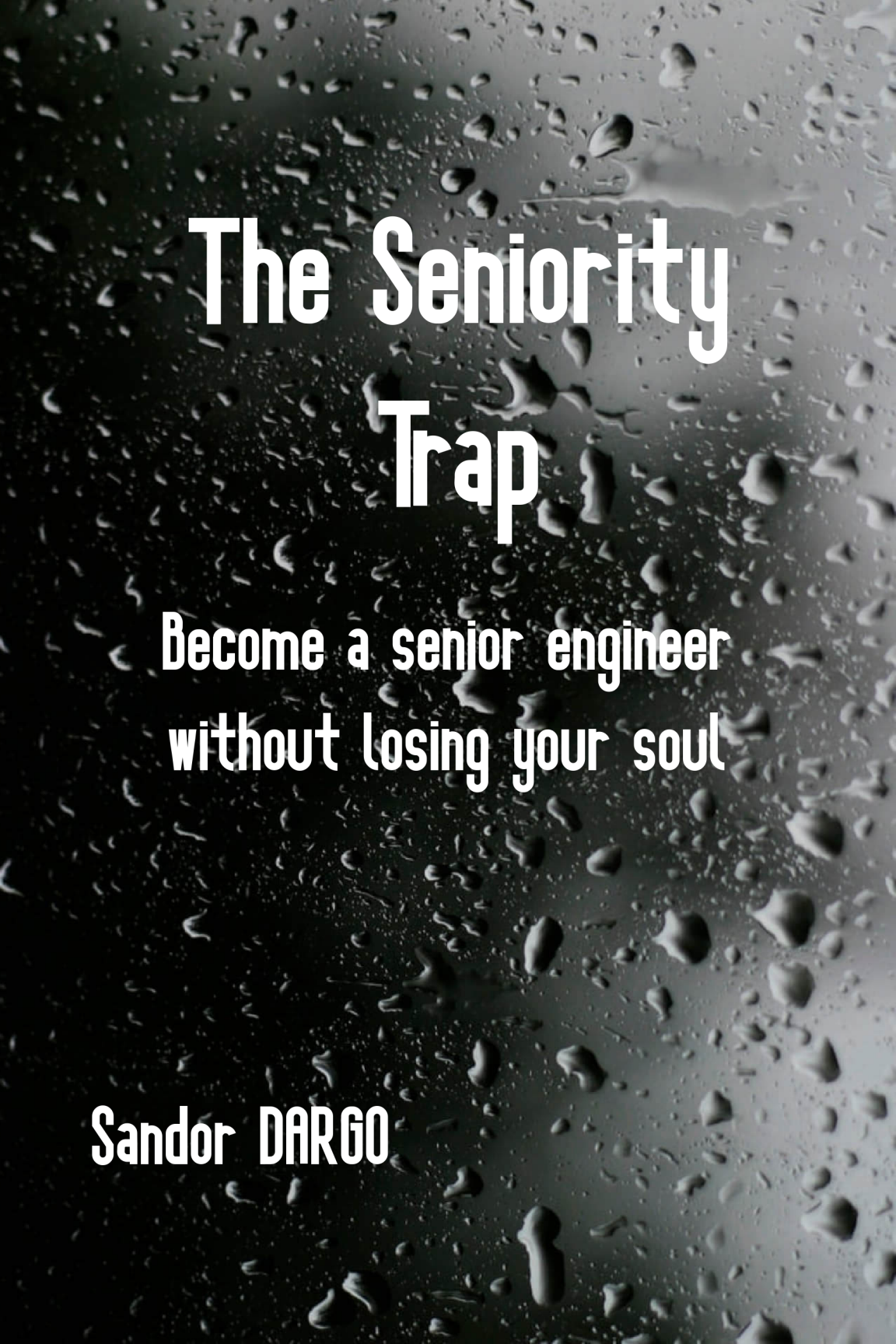

The Seniority Trap
Become a senior engineer without losing your soul
Sandor DARGO

# The Seniority Trap

## Become a senior engineer without losing your soul

Sandor Dargo

This book is for sale at http://leanpub.com/theseniioritytrap

This version was published on 2023-07-08


This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

# Contents

# About the author

Hello, my name is Sandor.

Thanks for buying my book.

I'm going to share quite a few personal stories in the coming chapters. I'd like to share a bit about my background so that you understand better my message.

I was born in 1985 in Hungary 4 years before the fall of the iron curtain.


**My motherland, Hungary in the heart of Europe**

I started elementary school in 1991. I was among the first generations who did have to learn Russian but I could learn English instead as a second language. A few years later I was also among the first guys who didn't have to serve half a year in the military.

I grew up in a modest and loving family. Money was always scarce and it gave me unpleasant thoughts. I fell in love with computers early, but mine was always way outdated while my best friend had always a decent one. It was not only about computers and it often made me sad.

We were living in the suburbs and I was dreaming about breaking free.

I must have inherited some good genes. I finished elementary and high school without lots of effort. It was enough to pay attention during classes and I passed with the best grades. That's something that backfired later on during university. I didn't know how to learn.

While I knew from around the age of 10 that I would like to work with computers, I became hesitant at certain times during university. Those years were difficult. I didn't know how to learn and half of my classes were in French. For the last two years, I choose a specialization that was the least close to pure IT and a bit closer to management.

After graduation, I learned quickly that your specialization doesn't matter much.

I started my corporate career as a database administrator.

Being a DBA in a heavily process-oriented company didn't offer me enough challenges and joy. On the other hand, it offered me further French lessons.

One day while I was in a French class when my phone rang. It was a French number. I answered it.

A few months later, with my wife, we started a new life on the French Riviera and I started to work as a software developer.

**Antibes (France), the city where I've been living since 2013**

It was difficult. A new job in a new field in a new country. I received so much help from my colleagues. I'll always be grateful. They are the main reason why I'm so obsessed with knowledge sharing.

They are the main reason why I ended up writing this book.

You'll get the details later.

I hope you'll enjoy *The Seniority Trap*!

# Introduction

I'm a young person. I'm 36 years old.

*What? Is that young?*

I can hear you!

In our profession that is considered old!

I hopefully haven't lived through half of my life, yet sometimes in my profession, I feel old when I look around.

I'm working in IT for a bit more than a decade and I'm considered senior. After a few years of database management, I switched to software development around 9 years ago and now I work as a principal engineer.

Tell me any other profession where it is possible to be considered senior after so little time!

## The number of developers is constantly growing!

There is no surprise to that. During the last few decades, the number of developers has roughly doubled every five years. While there are many people nowadays switching their careers to become programmers later in their life, most of the new programmers are still young people.

They are just out of school around the age of 23. It means that half of the developers started their careers within the last 5 years and only the rest, about half of the developer population is more than 28 years old. Half of the half is more than 33 years old and… You

are a smart person dealing with computers, you can figure out the rest.

## We lack respectable mentors

This means that there are not many mentors. At least not enough. At the same time, it also means that when you are 36 you are definitely among the older programmers. One would say, you are a senior.

I disagree with that conclusion. I disagree with that with my whole heart and my mind. I remember that my friends kept telling me when I was around 21 years old that I have to respect this and that person because of their age.

I said I couldn't care much about one's age. Obviously **everyone deserves to be treated respectfully**, but respect is not a right, respect is earned. Nobody should be respected more than the others simply because they stayed alive in the world for a longer period of time. That kind of extra respect must be merited. Let me see what you achieved in your life and I'll respect you based on that.

It's clearly not the majority who earns such respect. By definition most people are average and being average is not something admirable.

You might say that average people can still teach, they just need seniority. Fine, but…

## What is seniority after all?

Following the age analogy, in my eyes, you will not become a senior developer because you managed to keep the same semi-comfortable chair under your bottom for long years. Seniority comes from applied knowledge. From performance.

What is seniority? And does it necessarily need a certain number of years? Does it necessarily result in endless meetings and more and more spreadsheets and project reports to fill?

The answer is not clear. I think it's difficult, but not impossible to completely avoid getting buried under spreadsheets. I managed to seriously limit the number of meetings I have to attend and limit the number of activities that I dislike, yet I managed to grow not just considering what I know about developing software but also in terms of corporate status.

To be fair, most of the materials that helped me to grow were not programming books. Technical books are important to increase your level of technical competency but it does not have that much to do whether you'll rise to the level of a senior developer or beyond.

Self-help books, psychological books, management books and career advice from successful people were much more helpful. You've read that right. And I'm not a manager and I don't even plan to be one. I'm a *Principal **Engineer***. An individual contributor without formal authority, yet with a broad scope and with some influence over a department and sometimes beyond.

For years, I didn't understand the rules and while I was more and more competent technically and I helped the others around me to write better code, my end-year reviews always left me in deception. It took me a few years to understand the rules of the game. With the help of some books, with the help of my senior manager(s), in 3 years I was promoted twice and I made a journey that I thought would take me at least double the time.

