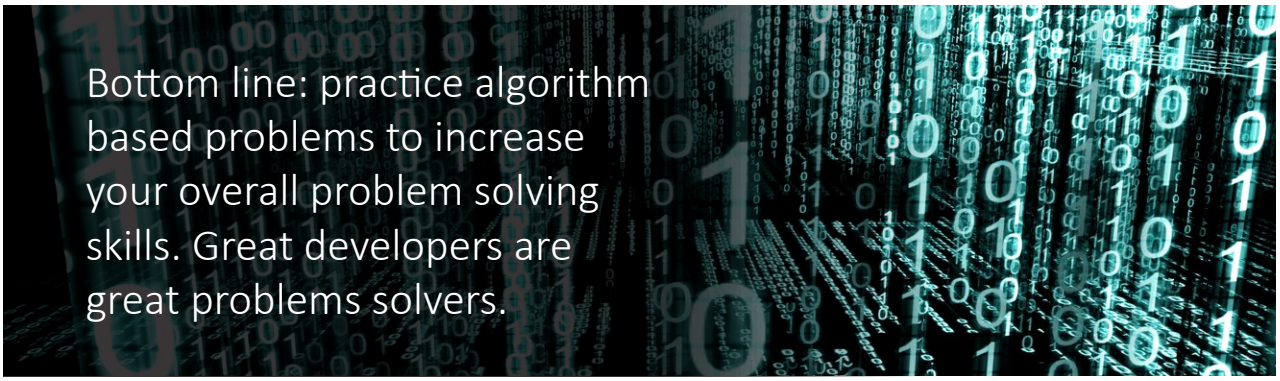
The problem with solving problems is that problems come in all kinds of shapes and sizes. But, even though there is a large variation in the kinds of problems we might encounter and need to know how to solve, we can improve our overall problem solving skills by actively practicing them.

One of the best places to start is with algorithm problems. These kinds of problems are problems which require us to develop an algorithm to solve and write code to implement that algorithm. An algorithm based problem may be something as simple as writing an algorithm that can reverse the letters of a string, or it might be as complex as writing an algorithm to solve the issue of finding the shortest route for a salesman to visit all the stops he needs to on a cross-country trip.

Practicing these kinds of problems puts tool in your toolbox that you can later use when you find a similar problem in a real world situation, and it gives you great practice using your programming language of choice in ways that stretch your abilities with it.

There are several good books you can find that contain some algorithm type problems you can practice with. (One good one that I recommend is Programing Pearls by Jon Bentley.) But, one of my favorite resources for developing these skills is a website called TopCoder (http://topcoder.com). TopCoder has an algorithm competition section where you can compete against other developers in developing solutions to algorithm based problems. They have a huge backlog of previous problems that you can practice and they have weekly competitions. Even if you don't compete, you may want to take a look at the previous competitions and practice those problems.

Even though the kinds of problems you solve in your day-to-day job may be less constrained and clear-cut as the algorithm type of problems you would solve on TopCoder, you'll invariably find that the mental process of solving these kinds of problems will actually enhance your problem solving skills in all areas. I've also found that once you have an idea of the general types of problems that exist, you are much better equipped to recognize variations of those problems in real world situations that you wouldn't have seen otherwise.



