- The -

The Technical Coach

2nd Edition

- The - Technical Coach

BUILD SOFTWARE CHEAPLY & EASILY

BY LEARNING HOW TO EFFECTIVELY MANAGE SOFTWARE DEVELOPERS, CTOs, AGENCIES, & ADVISORS

<u>A PRACTICAL GUIDE</u>

FOR STARTUP FOUNDERS, ESTABLISHED CEOS, & INVESTORS

—

Jay Crouch

CONTENTS

ABOUT THIS GUIDE

This guide helps non-technical leaders select the best technology partners for their needs, and effectively manage the technologists that they hire.

It also helps both technical and non-technical leaders effectively organize the technologists that they lead.

You will find this guide helpful if...

- *You are a **non-technical startup founder***
- *You are a technical founder (or CTO)*
- *You are an established CEO, executive manager, or investor*
- *You are currently building software*
- *You are preparing to build software*
- *You intend to build a highly technical organization*
- *Or, you just want to bolt a little technology onto an otherwise non-technical operation*

PART I – ADVISORS

Explains everything that there is to know about Technical Advisors; e.g. why they are important, how to recruit them, and how to work with them.

PART II - EXECUTORS

Explains the pros and cons of working with software developers, technical co-founders, CTOs, and agencies; and illustrates the most important misconceptions to avoid when working with each.

PART III – GROWTH

Ties everything together while walking through the process of scaling a very small startup into a very large enterprise.

ABOUT THE AUTHOR

Jay Crouch

15 year veteran CTO/CPO, Technical Advisor, and Team Builder.

I help people avoid critical mistakes while building software.

Checkout my core philosophy, blog, and services at JayCrouch.com

- *Helped build a $2,000,000,000 unicorn (The Honest Company)*
- *Co-founded four companies (each of which raised millions)*
- *Advised dozens of organizations (all sizes)*
- *Led large, cross-functional teams (45+)*
- *Forged sales relationships with major brands (Marriott, Hilton, etc.)*
- *Helped author a formal assessment framework (Technology Credit Report)*
- *Published open source software (Active Record)*

PREFACE

Would you believe me if I told you that nearly 2/3^{rds} of all CEOs fire their most senior technology partners – <u>for cause</u>?

Well… it just so happens that I surveyed hundreds of startup founders, small business owners, and captains of industry; and this is exactly what I found.

—

> **<u>66%</u>** of all respondents indicated that they had been forced to replace their CTO, agency, or most senior engineer *(at least one time).*
>
> And, **<u>85%</u>** of these respondents indicated that they had been forced to start over *(or deal with a similarly devastating result).*

—

This means that most business leaders are, at this very moment in time, investing in the **<u>wrong people, the wrong solutions, and the wrong processes</u>**.

And, while this is a serious problem for organizations of all sizes, it is a matter of **life and death** for early stage startups… *(as well as all other kinds of organizations that rely heavily upon the software that they create).*

WHY DOES THIS PROBLEM EXIST?

Most business leaders do not know how to "code", and this means that they are unable to personally inspect the decisions that their technology partners make.

In turn, this means that most business leaders are fundamentally unable to manage the technology partners that they work with.

And, when faced with this challenge, most business leaders resort to **blindly trusting the technologists that they hire**; and this just sets up technologists to fail**.**

THE MAIN IDEA

"You get what you inspect, not what you expect"
-- cited in print since 1959.

No matter the size of your budget, winning at software development requires that you inspect the decisions that your technology partners make on a regular basis.

This means that you will need to find at least one Technical Advisor to help you manage everyone else on your team.

This person will need to help you inspect all the things that you are unable to inspect yourself, and support all the people that you are unable to support yourself.

This person will also need to help you identify all the responsibilities that your current technology partners are adept at handling, so that you can hire part-time, cost-effective specialists to handle all the responsibilities that your existing partners are <u>not</u> adept at handling.

This simple approach works whether you are a bootstrapped founder, a leader of a billion dollar enterprise, or a manager of a technical team that is operating within a larger organization.