Finally, I understood that seniority is not about delivering more and better. It is about helping others deliver more and better.

## You don't have to be special

There were times when I felt trapped. I thought that I was losing my identity as a developer and I either go for more money and give up hands-on development, or I continue coding but I won't be able to have a bigger impact and receive the compensation I was hoping for.

I don't feel trapped anymore and I think created a job and an environment for myself that I was looking for.

I'm not a particularly talented person and I don't work 10 hours a day - instead, I read and learn a lot even outside business hours. I'm convinced that if I could do it, you can do it too.

I'm not special, the vast majority of us is not special at all. While I acknowledge that no two paths are the same, no two lives are indistinguishable, we all face very similar issues and questions throughout our careers.

# What are you going to find in this book?

The order of the chapters follows a logical order of the questions and problems we face in our careers. In the beginning, we have that daunting question of what kind of company we should work for.

I deal with corporations in this book because that's what I have experience with. It doesn't mean that the information and advice you'll find in this book will be useless for you if you don't work for a corporation, but probably applies less.

Often, it's difficult to understand, difficult to adapt to the complex structures, hierarchies and strange culture of a company (*chapter 1*). At the latest, once we are more or less comfortable with the dynamics, we should get rid of the junior mindset as it limits our

reputation, our answers to problems and our perception. Sometimes we have to change teams for that and it's fine (*chapter 2*).

I fully agree with Cal Newport who wrote in So Good They Can't Ignore You[1] that many young people make a big mistake when they look for their calling, when they look for their passion, when they choose their career, their first employers. We should work hard on something, anything that we find at least interesting and then passion will come. It doesn't mean that our choice doesn't matter, but we shouldn't look for something that we are passionate about. We should look for something that we **can be** passionate about as we'll discuss in *chapter 3*.

In our career, we'll sometimes succeed big but we'll also fail. Even the greatest people fail from time to time. Or better to say, the greatest sometimes succeed, other times they learn something new. That's a mindset to live by as I explain in *chapter 4*.

In *chapter 5*, I explain that during your career there will be very few things that you'll be just given. Most of the rewards have to be taken or at least you have to ask for them specifically. While it's not granted that you'll get what you want, for sure you'll not lose anything by asking.

Other times, your duty will be more than asking. You'll have to stand up for yourself, for others. As I explain in *chapter 6*, standing up for something involves conflicts. You should avoid them most frequently, but there will be some cases when you must stand up for your values, for your beliefs. Some people will not like you for that. But they will respect you.

As you grow as a developer, you'll have to make some hard choices regarding your career. Well, hopefully. If you have to decide how much your coding time are you willing or you want to give up for more responsibility, it means that your career isn't just happening to you, but you actively form it. Anyhow, accidentally or willingly,

[1] https://amzn.to/32RT3LH

you might end up in a situation that I call **The Seniority Trap** (*chapter 7*) the focal point of this book.

In *chapter 8*, we'll discuss that sometimes you'll be offered seemingly great opportunities but they are non-coding roles that you'd suffer from. Their titles look great, they pay well, but in the long run, you'd hate yourself for choosing such a position instead of doing what you love.

I think you should do what you love, but as I explain in *chapter 9*, it doesn't mean that you have to give up on growing authority and responsibility. If you are proactive, you can turn essentially non-coding tasks into somewhat more technical challenges, you can turn boring solo tasks into fun team efforts. And while you enjoy yourself, you demonstrate skills that management is looking for in people who they want to promote into senior technical roles.

On your road to senior (technical) roles, there is one thing that is probably more important than anything else. To stay focused. In order to stay focused, you must limit the number of your goals (*chapter 10*). Those who have too many goals have none. Focus on no more than 3 things and with practice, perseverance you'll perform well. You'll succeed.

But as we see in *chapter 11*, success is the biggest threat. It's one of the main reasons for failures. Don't become comfortable, if you do, you'll fail. Don't go for status, because you'll easily become content. Look for growth, not for status and there are no limits in your career! Keep in mind, that you will never reach your ideal, as Matthew McConaughey said, your ideal is always somewhere 10 years away from you!

At the end of each chapter you'll find some challenges that if you accept and complete, you'll make big steps forward in your career. You might not be able to complete all the challenges at the first read, you'll have to revisit them from time to time and even redo some of them.

Now, let's get started!

# Chapter 1: The value of corporations for a junior developer

Where should we start our careers? Should we join a big corporation and be a small cog in a huge machine or should we rather join a small company, maybe a startup where we'd feel that we have a more direct impact on the product?

There is no black and white answer to this question, I don't even try to give one. After graduating, I chose a corporation. In fact, a corporation that according to the NY Times was the most process-oriented company in the world at some point in time.

In this chapter, I'm going to share with you the different advantages and challenges a corporation can offer to you.

## Searching boredom

While I was not particularly lazy, I found my job rather boring. I experienced one the five different boredoms[2], *searching boredom*. It is marked by actively looking for something to do.

I tried to grab every opportunity to avoid my daily work.

The job was boring, but safe at least. So safe, that I wouldn't even have considered changing a job. I used to be a strongly risk-averse person.

In the beginning, my job was interesting, but after spending 2 years as a database admin in a process-oriented corporation I kept asking

[2] https://www.inc.com/jessica-stillman/boredom-apparently-comes-in-5-flavors.html

for changing teams. I emphasize the expression *process-oriented* because it can be quite discouraging for innovation.

During one of my last discussions with Luca, a DBA manager in my organization, I was told that our DBAs can't even follow simple processes, so let's not give them options. In hindsight, Luca might have been right, but it felt belittling.

As I figured out later, those 2 years were not so much of a value on the market, even though I did a lot of side activities to keep myself awake. Probably those extracurricular activities were the most useful in terms of personal growth.

Coordinating a continuous improvement task force to analyze and fix the most often recurring problems, automating boring and error-prone processes wouldn't have been expected from me at that point, but such activities gave me a reason to go to work and I became a much more autonomous professional.

Yet, I was bored.

How could I have had some fun from 9 to 5?

## Keep drinking coffee far from your desk

Spending hours in different coffee rooms, laying in big fancy sofas! Oh boy, we had many of them! But it's not just boring alone, in addition, it's suspicious… Other people can see you. Including managers, spies and sneaks.

Maybe if I spent that time with someone else? For sure! I've seen people spending several hours drinking coffee and chatting about non-business related stuff on a daily basis.

A guy managed to work about an hour a day for years until he was finally fired. The rest he spent at coffee rooms with others. But I'm an introvert. If I had to chat so much… It would not be just boring, but something much worse! Stressing and exhausting!

There must be something better to do in a corporation as a database admin - I thought.

I mean apart from deleting some files to free up space on servers and therefore fixing one more ticket in an always replenishing queue.

## Internal events to the rescue

A great possibility is attending as many internal events as possible. Some or maybe even most are boring, but you can always go with a laptop. I could have written *with your smartphone*, but that's still dangerous. Would the others think that you mindlessly scroll on social media or that you are answering your business mails and chats?

With a laptop, people won't think you are just entertaining yourself. Or… Anyways, it's not so evident. And if the event turns out to be interesting you can even take notes!

And if not?

You'll play solitaire before going back to your desk with a busy look on your face.

Those internal events have tons of advantages. At least they had before the COVID took most of them away. But as office workers are going back to the offices, these advantages are coming back too. If you stayed remote, you can still register for these events, share them with your colleagues and make them publicly visible in your calendar.

## Free food

If you are on-site, you might get some free food. Lots of free food.

If you are lucky, you even like it.

If you are even luckier, you don't like it so you won't gain weight.

But if you are among the luckiest ones on Earth, you like it and you manage to cut down your food budget. Maybe you can smuggle some home for other family members or for breakfast.

**Sandwiches by Daria Shevtsova from Pexels**

Or at least if there is some left-over, you can bring it back to your team. Or to people you like. Beware, there might be some coincidental overlap. They will be so grateful. Joke aside, this is rather a disadvantage. Based on my experience, these office junks are usually not so great quality, yet addictive and lead to obesity. Ask friends who used to work for free-food companies like Google.

Nevertheless, in the short term, it seems an advantage.

Apart from free food, what other advantages do internal events have?

## Understanding the core values

Let's continue with a real advantage!

By attending various internal events you'll have a better understanding of the company culture. They organize events around

topics that are valued, events where they emphasize the core values of the company.

If you pay attention, you'll have a more precise big picture not only of what the company does but what you have to do in case you want to climb the ladder. But this book is mostly not about that.

Having this better overview and understanding is not something extremely valuable in the short term, but the longer you think, the more essential it becomes.

