Top Developer
ACADEMY

THE 5 PROVEN STEPS
TO BECOMING A
SOFTWARE ARCHITECT & TECHNOLOGY LEADER

Why become a software architect?

Maybe you just landed your first job. Or you've already been working as a software engineer for a few years. But you don't want to blend with the mass as just another member of the technical staff who does their job well, but can easily be replaced with another engineer from inside the team or an external hire.

Instead, you want to be the most indispensable, respected, and valuable engineer in your team or organization - the one whose advice and expertise is requested when important technical decisions have to be made. Without it, your managers and peers do not feel confident making big architectural changes in the system or codebase. But for you, this is just another day in the office. As the Software Architect and Technology Leader, you have the knowledge, confidence, and authority to guide your company to success. You know the system better than anyone since likely you are the one who designed and architected most of it. And your deep technical knowledge puts you in the position to make a big impact on the company's revenue.

If this is who you want to be, then keep reading. In this 5 step process, I will guide you on how you can become a Software Architect and Technology Leader for your team.

Who is a Software Architect?

A software architect is not a degree that you graduate with, nor a certificate. In most organizations, it's not even a position or title.

It is a **role**. A role filled by the most trusted engineer in the team. Trust is key because it has nothing to do with the number of years at your job or company, formal education, or title. In most cases, a software architect is an individual contributor who is trusted to make the right technical design decisions while:

- Building a new system component
- Developing a new feature for a client
- Researching, suggesting, and choosing the right technologies, programming languages, or tools
- Optimizing performance to save the company money
- Ensuring day to day stability, scalability, and fault tolerance for the entire system

In that sense, a Software Architect and Technology Leader are the same things. A Software Architect cannot make design decisions without having the right technical skills and without being well versed with the right technologies that will be used to execute the design.

The 5 Step Process

So finally, the part that you are waiting for: the 5 proactive steps that will make you a software architect and technology leader as well as an indispensable, valuable, and highly sought after software engineer.

Step 1: Learn the System

First, understand the existing system and look for opportunities to improve it. Study it without judgment but with scrutiny. Don't jump to conclusions before you understand why something is built a certain way. But also don't assume that what already exists is: Set in stone, correct, or designed by a super human engineer who had all the knowledge and time in the world to make the best design choices. What's more, some decisions might have been correct at the time, but as the business evolved, are not so anymore. Those are the places to take note of as low hanging fruit where you can show your leadership by improving upon them.

The points to focus on are:

- **Follow the data** - Your starting point should be to understand how your clients are using and interacting with the company's product. This will point you to

where the data is coming from, how it flows through the entire system, and where and how it is stored.
- **Identify the technologies involved**, including the frameworks used, databases, data delivery technologies between services, and most importantly, the threading model and configurations in performance-critical services. It's ok if you are not familiar with most of the technical stack. This will be addressed in Step 3.

Step 2: Identify the business trend and technical pain-points

The most important aspect of making a meaningful impact is focusing on the most mission-critical parts of the system. Optimizing and re-architecting a subcomponent that is going away in a year or has no big importance to the business would be a waste of your time, and improving a system that works just fine is also not going to provide a lot of benefits.

So your goal in this step is two-fold:
- Understand the most mission-critical, high traffic, revenue/customer experience impacting parts of the system.
- Study which part of the system has the most critical pain-points, the most outages and bugs in the last

year, performance or scalability issues, or is just hard to test, maintain, deploy or add features to.

The systems that have the highest importance and the most critical issues are the ones to focus on first.

Step 3: Become an expert in the problem domain

After you know which system you are going to focus on re-architecting and the problems you are going to solve, the next step is to understand the roots of those problems and become the expert in their domain. At this stage, the worst thing you can do is be impatient and suggest a solution before fully studying the problem at hand or having the expertise to address them.

Instead, zone into the problematic component and study it in depth. While most of your colleagues are stuck in the mindset of focusing only on the tasks they are asked to work on, you are thinking big picture and long-term and are going to significantly up-level your skills related to the problem at hand.

In the process of diving deep, you might either 1) understand the problem but require expertise to solve it,

or 2) you need the expertise to understand the problem.

Such skills include but are not limited to: multithreading and concurrency, fundamentals of building highly scalable distributed systems, advanced features and capabilities of your programming language, related cloud or open source technologies, and the implementation details of the framework or technology you are investigating.

Taking a **credible** online course, attending training, or picking up a book on the topic are excellent investments as they will save you a great deal of time. When choosing your sources, make sure to filter out the professional entertainers, bloggers, and teachers and learn only from professional engineers who have successfully done what they now teach.

By the end of this step, you don't need to have a solution for the problem you are investigating but you do need to feel confident that you understand it well and have the expertise to try and solve it.

Step 4: Research a solution or an improvement to current architecture

Now we look for an architectural or technological change to improve the existing implementation. This step

requires research and prototyping. With expertise in the problem domain, we can confidently explore similar technologies and understand where they can address the problem better. If such technologies don't exist, that likely means an architectural change is required. Then you will need to think about how to change the system to avoid running into the scalability, performance, or other types of problems that you're dealing with.

This is the ultimate step in being a real software architect and technology leader. Researching and suggesting alternative technologies and architectural changes is both science and art so don't feel frustrated if the solution doesn't come to you right away.

If you are lucky, you might find another team or company that ran into similar issues and has presented their solution as part of a conference or software architecture magazine. If that is not the case, you would identify the potential pool of solutions and try them to see what yields the best results. For each solution that you try, capture enough data so your final choice is based on a quantifiable metric and not solely on a gut feeling.

Step 5: Create a design proposal with clear goals and benefits

Once you have gathered enough evidence that your proposal will improve the system, it is time to formally document the improvement and present it to the rest of the team or organization. The format for the presentation is called a **design document** or **design proposal**. The exact format of this design proposal is out of scope for this article, but the important things you need to present in such a document are:

- The specific technologies you're proposing to utilize
- What alternative solutions you have tried and considered
- What the system will look like and behave after the proposed change
- What your design does not address by clearly defining the scope of your solution and its benefits

If everything has gone right throughout this five-step process, you should end up with an approved design doc, at which point you might be the one implementing it or delegating to other engineers.

By continuing to repeat this process (Steps 2-5), you will establish yourself as a software architect and technology leader engineer for your organization.

How to start now...

Becoming a software architect is the ultimate goal for any software developer. It is not only impactful but exciting and fun since you solve the most challenging problems and it is literally your job duty to keep learning, researching, and growing your skills.

By following these 5 steps, as well as by investing in your technical education, you will become an expert that any company would want to retain and hire. If you choose to, you can apply your skillset to your projects or even start your own company.

Top Developer Academy's online courses are designed specifically for engineers on the path of becoming a software architect or technology leader. By bringing years of practical experience and in-depth theory into every lesson, we help you build the core fundamental skills and ways of thinking required of a software architect. Your time as an engineer is valuable so the courses are made to be both concise and effective.

Browse our courses here and join thousands of engineers who started on their paths to success!

- Michael