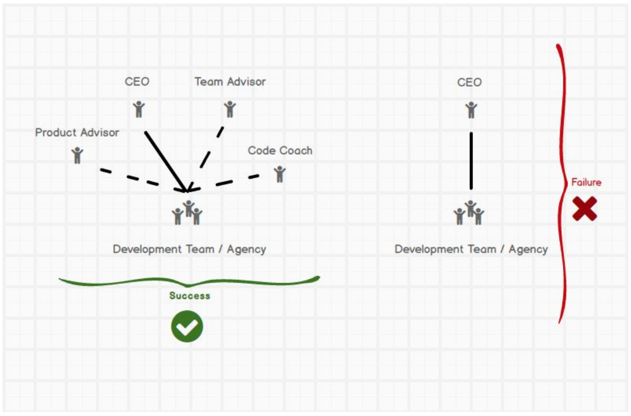
It works whether you prefer to manage software developers directly, outsource everything to technical agencies, or delegate everything to internal leaders (VPs of Engineering, CTOs, and Technical Co-founders).

And, remarkably, it even works when managing other kinds of risky and expensive endeavors that have nothing to do with software development.

THE DEFINITION OF SUCCESS

For very small teams, success usually looks like a single coder that is supported by several advisors, coaches and contractors.

And, for very large teams, success usually looks like an army of technical staff members which is orbited by a somewhat smaller army of independent advisors, coaches and contractors.



A QUICK NOTE ABOUT MINIMUM VIABLE PRODUCTS

Many readers will interpret this guide through the lens of a widely popular methodology known as MVP (Minimum Viable Product), advocated by Eric Ries in his book "The Lean Startup".

While the methodology itself is not the topic of this guide, readers that are familiar with it should know that the approach often fails because CEOs and investors are fundamentally misaligned with the technologists that support them.

This means that you will need more than the MVP approach to prevail.

This guide will provide you all the additional tools required.

The Lean Startup can be found at https://www.theleanstartup.com.

INTRODUCTION

As a CTO-turned-entrepreneur, I often find myself hiring people to do things that I cannot do myself.

And, when this happens, I usually end up asking three questions, over and over again.

1. Is my team talented?
2. Is my team working cost-effectively?
3. And, **most importantly**, is my team solving the right problems... or am I just chasing foolish and misguided solutions?

Perhaps you can relate.

If so, learning how to answer these questions is what this guide is all about.

It’s about leading.

It's about venturing into the unknown with a team of people in tow - even when you don’t have the slightest clue what you’re doing.

TRUST BUT VERIFY

Early in my career I thought that the key to success was finding and recruiting people that I could trust.

However, over the years, I realized that I was *dead wrong*.

<u>I began to understand that trust was **never enough**.</u>

I learned that, as a manager, I had to inspect all the key decisions that my teammates made as we went along together. Otherwise, I found that even my most capable and well-aligned teammates <u>unintentionally</u> wasted all my time and money trying to solve problems that they didn't need to solve.

And, of course, I found that the rest of my team <u>intentionally</u> wasted all my time and money trying to find ways to pad their pockets.

Which is why, today, I sing a completely different tune.

Now, my motto is “trust, but verify”... a management tenant so basic that you may have actually fallen asleep while reading this sentence.

However, as boring as this concept might be, it brings up one <u>very</u> important question, and that question is: “how does a leader actually verify and inspect all the things that they don’t understand?”

To which this guide provides the following answer...

THE FRAMEWORK

Disclaimer: This isn’t rocket science.

Whenever I hire someone to do something that I am unable to do myself, I actually hire two people. First, I hire an expert with decades of experience, and then I hire a cost-effective “worker bee”.

If I have $10 to spend, I give about $8 to the worker bee and the rest to the expert.

And, here comes the key - <u>I don’t ask the expert to manage the worker for me</u>. Instead, *I ask the expert to show me how to direct the worker myself.*

Later, when I understand how to measure the performance of my worker, I usually turn my coach into a manager so that I can move on to other things... but, when I do, I usually hire a new coach to show me how to manage my old coach-turned-manager.

And then, I rinse, and repeat the cycle.

Just to be clear, here's what I <u>don't</u> do:

1. I don’t hire inexperienced workers and then expect them to magically solve all my problems without regular inspection and support

2. I don’t hire experienced managers and then expect them to cost-effectively manage things for me without knowing how to evaluate their performance

3. And, I don’t hire external agencies, contractors, or consultants; and expect them to educate me about how to spend less money with them

Instead, I hire strategic advisors and tactical coaches, and then I expect them to give me control when I otherwise have no control.

By doing this, I gain levers that I can pull to achieve results, and I gain checks and balances which protect me when I don’t have time to pay attention.

Please consider using this approach whenever you find yourself "leading... that is, *doing something that you’ve never done before.*

And, when it comes to expensive and complicated endeavors *such as software development*, I encourage you to consider that this approach might be the **<u>only</u>** approach that makes any sense at all.