I was working for a company where every single meeting started with a safety-related topic. Maybe we simply discussed how we could leave the fastest way in case of an emergency. Sometimes it was ridiculed by the choice of topic, such as how to use a chainsaw properly. For big-city office workers who barely see a real forest…

**Unsafe chainsaw usage by Ron Lach from Pexels**

Other times it was useful, what to pay attention to when you choose sunscreen, or when it is time to change your tires. Moreover, you didn't just gain *safety points* by delivering such presentations that you could later exchange for some gifts (dinner for two, safety test driving e.g.), but you could also mention anything safety-related in your yearly performance review. In fact, it was even expected.

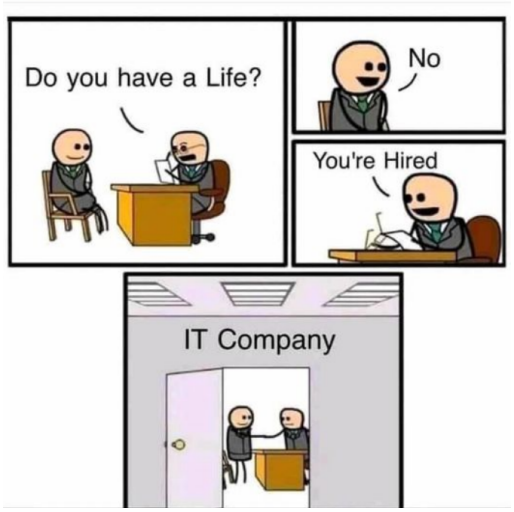
Attending internal events helped me to understand the importance of safety for the company and for myself.

Anywhere you go, internal events will help you understand the dynamics, the company values, what is valued and what is clear bullshit. Again, there might be some coincidental overlap.

## Showing how well you are integrated

A third - okay, second - and more short-term advantage is that you can share with your boss how well you integrated into the company culture. You shouldn't get carried away though, because you can be easily perceived as someone very lazy who does nothing but frequents internal events.

Share only the most important ones and pay attention to what you say about them. Act like a person who cares about the big picture, about the company. Speak as someone who found his big family and doesn't come to work just to get the job done. You will seem like somebody with a prospect. (And no life. But who cares.)




**You have no life? You're hired!**

Here I also have to mention that attending too many internal events can harm you. There was an intern called Richie in our team who was everywhere. Almost everywhere. Except for his own desk. He was supposed to sit next to me, but I barely saw him.

Instead of making something useful, Richie was chatting with anyone, went to all the meetings, hitting on all the women at the coffee room - including my then-fiancée, now my wife - without much success. His best question probably was why do you think this coffee machine has a door hiding the freshly brewed coffee that opens from the side. That could have been an interesting question, if you knew the answer. Nobody knew, including him.

Richie always managed to explain why it was important to attend the latest meeting about cutting-edge e-services or about rethinking how we think about thinking people in the company who spend their time thinking about tinkering, etc. This went on until once Richie actually spent more than two hours at his desk. Snoring. Loudly.

One of my colleagues, Kate, was nice enough to close the shutter

and make the office a better place to rest.

That was his last day at the company.

At least it was relaxing.

You have to do some work - at least in the beginning - it's not enough to simply "get integrated" into the corporate culture.

## Learning new skills

Last but definitely not least, you might even learn valuable skills by attending internal events. Most of these gatherings are useful mostly for the organizer. Even if there is no useful outcome, the organizer gets a chance to shine more at the upcoming performance review and he might get a slightly bigger bonus.

Yet we must admit, there are a few inherently useful events.

At my first corporate workplace, a bunch of enthusiastic people organized a local branch of Toastmasters International[3]. It was scheduled partly during lunchtime, partly during work hours and the company provided free lunch for the participants. More than enough food was more than enough extra motivation!

Most often it was just the usual - by the way, not so bad! - sandwiches, but sometimes even pizzas were delivered by the end of our meetup.

Unlike at other meetings, there was not a lot of nonsense going on there, even the participating managers left their corporate bullshit generator devices at their desks. Well, most of them.

At the same time, it was a particularly safe place to learn public speaking. I was, in fact, I am still quite antisocial and introverted. In addition, despite of all my prior political experience (not that much), I was afraid of public speaking. I had stage fright.

[3] https://www.toastmasters.org/

According to some studies, more people are afraid of speaking in front of an audience than death[4]. Astonishing, if you think about how death influences your life compared to saying a few *eh*s and *ah*s in front of a bunch of colleagues who don't really give a damn…

When I say that I had stage fright, it was not on this level. I didn't wet my pants, I was not shaking with the mic in my hand. Still, I didn't feel comfortable.

The first useful lesson I learned at Toastmasters is that I was not alone. A lady - a few years older than me who always seemed to be very firm and confident - could barely open her mouth in front of an audience. I was astonished - and relieved - to see her struggling. Her struggles made me understand that I was not alone and there we were to learn and improve ourselves. After just being a visitor for a few times, I joined the club and volunteered to make my introductory speech.

Watching every week the others dismantled a big internal barrier and I practised public speaking with growing enthusiasm. I never became a master of it, but I learned to enjoy speaking in front of an audience. Most importantly, I got rid of my stage fright. More than that, I got rid of most of my fears from new things. Most. Not all. Don't call me bungee jumping because I rather jump off my balcony. I live on the first floor, but still.

I built some connections which after all were not so important when I left not just the corporation but even my home country.

Nevertheless, Toastmasters clearly changed my life.

It helped me learn presenting, speaking in public, standing in front of other people. Without that, I could have never reached that level of seniority in my next job. I could have never delivered that many knowledge sharing sessions or spoken at international conferences.

All that happened due to an internal corporate event with free

[4] https://www.tandfonline.com/doi/abs/10.1080/08824096.2012.667772#:~:text=This%20study%20found%20that%20public,students%20selected%20death%20most%20often.

lunch.

This was not the last lunchtime activity of a corporation that I consider life-changing, but I will share more about the Software Craftsmanship Community in a later chapter.

# Work for a corporation!

I don't want to tell you that working for a corporation is the one and only right choice. I don't really want to make any such claim about anything. Things are seldom black and white.

If we simply think about what kind of company to work for, there is no general right choice. For some, a corporation may seem better, for others a startup. In fact, both can be the right choice, even though there will be people who really stick with one option. And there are so many other possible choices.

In the next pages, I'll share why I think working for a corporation can be a good thing. Apart from the numerous internal events and free food I already wrote about.

## Try out different positions without risking your job

Starting your career in a corporation has the great benefit of having a lot of opportunities. You can try so many things without changing jobs.

You can literally change positions without risking your job. Many companies even have policies making it mandatory for fresh grads or simply new hires to see multiple areas of the enterprise in the first N years.

While this might be a burden if you think you found your calling at your first team, it's quite unlikely that you are right. On the con-

trary, it will benefit your career and your personality by widening your horizons.

Changing positions too frequently has the potential danger of creating *expert beginners*[5], who every second year or even more frequently jump to a completely different project or position without gaining any real expertise. Yet these people, these IT workers tend to think they have superhuman or ninja capabilities. Usually, that lasts only as long as the interviews are internal.

These professionals whom I refer to as *expert beginners* after Erik Dietrich[6], they are people who consider themselves highly knowledgeable in a given or actually in multiple topics, whereas they are only slightly more advanced than an advanced beginner. They lack both the big picture of the projects they work on and a deeper understanding of the technologies they use. It's not a surprise, they don't spend enough time on one topic to become a real expert.

After my graduation, I spent a bit more than 3 years as an MS SQL DBA in a shared service centre. When I finally decided to leave the corporation, I thought I was good.

A good DBA, a good workforce.

For sure, I knew more than the majority of my team. On my last day, I had to explain to a colleague who considered herself a senior DBA how she could check the version of a given installation. At that point I was not just a bit smug, but also extremely frustrated.

Another claimed, he is not a programmer to understand a small script of SQL commands. They were more "experienced" than me, with at least 30% higher salaries due to their years of service. This highly contributed to my frustration.

Soon, I realized that what I knew was far from enough, I was struggling at my interviews. I passed through many rounds for two

[5] https://daedtech.com/how-developers-stop-learning-rise-of-the-expert-beginner/
[6] https://daedtech.com/about/

positions and I almost signed a contract with one of them, but they were far from being easy challenges.

I was assured that it was more my attitude that convinced them rather than the depth of my technical knowledge.

> This can actually be a nice lesson on the importance of soft skills and attitude… You can learn whatever technology if you have the right attitude.
>
> Changing your attitude is a tough nut to crack.

During the months of my job search, I realized that it's not enough to be good at a given job in a given company. You have to keep an eye on the wider industry, the latest technologies, articles, books, trends. And it would still not be enough.

You must practise, you must try most of the things you learn about - and I don't mean the latest Javascript framework, but rather more advanced features of your framework of "expertise". Otherwise, just reading about stuff doesn't mean a lot.

You know about the features, the possibilities, but you have no hands-on experience. However, the experience doesn't have to come from your company, it's fine if you learn and experiment on the side. But that's something I didn't consider at that time.