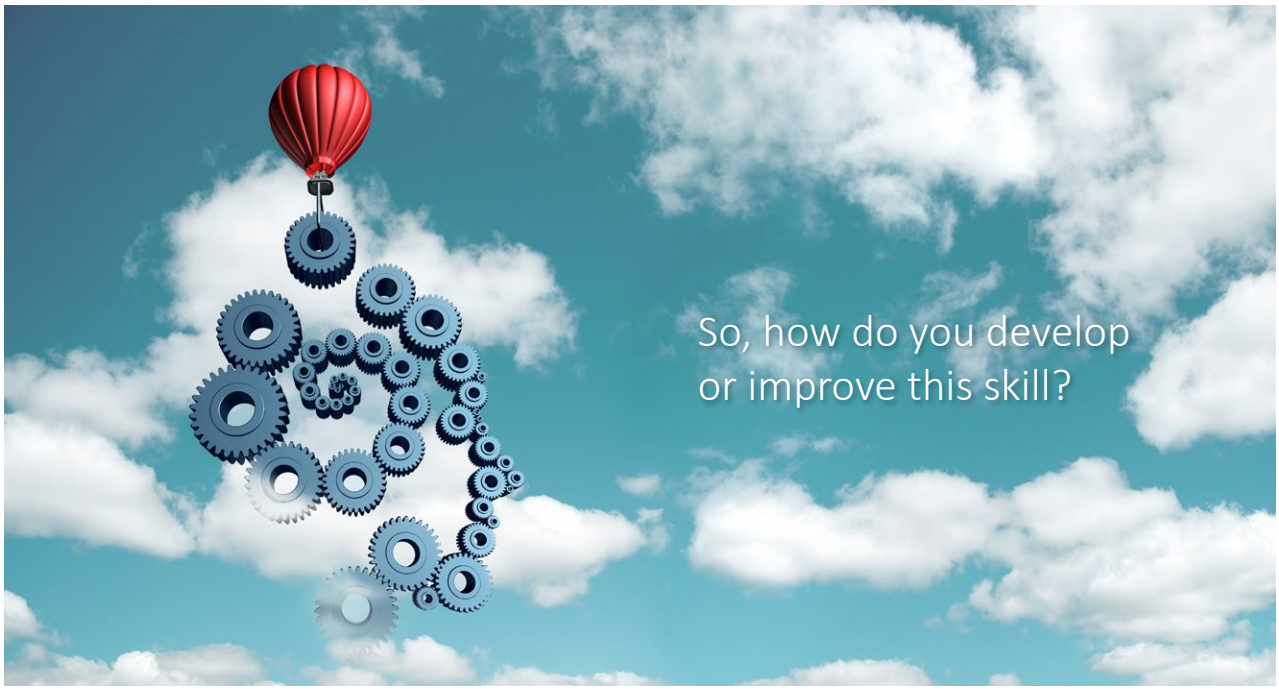
Learning how to **learn things quickly**

The next major area that most software developers can focus on to really increase their value and take them from good to great is learning ability.

The field we are in is constantly changing. It used to be that programming languages and operating systems changed perhaps every few years, but today things change on a daily basis.

Take a look at the evolution of JavaScript frameworks just over the last few months. Even by the time you read this the tech-scape will be significantly changed from when I wrote this very sentence.

I'm sure I don't have to spend anymore time convincing you that the ability to learn quickly is a huge asset for a software developer today.



Surprisingly, the first step to increasing your capacity to learn rapidly is to let go of your resistance to not do it. This is more difficult than it sounds.

We all tend to have what is called a confirmation bias—our brain tells us what we want to hear. Part of what our brain tells us is that we are already using the best technology or programming language and that what we don't know is not very important.

Now, it is quite possible you've gotten past this hang-up. But, most software developers, myself included, from time to time, have serious problems letting go of our religious feelings towards familiar technologies which we are already good at and understand.

The single biggest barrier to learning new things is getting rid of this confirmation bias and realizing that your skills and abilities are like sand in your hand. The more you try and hold onto them, the more likely they

will be to slip through your fingers. The best strategy is to always be getting more sand.

Honestly, once you've gotten past this hurdle, rapid learning isn't all that difficult. Our brains are amazing at absorbing and processing new information. On a daily basis your brain deals with a huge volume of information; you just have to trust it to do what it is supposed to do.

I've actually developed a 10 step process I use to learn things quickly. I've developed this process over the course of several years creating online training courses for developer for a Pluralsight.

I've broken it down into ten steps, but really all I am doing with the process is defining clearly what it is that I want to learn, charting a course to do it, and finding the appropriate resources to execute my plan. Learning is a much simpler process than many people believe or would lead you to believe. As long as you have a clear goal in mind, find the right resources to gain the knowledge, and are willing to put in the time required to do it, you can learning anything and you'll probably be surprised how quickly you can accomplish it. You don't really need a teacher other than yourself, especially today when there are so many resources available for just about any kind of topic you can imagine. Take responsibility for your own education. You are your best teacher after all.