In my case, simply ensuring that the servers are up with the least amount of downtime, backups are taken as needed and that jobs and scripts are executed right on time took most of my time. Writing any script, designing tables, God forbid thinking about indexing was out of scope for us. If you wanted to learn such things you were on your own and you had no real environment to practice. These actions were done by application developers only.

In a smaller company, as a DBA, such "advanced items" are definitely your main concern.

I was a very qualified junior with a good attitude.

On the other hand, if you are willing to stay and you are not very unlucky, you will have your chance to switch to another - probably internal - position with relative ease and you will be a highly qualified beginner in another domain. A few different positions and you'll be an expert beginner.

Then you face different choices. Either you continue in a corporation, preferably where IT is the main concern. Then you try to find your preferred domain and try to be really good at it.

If you already found your field of interest then the other choice is to go to a small company to niche down even more. While it's possible to be an expert of a given small subdomain in a corporation, the chances that they will not let you focus on it for a long enough time is quite high. Still, a corporation is a great place to discover yourself and many fields.

## Don't worry about passion

As you can see, one of the biggest advantages of starting to work for a corporation is to find yourself, find your calling, your vocation. You can find the area you are the most interested in with the smallest risk possible.

You might argue that finding your interest is costly and you should just choose an area as fast as possible in the beginning and stick to it. You can argue that if you change fields, you might have to switch over to a less senior position for less money. If you measure life purely in dollars earned, then maybe that's right. Maybe it's the better option to stick to the one ladder you grabbed at the beginning. But what if you grabbed the wrong ladder?

Life is more than simply dollars earned.

Working with joy, going in on Monday mornings with sparkling enthusiasm are priceless feelings. Joy and passion might not come right away. In fact, most probably you won't feel any of those

kinds of things. I'm a big fan of Cal Newport's books and ideas. In his book So Good They Can't Ignore You[7] he claims that in the beginning, one has no passion. Hence, you shouldn't follow your passion, simply because you cannot follow something that you don't have. All one can do is grind, learn, understand deeper, find connections and then build up the passion slowly while you become so good that they won't be able to overlook you.

It doesn't mean that you shouldn't try a few different areas, you should. Find those that spark a bit more interest in you, but don't expect necessarily passion. Then pick one of these ideas, put in everything you have, work smart and hard at the same time and passion might come along the way.

Don't worry about time. You have enough.

Even if you spent long years studying and didn't join the job market before the age of 25, you'll spend at least 40 years working in case you don't die before retirement. Possibly even more.

That's a hell of a lot.

Maybe more than you lived so far. Definitely more than I have behind me.

In the past, it was highly probable that you'd do the same thing when you die or retire that you started to do at the beginning of your career. Which is probably the same as your father and grandfather did.

It's not the case anymore.

We switch jobs, domains, fields several times during our lifetime.

Before niching down into one specific area, it's beneficial to try and see different fields.

I took a part-time project coordinator job at the age of 24 while I was still a university student. I already knew more than a usual student

[7] https://amzn.to/39ElAoB

due to my experience in politics and campaign management, yet as a student, I was cheaper both in terms of salary and taxes.

Our field of expertise was helping local governments to restructure their organization with the help of IT, with so-called decision supporting systems. I had the possibility to be the bridge between developers, leaders of municipalities and financial professionals. I had the chance to visit a few different places in the country and met people who… Well, I met a notary "who met aliens", according to his stories. Indeed, those days on the border of two counties in a village of about 2000 inhabitants were extraterrestrial.

We laughed a lot with my boss driving back home. Maybe others somewhere above us also laughed at us.

No matter how many first contacts we missed with ET et al, I was appreciated a lot by my bosses, but as soon as I graduated I left. Probably not the nicest, not the wisest way, but I already understood that counteroffers should be rejected. They don't solve your main problems with a job. I wanted to use and improve my English and wanted a fixed salary with fixed hours. Safety and comfort at the age of 25…

I had to change. In many ways.

When I was leaving in order to become a Database Administrator, one of the owners, Attila, asked me

- *Sándor, when are you going to be a boss like that?*

I was very surprised and told him:
*Attila, I'm barely 25. Even if it will take 10 years for me to become a manager or something like that, I'd be only 35. Still, 30 years to work. At least.*

I don't think that was an answer he liked, but he understood and accepted it. Anyway, he didn't have many choices.

I didn't have the urge to become a manager. A boss. Nah. Not even today, I don't look for that.

Had I stayed there, I would have become better at coordinating projects, probably I would have become a senior project manager before burning out. I would have either say *"oh just fuck"* it and gone to the Caribbeans to become a bartender or I would have drunk anything I could, still continuing to climb the wrong ladder and I would have gotten positions I hated.

Instead, I joined a multinational corporation, I tried myself in something else, I experimented with database management and I started to pick up tiny bits of scripting, reading and writing code.

Then I finally became a developer in another country, in another corporation where I could try different technologies and while at the same time taking on side roles coordinating, teaching, leading technical initiatives and working to help my teams as a principal engineer.

Along the way, I became more and more enthusiastic. That passion that Cal Newport wrote about started to build up, just like my knowledge and confidence.

## Sharpen your axe and chop down the tree quickly

There is nothing wrong with working hard. Sometimes you'll have to bust your ass off. Yet it should not be your default way of working. Working smart is a better option. But how can we work smart?

If we want to work smart, we should actually become smart.

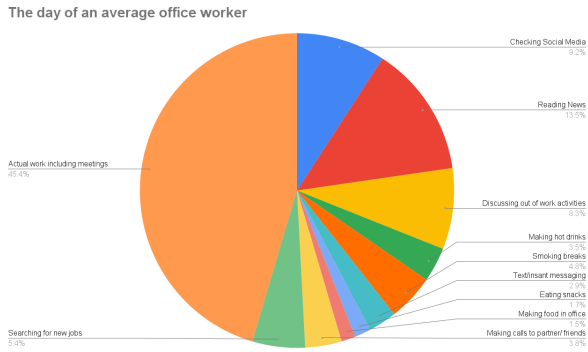
As Abraham Lincoln famously said[8], if he had six hours to chop down a tree, he'd spend the first four hours sharpening the axe before touching the tree first.

This mindset was key to transforming my workdays.

[8] https://benjaminhardy.com/want-to-become-the-best-at-what-you-do-read-this/

## How to find more time to learn?

The other crucial thought was a research made by a UK money-saving brand[9]. According to their findings, the average office worker doesn't actually spend more than 3 hours a day with productive work. In fact, a little bit less. 2 hours and 53 minutes.




**The day of an average office worker**

That seems to be a ridiculously low number!

How can it be true?

Think about it! How much time do you spend in the - virtual - coffee room with others discussing mundane non-work related issues? How much time do you spend on social media? Watching cat videos? And of course, there are the stupid meetings you have to limit - don't get me wrong, not all the meetings are stupid.

They reported exactly these kinds of activities. Spending time on social media, news sites, texting, drinking hot beverages, smoking, etc. take up so much time. And they haven't even mentioned yet all those time-waster meetings.

I was shocked. As a diligent introvert, who used to toil through political campaigns where no time is enough and there is no time

[9] https://www.vouchercloud.com/resources/office-worker-productivity

to lose, I immediately understood why I was always ready with my tasks on time and reported back that I have plenty of time to pick new ones.

My bosses were always surprised, just like me.

Before you might think I didn't limit my daily productive work to 10380 seconds, that's not true. I very rarely do unpaid overtime and in most cases, it's more about mutual flexibility - meaning that I either worked less the day before or I'd take some time back the following day.

On the other hand, I understood better where the time goes, what are the realistic expectations of bosses and what I could be capable of if I used my time well.

Even before the big realization, my managers used to praise my time management skills.

I doubled down and I tried to close out or at least limit most of the social media and news portals consumption on which I wasted so much time anyway.

I started to demand meeting agendas even from my superiors and in addition, I schedule learning time into my workdays. I blocked out half an hour every afternoon that I spent researching a topic closely related to my work.

I advise you to do the same, create a recurring event in your agenda, call it research and take that time to read up on a new language feature, a testing technique you wanted to try, an obscure part of your standard library, you name it.

Take time to get better. It will soon pay off with huge dividends.

Of course, there can be teams where people wouldn't be happy if they saw your screen full of non-directly-work related content (books, online compilers, etc), but you might be able to navigate yourself to a good desk or even to occupy a meeting room.

And if you are worried that you spend so much time away from work, think about others who take so much time to smoke.

What if you are one of them?

The answer is simple - but not easy. Stop smoking. Your body will thank you. Just like your bank account. You're welcome.

Spend that time getting better. Your career will thank you. Again, you're welcome.

I used the early mornings and a bit of the after-lunch food coma time to work on my personal growth. I'm pretty sure that some managers would find this outrageous. I wouldn't like to work with them. Anyways, I never shared with any of them how I manage my time down to this level. In the end, some would feel a big urge to micro-manage my time. Thanks, but no thanks.

## Let your knowledge compound

Yes, spending all that time on learning is selfish. But I've always spent it on topics that could help not only my personal but my internal projects as well. Those hours made me more efficient and proficient, often they were instruments and motivation for knowledge sharing and mentoring sessions. According to some of my formal or informal mentees, I turned their career around.

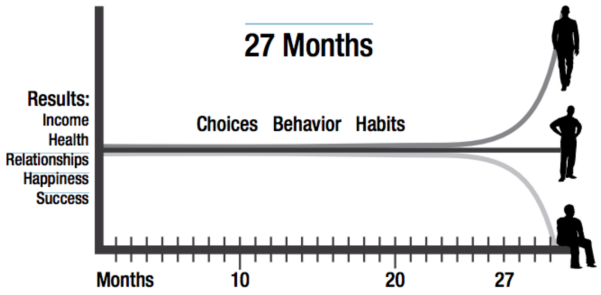
Not necessarily by giving them hard knowledge, I'm not an expert of any language after all, but my attitude.

This sounds very snobbish, I know. So let me add quickly. I didn't do anything extraordinary, I just cut down on time wasters and put in practice the advice of giants, like Mancuso, Uncle Bob, Feathers, Kent Beck, just to name a very few.

Learning a bit every single day compounds fast, that's the main idea behind Darren Hardy's book called The Compound Effect[10].

[10] https://www.sandordargo.com/blog/2019/04/17/the-compound-effect

You don't have to save the world on any day. You only have to make one small step forward and soon, with consistency, you'll be far ahead of the others.




**The compound effect**

I obviously wasn't the first one applying this in the field of software engineering. First I read about this concept in Coders at Work[11], where the father of Erlang, Joe Armstrong quoted Richard Hamming:

> "I always spend a day a week learning new stuff. That means I spend 20 per cent more of my time than my colleagues learning new stuff. Now 20 per cent at compound interest means that after four and a half years I will know twice as much as them. And because of compound interest, this 20 per cent extra, one day a week, after five years I will know three times as much".

Compound interests combined with learning is that powerful.

If others could do it, why not you?

With good scheduling, you can create a safe space for yourself to learn and by getting better, you'll accomplish your tasks not just with higher quality, but at a higher pace too.

[11] https://amzn.to/2TNkFNp

Think about it! You'll do the same work better and faster. You will end up having even more time.

Up to you how do you use it!

## Don't be a victim

More learning, more personal projects or more work tasks? Maybe a combination?

While there are hard deadlines even in corporations, mostly they are based on arbitrary decisions by people who have been promoted into a job they had often no competency to do.

Deadlines are barely life-and-death market dictates, but pretty much arbitrary dates.

I'm not saying you should not respect them. You should, because it has an effect on how you are seen, but don't be afraid and feel free to challenge them.

Heads - at least on individual contributor level - will rarely fall because of unmet deadlines and anyways, most of them will be renegotiated between the stakeholders.

You'll barely have to work your ass off, at least not because other people force you to. Exhausted, overworked people always have one thing in common.

The victim mindset.

We all know them. They like to be victims. They like to complain and explain how bad the world is with them. We are all prone to do such things in different areas of our life, but what is important to understand is that it's our choice.

The main question is whether we accept it if someone tells us to slow down and think differently. To stop acting like a victim.

Most people just take offence and they continue living their self-limiting life.

If you are different, you either lie even to yourself or you are part of the minority.

You are reading a book with the hope of becoming a better version of yourself, you're already part of a minority.

# What do you want from a job, that is the question

When you accept your first job, you probably don't have a vision of what you want. You know you have to work, and in exchange, you want to gain some experience that will get you a better job. Obviously you also want some money, because you want your own place, you want to be able to go out, to buy starters at the restaurant at your convenience. Or maybe you have to support your family, maybe you have some debts to reimburse. The common things is that in the beginning we don't really know what to expect from our jobs.

Do you want a low number of working hours? Maybe tons of money? Are you ready to work your butt off?

## Financial safety, no work and a high salary?

When we hire a handyman we have to choose two among the following three characteristics:

- quality
- speed
- price


**Quality Speed Price triangle**

We might find someone who is fast and cheap, but his work will have to be redone soon. Or if we don't want to go further away from our industry, think about the proof of concepts that must be delivered fast and cheap, but it includes a nice batch of technical debt.

The second choice is to opt for a good and cheap solution. You might find someone willing to deliver it, but you'll have to accept that either the work will take a long time, or more probably it will have such a low priority that you might have to wait until doom's day to have it finished.

If you want something of immaculate quality and want to have it fast, you'll have to pay. A lot.

In terms of a job, what could you wish for?

- A hefty salary, of course!
- A not daunting workload! Who wants to work their ass off? Generous vacation plan, sick and parental leaves, training plans…
- And by all means, the company should be financially safe! Salaries should be paid on time, just like bonuses and you shouldn't worry about being laid off, in particular for financial reasons.

Bad news, you can choose only two.

In fact, in some cases, only one is already difficult… Especially if you want the big money.

## Financial safety and high salaries?

Do you want the - relative - feeling of safety and big money? I've got some bad news for you, you'll have to work hard. Maybe even harder! After some years you might end up in the league of big players if that's really your goal, but that's another story and it will require all of you. That's right! Not *from you*, but *of you*.

You'll have to sacrifice your life, but once you're there you'll make such a big money that most only dream about and you probably lose your skin in the game. Meaning that even if you mess it up, the worse that can happen to you is that just go to another company to do something similar. Think about CEOs, football coaches, etc. They are barely sent to retirement, they just take on a similar job after they ruined their environment.

## Lots of money, lots of rest?

And if you want the big money and small workload?

Well, that's probably not possible. Truth be told, there are positions you might find later. If you give up your life and work a lot, somewhere in upper/middle management you might find some positions where you can delegate almost everything away and you make a lot of money, but I wouldn't bet on it. Besides, that's not something you get in the beginning and you have to earn a reputation to have such delicacies available to you.

## Financial safety with low workload?

You want security and you prefer a not so great workload? That's totally possible in a corporation, you can float around, without

anybody complaining seriously about your deliveries, but you'll not make an extraordinary living. Probably this is the easiest choice, followed by many.

Corporations are financially safer in most cases than smaller companies - more on that later - and due to some - bad - decisions above-average individual contributor performance barely comes with above-average remuneration.

Except for totally altruistic people, for people having a drive tied to their company or for people who are simply trying to avoid their homes, this means that the best thing that above-average people can do is to do their job as fast and as efficiently as possible and invest the rest of their time in something else. Of course, in incognito. For some reason, still, many managers think that people who can work more efficiently should be given more tasks without more money.

## A realistic combination

What is clearly possible is a relatively stable position with a decent but not great salary and a bearable workload.

But there are some glass ceilings that are hard to break even if you work hard. Not impossible, but difficult.

Getting a raise or a promotion is often easier to get by changing jobs than by just asking for them. Hitting the expert levels - if they exist in your organization - often seems extremely difficult. Many times what you'd need is more a political than a professional A-game.

You need a good network in the company and you have to stay for a long time, longer than is usual nowadays.

I saw people promoted to these expert positions mostly because they were there for a very long time. Not that they were not good, they were. Just maybe not **that** good, or not good for that particular role. But there was an opening and their management wanted to reward them for being quite good and loyal for more than a decade.

This has two messages.

First, time and timing matter.

Second, you need a job opening. In other words, you always need a business opportunity. You can be good, but if there is no place for you to be promoted, you won't be. Not because your management is evil, but simply because they cannot advance you without having the money and the place.

Don't be discouraged! We already discussed that most of us still have decades of work in front of us!

If you chose to work for a corporation, most probably you are already in the big league in a sense that systems are huge, transactions are flying with a rate of at least thousands per second, there is a real architecture, and even if it resembles a big bowl of spaghetti, at least, it serves a lot of customers.

If you later move on and join a startup, most probably you'll have a better understanding of how to scale their systems if ~~not~~ by chance it's one of the few succeeding unicorns. The other way around is not so evident, you can come from a small startup being the king of code, arrive at a huge enterprise and you won't understand anything, not just the huge and hopefully relatively robust system, but the processes in general - not to mention the political games.

I think moving from a process-oriented company to a less process-intensive is relatively easy, but the other way around can be intimidating. At least, that's what I've seen, but after all, we are humans. We can adapt to almost anything.

Yet, it's a huge difference that in a corporation you have processes for everything and anything, starting from how to create your branches to how to name your tests, what to put in your mail signature and how to get a new battery for your wireless keyboard. Don't worry, you'll get it in just a few days until then open your laptop or get a spare wired keyboard from somewhere.

And if you find something that has not been documented and

turned into a process you'll be more than welcome to fill the gap. Managers simply love folks who write processes. In my company, attributing to processes is one of the key responsibilities of a principal engineer.

Does that sound odd to you?

Maybe. But at least, if they are well written - which is rarely the case - you will have an easier time to ramp up.