I'll offer one more tip about learning quickly that is technology focused before we move on. If you want to learn a new programming language, framework or technology, your time is best spent actually using it to build something useful rather than just reading books. Don't be afraid to dive in and get your hands dirty. What is the worst that could happen? You get stuck and don't know what to do? When you take the approach of diving in first, you develop relevant questions that you want to find the answers to. Then, when you turn to a reference to find those answers, you appreciate what you are learning more, and it is much more likely to stick. If someone tells us some information or a fact, we are likely to forget it. But, if we ask them a question and get the same information, it is much more likely to be remembered.

Selling yourself

One of the most overlooked ways to boost your career and create more opportunities for yourself is through active self-promotion, or marketing. Most software developers I've talked to haven't even considered the idea of marketing themselves.

Marketing tends to get a bad reputation, because bad marketing leads us to imagine spam emails in our inbox trying to sell us cheaper Viagra, or the idea of a pushy salesman trying to get us to buy his product or service.

But marketing is a very powerful and necessary element of commerce. And, like it or not, you are part of the economy of software development.

Consider, this very article. Why am I writing this article? I am writing this article primarily with the motivation of it being a good opportunity to market myself. But, notice the way in which I am doing it. I am providing value to you and through that value I am hopefully increasing the stock you hold in my abilities or services. If I don't provide you value, but simply say "I'm the best" or "hire me" I'm not doing a very good job of marketing myself and you are much more likely to just ignore me. (In fact, doing something like this is likely to have the very opposite effect.)

The point is, there is a huge value in providing other people value and doing it in a way that gets your name out there and builds your own credibility.

This doesn't mean that you have to be self-employed or run your own business. (Although, it is certainly arguable that it is extremely important in that case.) As an employee, you'll find that learning some good marketing skills will increase your value and worth to your organization and will often lead to increased opportunity and promotions.

I've spent quite a bit of time developing marketing plans for software developer careers and coaching software developers on marketing themselves. In that time, I've learned a few basics that I will share with you here.

First of all, you need to think of yourself more like a business. Most businesses have some kind of branding, and you should too. A brand is simply a set of expectations about a product or service. You can build your own personal brand by clearly defining what you are about and what kinds of services you provide. In doing this, it often helps to "niche down," which basically means to pick a narrow and specific target you use to define who you are and what you are about, (at least in a professional context.) For example, you could be the C++ pointer guy or the JavaScript framework news girl. Each of these are very specific specialties which would lend themselves well to branding. Most brands have a logo that is easily identifiable and helps to associate people with the expectations you are trying to promote, but it is possible to have a brand without a specific logo.

Once you've established your brand, you'll want to carefully craft and control the message you are putting out through that brand. If you want to position yourself as an expert in C++ pointers, you should probably have some kind of home base or blog that you can use to project your message. A book, magazine article or conference appearance can also easily fill this role. I utilize many mediums to promote my brand, including: blogs; YouTube videos; podcasts; magazine articles; books and more. Whatever mediums you choose, you need to provide a clear and consistent message about the value that you offer.

These two concepts are the core to marketing yourself. First develop a brand, then use various mediums to promote that brand by offering a consistent and clear message. But, in order to get someone to pay attention to you and listen to what you have to say, you need to be providing them value. We've already talked about this part, but I want to drive home the difference between shameless self-promotion and marketing yourself through providing real value to others. One will make you look like an arrogant tactless person, and the other will build a solid reputation.

Strive toward **continuous improvement**

If you want make that transition from merely good to great, it will do you well to consider these areas I've presented in this article. First, by starting out and evaluating your own path to success and what is making you the most successful. (Do more of it.) And secondly by focusing on three key areas that have a big potential impact on your career: problem solving, rapid learning, and marketing yourself.

But, perhaps most importantly, you have to subscribe to the idea of continuous improvement. Nothing in life is static, certainly not technology or our careers. We have to take an active role in constantly evaluating where we are now and were we want to be if we want to constantly improve.

About Zephyr

Zephyr is a leading provider of quality management solutions, powering intelligent DevTestOps for more than 11,000 global customers across 100 countries. Project teams and enterprises of all sizes use Zephyr's products to enable continuous testing throughout their entire software delivery pipeline to release higher quality software, faster. Zephyr is headquartered in San Jose, CA with offices in King of Prussia, PA, Europe and India. For more information, please visit www.getzephyr.com.

ZEPHYR
Real-Time Test Management

Contact Zephyr Today!
sales@getzephyr.com
www.getzephyr.com
+1-510-400-8656