In other words, to join a growing team or simply to replace someone.

## The safety net

In most corporations, nowadays, as a developer, you have relative freedom of organizing your workday unless you have a micro-manager. At the same time, you don't have to die because of overwork. This, combined with the stable financial possibilities of many behemoths, gives corporate employees a considerable safety net. A safety net that is of course relative, just think about any major setback of a company or God forbid the whole economy.

Laying off 10 per cent here, firing 25 per cent there, does it ring the bell for you?

Who will be the first one to get rid of?

Who will be the first ones to make some economies on?

The CEO's multi-million bonus?

Okay, not that close!

Maybe, the CEO's multi-million salary?

Not even closer!

No, it's always the employees. Your benefits will be wiped out, your salaries won't be raised or will be even cut or you'll be actually laid off if that's what serves the company's best interests.

As they say, we are not just a company, but a family after all. And what do you do with needy family members in hard times? You kick them out of your home!

I hope not.

I'm being kind of sarcastic here. If you buy that bullshit that your workplace is like a family, the shame is on you not on them.

Anyway, can we blame the companies because of the just described almost automatic mechanisms?

I don't think so.

Their main purpose is to pay interests and dividends to their shareholders not to support you emotionally or financially. They don't **have to** give work to anyone, they don't owe anyone, only their investors.

Actually, it's partly this behaviour that makes them long-living and quite resisting. That's why you still have a relative safety net and if you walk around with open eyes, you can start feeling when there is no more air around and when you should start looking for another job. The smaller the company, the faster these calamities escalate.

If corporations are in such a bad position that they have to let people go, many small companies have faced the same fate or worse and even earlier.

The safety net that is provided by a corporation is relative, but it's usually better than that of a startup.

## Conclusion

I still don't say that everyone should start working for a corporation, not at all. But it might fit you better, and now you have the reasons why. I think it's an ideal place to try different technologies, different fields, maybe even different roles while you can preserve a relative safety of income and future choices.

Before finding your niche, you can and you'll be asked to develop yourself into a generalist.

Are you afraid that you lose time as such? Don't be. As I told my former boss, Attila, you have a few decades between entering the job market and retiring - unless you really aim for financial independence and early retirement.

Use those first few years to discover many areas and opportunities that arise in a corporation. Pick up whatever skills you can, especially the soft ones. Most importantly, leave the junior mindset behind as early as humanly possible and start taking responsibility. Not for teams, but take responsibility for your code, your projects, your career.

## Challenges

1. Choosing the right path requires that you understand your own needs. What do you value the most at work? Would you prefer niching down into a technology? Or would you rather become a generalist? Do you enjoy non-technical tasks? In what quantity? Try to spend half an hour on your own, maybe with a notebook and answer these questions.
2. How would you spend extra 2-3 hours a day at work if you had more time? Would you socialize more? Would you *work* more? Would you learn more? Now think about how you waste your time during the way and try to free that time!
3. Where is your sweet spot in the salary-workload-safety triangle? Will you be willing to sacrifice your free time for more salary? Are you willing to take risks to get a higher pay? Or do you prefer something more safe with less workload? There is no right or wrong answer, only personal preferences and they can change over time.

# Chapter 2: Getting out of the junior mindset

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniioritytrap.

## It doesn't take years to gain some experience

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniioritytrap.

## What is a junior mindset?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniioritytrap.

### The lack of balance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniioritytrap.

### The urge to overcomplicate things

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniioritytrap.

### The sentiment of arrival

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Greenfield projects vs maintenance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Lack of responsibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## How to be a junior developer without the junior mindset?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Being a junior doesn't mean that you don't bring something unique

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Eradicate the victim mindset from all areas of your life

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Use the people around you as a motivational force

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Remind yourself of your worthiness and act as someone worthy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 3: Find yourself something you can be passionate about

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Miraculous transformations can happen to anyone

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Accelerate your growth with the right resources

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Read to learn about transforming ideas

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Step on the shoulders of others

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Let others step on your shoulders

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Passion will fuel your growth. But how do you find it?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Grow through deliberate practice

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Believe in yourself

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Why you should find your niche

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Passion leads to specialization

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## The niche that you're passionate about will make you grow

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 4: Handle failures and obstacles so that they help you

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## It's always your fault

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Do you still blame others?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Take ownership of your situation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Lead and fix the problem

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# The dangers of taking all the blame

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## People won't blame but follow you

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## You can only be responsible for things you have control on

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# How to turn failures into growth in 5 steps?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## 1) Acknowledge you are flawed

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### 2) Understand that you need someone to help you



### 3) Look for someone who can help you



### 4) Ask for help



### 5) Implement the ideas you received



## Tackling obstacles by asking questions and taking responsibility

## When to act, when to let it go

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 5: Ask or you won't get it

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## They will take advantage of you

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Successful people take their fair shares

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## But why don't people ask?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Pride leads to worse outcomes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Don't miss out on uncertain gains

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## You have to ask for it

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### There is nothing to be afraid of

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### They will like you more

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Someone's trash might be the other's treasure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Ask to remove the burden of decision making

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Ask so next time it'll be more difficult to say no

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 6: Stand up for yourself and your values

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Do you hold values?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## They don’t have to like you

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Be a leader _____ an individual contributor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Don’t burn all the bridges

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Respect your values

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Is it worth having conflicts?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Never be a victim

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 7: The Seniority Trap

Have you ever seen a senior developer who could probably code circles around you who at the same time is actually barely coding?

Maybe you have been there, maybe you are right there right now or maybe you'll get into a similar situation soon. Maybe you don't work for the type of company where this could ever happen. Yet, most of us at least heard about highly knowledgeable, so-called senior developers who spend most of their time on non-coding activities. Let's put it in another way, they go to the office and they don't do what they are best in.

I don't say it's inevitable, but there is a fair chance that it will happen.

Some bosses could ironically say oh, you haven't coded in 3 weeks, 2 days and 6 hours, Mr. Senior Developer? What a pity! You have more important things to do than being so geeky! By the way, have you already sent those TPS reports?

You can reach this point in different ways.

- You were explicitly looking for an opportunity to code less
- You were told that this is the way to get promoted
- It just happened to you

Let's have a look at those paths.

## You wanted to code less

Maybe you're attracted to non-programming activities because you find coding boring as you never learned it well. You think that you're not born for this - none of us was - and just want to get out of it without risking your existence. Or perhaps you learned to code too well and you wanted to get a break. Your preconception can be false, but nobody should judge you, it's your choice. It's all fine, you wanted it.

I've seen some funny cases similar to the one I just described.

Why only similar?

It can happen that someone thinks he's a great developer, he is delivering lots of code and fixes, he has great velocity, but in reality, he is a subpar coder.

But there is nobody around to tell him. Maybe the others are even worse or just inexperienced, the manager forgot how to code a long time ago, so there is nobody to hold a mirror to this person.

The situation can become worse if this person becomes a project manager or a line manager. He will consider himself as a former great developer and oh, God, that's dangerous. If you are such a person, you will have false expectations regarding good engineers in terms of what it takes to deliver quality.

A few years ago, I was working on a bug in a relatively new component. It was the case of a complete rewrite of a mainframe application into something more modern. It didn't go so well as management only hired inexperienced people instead of making sure there is the desired healthy mixture of seniors and juniors. In addition, the project was always late and management just pushed, pushed, pushed for the delivery. People worked overtime, more - inexperienced - engineers were brought in and the project got delayed even more while it accumulated some enormous technical debt.

The managers there clearly never read the Mythical Man Month[12]. I'd assume that they never read any book on management. That's a kind of a danger with developers promoted into management ranks for their good work. They never learnt to be a manager, probably they didn't even want to become one, but that was the only way to get a better salary. Anyway, in the Mythical Man Month it was quite clearly explained that if you bring in more people to a project already late, the situation will be just worse. New people will require a decent amount of ramp-up which will take quite some time away from those who are already familiar with the project.

Needless to say, the project was not a success and basically, the whole department changed (left) in a very short period of time.

A tiny part of their legacy was a 653 lines long function consisting of complicated, sometimes contradicting, sometimes completely duplicated `if` statements.

What about the specs? - you might ask.

There were some, we managed to find them. Sadly, but not surprisingly, they were not aligned with the code. Or better to say the code at many places had little in common with the specs.

Somebody already found a bug a couple of months before, but he didn't want to change this monster. Maybe he didn't have the time or the will or he was told not to.

Who knows?

So what did he do?

He wrapped the function call into another pile of `if`s. Not surprisingly they were also buggy. Soon he got fed up and left the company for a researcher position at a university.

So what could I do?

I wrapped the wrapped function into another layer of `if`s.

[12] https://amzn.to/3zOBDvR

No. Of course, not. I would have hung myself before doing that.

I went down to the dungeon, extracted what I could into different classes and into a lot of small and - at least I hope - well-named functions. I unit tested everything and I also documented all the deviations from the specs.

When you see that the specs differ from what is implemented you have two choices.

You can fix the code and as such risking that you change behaviour that users rely on. It can be dangerous. Hyrum's law indicates that…

> *… with a sufficient number of users of an API, it does not matter what you promise in the contract: all observable behaviors of your system will be depended on by somebody.*

Changing the API or the behaviour of any existing system might lead to hoards of angry clients. That's a dilemma exposed also by the Gilded rose kata[13]. You should not just change the API or the behaviours unless you have a very strong reason to do so.

The safer option is that you separate the code and document it - preferably through unit tests. I aimed for testing the logic 100% and whichever part was not in line with the specs, I made a comment about it and referenced the relevant section of the specifications.

This takes time.

A few days later after I understood where the problem was, the manager overseeing maintenance activities came to my desk and gently asked about the status - even though I already explained it during the daily 15 minutes of shame. Never mind. I explained once more - now more in detail - and keenly showed him what I found. George told me the product manager was complaining about the

[13] https://github.com/emilybache/GildedRose-Refactoring-Kata

time it takes me to fix the bug, she - the product manager - used to be a developer too and she was sure it couldn't take that much.

I stopped. I turned in my chair, facing George. I accurately adjusted my glasses and I looked at his eyes and said:

*Maybe that's why she's not a developer anymore.*

George tried his best to hide his smile and told me that he got the point. He wouldn't transmit what I said, he would just take care of her.

She is the perfect example of someone who thinks she used to be a great dev but actually wasn't. Ok, it could have been that she was actually a seasoned engineer and wrote maintainable and reliable code rapidly.

The problem is Git. Git never forgets and never lies.

Just let me make it clear. It's not a problem that she wrote somewhat questionable code. Not everyone is a great developer and not every developer could be a great project manager. The problem is that she claimed that she'd know how much it takes to fix a problem because she used to be a developer and questioned the work of actual developers based on her past role.

The whole team had similar problems with her. She was always looking for fast solutions, letting our codebase down. Letting quality down. Letting our company and eventually our clients down.

Some of our developers simply didn't care anymore and the rest were too young and shy to push back. Luckily she changed positions. Our team's path and hers parted in different ways.

You might become a non-coding developer because you want to code less. You want to deal more with project coordination, resource management, maybe with educating people.

If that's the case, you have to consider that your coding skills will deteriorate, they will become a bit rusty. Not that you'd forget

coding completely, but you'll not be the reference person anymore, even if you were.

## You were told that this is the way to get promoted

The second possibility why you might have ended up in the seniority trap of not coding anymore so much is quite simple and straightforward. You were explicitly told that it is the way to get promoted to a senior developer and to be paid better.

Anyone who ever heard C-level leaders or HR prominents speaking about employee engagement know their mantra by heart:

> People are motivated by fancy swags, bean bags, interesting technologies, and iMacs. Not by their salary.

What a peck of bullshit.

People do work for money and most of us will eventually feel bad if we are underpaid. Most employers are shortsighted and prefer to pay people just enough that they don't leave immediately. I say it's shortsighted to do so because not everyone will ask for more money when they are fed up. Small things add on the top of being underpaid and it's impossible to save the relationship anymore.

At the same time, it's easy to forget that replacing someone is expensive. Its cost grows with the seniority of the person to be replaced and when the knowledge is gone, it's gone.

To build a lasting company, to build a good culture where knowledge is valued, it'd be better to simply pay a little bit higher than the bare minimum. To pay enough that people don't care about their salary.

Anyway, I digressed.

Those who don't just jump on another job with the promise of a higher salary, they will ask at a certain point what they have to do for more money.

Hopefully, you'll not be asked to do anything inappropriate or illegal.

After thinking a bit more about it, maybe those wouldn't be the worst options.

You will be told that you have to coordinate this project, lead the activities on that stream and you might get recognized. In other words, you have to do the job of a (project) manager without getting paid like one. But you have a not so strong promise that one day, if you do it long and well enough, and if the constellation of the stars is right, and if you satisfy all the gods of all religions, then you might get a bit more than now.

Let's be fair. You'll learn important soft skills along the way and you can experience a bit how life would be on the dark side.

Reporting is not bad by definition. Reporting is useful and if you learn to do it well, it will benefit your career.

Speaking in front of others is an extremely useful skill. It can make or break a career, even for developers. If you ever slept during a training about Linux kernels that was supposed to be great, you know this. If you ever enjoyed the talks of distinguished developers at a renowned conference, you also know this - and you are luckier than the ones sleeping at trainings.

The activities that you'll be asked to do will not teach you how to report in an effective way. They will not make you a skilful presenter. But they will give you plenty of opportunities to practice. To get better.

Can't you learn these in another way?

Do you really have to become a part-time project manager, a full-time scrum master or a not-so-tech lead to learn these things?

No, there are definitely other ways to get these skills and there are slightly different ways to advance. You have to actively seek them out, they won't just appear like the conventional and still the one-and-only way to get ahead in many companies. We'll see them later.

The good thing about this second option is that it's not a surprise. It's exactly what is promised. You are given the two pills and it's up to you to decide.

My friend, Andrey - who I introduced to you in a previous chapter - was already a good developer with quite deep C++ knowledge. He wrote probably the finest code in the team and he was teaching others - definitely me - how to code better in C++. Yet, he never got promoted or got an above-average raise until he became a scrum master and coded half less than before.

Instead, he started to organize ceremonies, inquired people at daily standups, filled spreadsheets with ~~assumption~~ progress reports and updated JIRA boards. And of course, he liked his job more. Ah no, sorry, much less. Then he finally received a double raise, around 5% - not a promotion yet.

It was quite a sobering revelation for him.

He was not valued as an engineer, a mentor, but as a project manager. Yet, he went back to follow his passions as a developer. Then it took him years to officially become a senior developer.

## It just happened to you

In the last case, it just happened to you that you got trapped. You didn't seek it and you were not explicitly proposed to any path. Probably now you raised the eyebrows. How can such a thing *just happen to you*?

I agree, it's a bad sign. It's bad when you are only a passenger in the car representing your career.

Yet, it can easily happen. Career planning, how to grow in the ranks of a corporation is not something you learn at school or something you get trained on at work. We all have to figure it out for ourselves.

The first years pass by quickly. Most of us, we have a whole adult life to set up. We have our life to put on track, we try to create our work-life balance, move to our own flat, often rebuild our social circles, and start a family.

Conscious career planning is the very last thing we want to think about. We go to work, we do whatever is required from us and probably even a bit more. We want to grow, we want to prove to our boss and our colleagues that we are worthy. And you just accidentally walk into the seniority trap without noticing it before it's too late.

If you are a young and keen software engineer with a bit of a character and not very impatient regarding your promotions and raises, your job will change little by little until you are caught in the trap.

You will learn to code better and better, you will probably have ideas on how to be more efficient and how others should be more productive too.

Management will be happy at first, but soon you will realize that they are not that much interested in doing anything in order to help you implement those ideas. Unless you managed to sell your idea as theirs. But probably you are not so canny yet at this point of your career.

Therefore if you really want a better life, you will have to take the lead.

Once you did that and you didn't completely mess up everything and something was eventually delivered in a reasonable time, the next similar tasks will find you fast.

You will be asked to coordinate this activity, make a presentation there, be the ear on that other meeting, etc.

Soon you will realize that you haven't coded in a while because you are only coordinating projects and doing the reporting for your boss, who thinks he is great as he delegated so much of his tasks to one of his unsuspecting subordinates.

The end result is pretty much the same in all three cases. You'll end up as a senior developer who barely codes. You finish almost as uncle Bob's famous non-coding architect, just with a worse salary and without any worthy title.

## The way out

A colleague of mine who was already a senior developer and deeply interested in systems design was proposed the role of an architect. He gladly took the opportunity.

Soon he started to feel some disturbance in the force.

He lost touch with the codebase. Pretty soon it became evident to him that the position was something completely different from what he expected. No coding, no mentoring, not even real architectural decisions as those were done by those who were coding. He spent most of his time in meetings and writing documentation.

The way out? He went back to a developer position as soon as it was possible for him and he made sure to be in control of what he'd work on.

So what is the solution?

It depends on what you want.

But for that, **you must know what you really want**.

### Look for more responsibility

If what motivates you is leaving programming behind while staying in IT, don't accept being trapped in a non-coding senior developer

position. Just move forward and do the necessary to become an engineering manager, maybe a product owner, or something alike. There is nothing wrong with that. Just don't be the developer who barely codes, and when he codes he does it in a degrading quality and doesn't make as much money as he could as a real manager.

If you know what you want, better to be single-minded and start climbing the right ladder. Let your management know about what you want, and make sure that you are aiming in a good direction.

You will not become a manager in an instant, but at least it will be clear for everyone what you shoot for and your management can help you make it happen.

If they don't know what you want, they cannot help.

But if they know, they will probably assign you some side activities that will help you grow in the desired direction. You might manage some technical activities, small projects or you might become a scrum master so that you can showcase the necessary skills.

Once there will be an opening for your craved position, you'll be ready to take it.

## Automate the hell out of it

What if you like programming and you don't want to end up in a non-coding developer position?

If you are in a corporation where a developer's advancement is not tied to coding but to other activities… well… it's not easy, but you must stay focused.

Don't forget that you're a developer and you want to become a better one.

Do the needful to keep your manager happy, but try to look for side activities that help you become technically better.

Don't get me wrong, soft skills are important, that's something we already discussed. Besides, it's also useful to know how to juggle with spreadsheets, even if it's not your main goal.

In my case, helpful side-activities were things like organizing coding-dojos, knowledge sharing sessions and helping ramp up our in-house technical mentoring program.

Let that sink in for a moment.

When I organized coding dojos, of course, I took on some organizational activities, but I had to review coding exercises and assess how they would help us grow as developers. I also had to facilitate the meeting where we were solving these exercises which made me grow as a developer.

A clear win-win.

While you might say that knowledge sharing sessions are about presenting, but in reality, you have to invest much more time in preparing the presentation and eventually extend your knowledge. Teaching is the best way to learn and knowledge sharing presentations are obviously about teaching.

You have to think twice about what you think you understand. Often you realize that you lack a clear understanding of some basics. Other times you have to explore new areas. Either way, you will learn things that you wouldn't have learned without preparing for a presentation.

While preparing, you might even ask for some help from more knowledgeable people and build some connections.

Sharing your knowledge is an awesome way to get better and get recognized.

But not the only one! If certain activities you would be interested in are not ongoing yet, you have to check if they would be welcome in your organization. If so, go for them. Create them! If you'll be the one who will start those activities in your department or in the whole corporation, the possible rewards are even higher.

Keep one thing in mind. Whatever you do, you have to communicate your priorities to your boss.

You have to say no. Frequently.

That will be your most important skill. To refuse opportunities gently, but firmly. Sometimes not so gently. Sometimes not opportunities, but tasks.

Focus on your essentials, focus on becoming a better developer while you eventually help others get better.

If your priority is, for example, to keep C++ coding in at least half of your time, don't accept a project with an undefined length where you'd be writing Ansible scripts, even if you would find it interesting.

## Leave and specialize

If you really cannot bear with this, I understand. Maybe you have to specialize in a niche and work as a consultant, or simply work for smaller companies where the corporate ladders are not created yet and where developers can focus more on development.

At least that's what you think.

Don't forget that in smaller companies, often you have to do many things that are done by separate departments in a huge corporation. Like maintaining your servers, or the cloud, or a VM within another VM. Whatever.

Again, it's not a problem if you have to do those things. But you have to understand your needs and tasks and find a way to keep them balanced.

You can also reject all these activities leading to fancier titles and higher salaries and focus on becoming a better programmer. You won't be a star in the eyes of your management and you won't get hefty raises, but by the time you might become a brilliant developer.

Is it worth it?

It is, but maybe sooner or later you'll have to change jobs. While we don't work only for money, we don't like to be underpaid either, right?

Besides, you might work in a corporation where you cannot use your coding skills up to their full extent and you need another type of company.

What you have to keep in mind all the time is that you are the only one responsible for your career. Even if your managers act as they know better what you need… Well sometimes they just keep repeating their own mantras, but they chose another path. You chose yours.

Be proactive. Communicate what your needs are, share what you enjoy doing and what you don't. If your opinion, your wishes are always ignored, draw the consequences and pack your box.

You don't want to get into the position of a non-coding architect!

## Conclusion

There are essentially three different ways to end up in a senior developer position where you barely code. I call this the seniority trap.

What to do if you are in a trap? Think about what you want. Then go for changing your official position if you want a non-coding role. Keep working on it to get the promotion or just simply tell your management what you like and what you don't like, what are your goals and dreams.

After all, we speak about your goals and time, not about those of your boss. You are the one responsible for your career, not anyone else. You will have to accept some trade-offs, that's fine. But things should not just happen to you. You should understand what's going on, what is under your control and what is not.

Do everything that you can, share your thoughts and needs, work hard and accept whatever you cannot change.

I'm convinced that you have many options to keep your coding role a coding role. But what if you are offered a non-coding role?

## Challenges

1. What do you want from your career? Do you want to stay on the technical track or you rather go for managerial? Maybe a mixture? Think about it a bit.
2. Are you heading in that direction right now? Are you happy with the direction you're going towards? What activities should you drop and what else should you pick up to correct your career trajectory?
3. Schedule a meeting with your manager and share all that you worked out about your career.

# Chapter 8: Non-coding roles or something you like?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Should you accept it or not?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Create your own portfolio job

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### What is a portfolio job?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### How to create a portfolio

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### What about the money?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Don’t become an energy vampire, it is never that bad

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Be always ready to change

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## You have the time to do what you enjoy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### When will you become a boss?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Keep your end goal in mind

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 9: Transforming individual ideas into challenges

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Are you willing to give up some of your coding time?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Would you code less so that you can work in a better environment?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Are you willing to take the lead?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Impactful changes need more than one cowboy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Three ways to become a multiplier

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Make the others' dreams come true

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Useful changes can attract other contributors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Share and delegate as an individual contributor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 10: Limit your focus to a couple of things

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Splitting your focus is the way to fail

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Start becoming a finisher

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Become a finisher for the reward!

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Whatever you think, your capacity is limited

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Spread your energies and you stop advancing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Set your goals, make a plan and follow-through

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Correct your course along the way

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Your goals must make sense

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Chapter 11: Keep being hungry

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Success is the best predictor of failure.

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Success brings more stress

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Never be the former someone

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

### Don’t go for the status

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Go for progress

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Know where you are going!

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Take time to celebrate

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## There are no complex goals, just unrefined plans

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Make yourself accountable

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Share your goals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## How to set good goals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

## Your confidence will grow

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Keep being hungry

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/theseniorityтrap.

# Conclusion

Thanks for having read *The Seniority Trap*, I hope you enjoyed it as much I enjoyed writing it.

In these eleven chapters, we discussed different challenges, events, patterns, situations that you will encounter during your career while you're working towards your goals.

The items in this book are inspired both by my career and other people's whom I look up to. While nobody's career can serve as a concrete recipe for success, you can and should take ideas from other people's life and implement them in your career.

In any case, don't forget to define first what success means to you. As for one of my former bosses, success at work was clearly about managing others, for me and maybe for you it has nothing to do with having direct reports, but more about making an impact on how things are being done.

In this book, you read about many helpful ideas to work on and pitfalls to avoid while you work towards your goals and become more and more senior. It would be hard to pick three items to give you as key takeaways, still, let me try to select what I consider the most important ingredients to succeed in your career.

As the Stoic philosopher, Seneca wrote *if one does not know to which port one is sailing, no wind is favourable.* Similarly, if you don't know what you expect from your career, if you don't know where you want to end up, it's very unlikely that you will feel successful.

You have to define your goals, you have to create your own definition of success. If you forget to do that, you cannot take steps to get there.

According to psychologists, one of the main reasons behind people not succeeding is simply that they don't know what they want.

The second ingredient I want to mention here is that you must learn to ask. You should assume that nobody will ever just give you anything valuable unless you take it, or at least you ask for it. Of course, you will - hopefully - find some counterexamples throughout your career, but those exceptions just strengthen the rule.

If you want something, a promotion, a raise, a trip to the conference, you have to ask for it. It doesn't mean that you'll get it for free, that you'll get it right away, but you'll already be at a great advantage compared to the others.

You expressed what you wanted.

It's the premise of setting out a plan with your boss to get there. And sometimes no plan is necessary. When you ask for it, you'll get it. Maybe it doesn't cost anything to the person who you asked. He simply couldn't read your mind and didn't think about your wishes.

Let's also not forget about the *Ben Franklin effect*; if you ask for something the other person will find you more sympathetic.

You have no reason not to ask for what you want.

I've been pondering what I should mention as the last item. Probably limiting your focus is more important in regards to your sheer achievements in your career, but there is something more important than what you achieve.

It's that you can look at the mirror without disgust.

It's that you can stand up straight with your shoulders back.

Maybe this should have even been the first key takeaway. In any case, if you see that something is not right, if you are treated unfairly, if you see that your colleagues are treated unfairly, stand up for yourself, stand up for your team.

You don't have to carry the cross of the whole world, that's clearly what not I mean. But when you face unrighteousness in your direct proximity, you'd better face it. Even if it will bring hardship upon you, even if it will be unpleasant.

If you don't follow this piece of advice, you will soon be resentful, stressed, negative. It will impact not only your progress at work but your human relationships too.

If you take away one phrase from this book, it should be this:

> **Set your goals, ask for what you want and stand up for yourself!**

P.S.: Please do reach out to me to share how the book helped you!