PART I
ADVISORS

Most CEOs assume that Technical Advisors:

1. Are optional
2. Are “strategic”
3. Should sit on a Board
4. Should be hands-off
5. Should be equity-based
6. Should have successful exits behind them

And, while most of these assumptions might hold true for most types of advisors, each of these assumptions are **dangerously incorrect** when it comes to Technical Advisors.

The truth is that failing to hire a “hands-on” Technical Advisor is the #1 mistake that founders make.

CHAPTER I
WHY TECHNICAL ADVISORS ARE SO IMPORTANT

In short, Technical Advisors are required to build software cost-effectively because managing software development is very unlike managing most other things (such as operations, marketing, and finance).

#1 Most executive managers do not know how to code

This means that most executive managers are unable to inspect the decisions that their technical partners make, or support their technical partners when they need assistance

#2 Most organizations are unable to afford experienced CTOs & CPOs (aka Chief Technology & Product Officers)

This means that most organizations have to rely on inexperienced technologists for guidance, and these leaders "do not know what they do not know"

#3 Most technologists benefit greatly from the process of building software (e.g. income, political power, and job security)

This means that most technologists are incentivised to build as much software as possible, and to avoid questioning the usefulness of the solutions that they create (including "well-aligned" CTOs and Technical Co-founders)

And, here's an simple non-technical example to help illustrate these points:

—

CASE STUDY (1 of 8)

This Founder Lost $80,000 and 9 Months –
Don't Let This Happen to You

A young founder named Billy hired me to review a mobile app that he was building, and I became concerned the moment that we sat down together.

Here's why: Billy's app allowed his users to share media with the other users who also had his app, but it didn't allow his users to communicate with the people who didn't yet have his app.

This meant that prospective new users had to download his app before understanding the value it might create for them. And, this was a real problem because most people don't download random apps until they see demonstrated value (see https://think.storage.googleapis.com/docs/mobile-app-marketing-insights.pdf).

To illustrate the point, I asked Billy if he, personally, would be more likely to click on a random link or download a random app; to which his response was simply: "Uh oh…" Soon thereafter, Billy began to understand that the features that he had built into his app simply didn't matter because nobody would ever see them.

Then, as we looked at his app even closer, we realized that this meant that he didn't need an app ***at all****.*

Next, I helped Billy design a new prototype that was 100% web-based and cost him only $2,000 to test. And, this new prototype immediately gained the traction that he was looking for.

This meant that Billy had just wasted $80,000 on an app that he would never use… all because Billy had walked into a development agency nine months earlier and asked if they could help him build his idea.

In fairness, readers of this guide that are familiar with the MVP methodology might say that Billy was at fault because he failed to test his assumptions early and often; and while this is true, there is more to the story.

The real problem is that nobody in Billy's circle of influence was motivated to show him the simplest, fastest, and cheapest path forward because everyone that builds software for a living makes money from building software - not from building *less* software.

Key Concept *- Everyone On Billy's Team Was Financially Motivated To Encourage Him To Spend More Money*

And, if this can happen at a product level, imagine how this same dynamic impacts the myriad of decisions that are made on a daily basis about the structure of your code - something that you will never be able to inspect yourself.

Frightening? It should be.

THE "PRINCIPAL AGENT" PROBLEM

Remember that software developers, CTOs, and development firms can not self-assess their own performance because they all earn a living building software, and there are no objective metrics by which you can measure their performance; unlike people who work in sales, marketing, operations and finance.

Academically, this problem is known as the "Principal Agent Problem", and this quote from wikipedia sums it up well: (https://en.wikipedia.org/wiki/Principal–agent_problem)

> "[The problem] occurs when one person or entity (the "agent") is able to make decisions on behalf of, or that impact, another person or entity: the "principal"… where the two parties have different interests and asymmetric information (the agent having more information), such that the principal cannot directly ensure that the agent is always acting in their (the principal's) best interest."

Pause for a moment and really consider this.

It should be fairly obvious that you'll never understand code at the same level as your development partners, and that you never want to.

But, if you don't understand code, how are you going to ensure that you don't overpay for what you get, dig yourself into a hole with poorly implemented solutions, or waste time and money building unnecessary solutions?

And, do you honestly expect your teammates to tell you when you should not pay them to build software; even when doing so means that they will have to find another gig?

Remember that:

- CTOs benefit from convincing you to build teams that require their management

- Developers benefit from convincing you to build things you could buy elsewhere
- Architects benefit from convincing you to build systems only they understand
- Agencies benefit from convincing you to build expensive and unnecessary solutions

And, every person that you hire is also limited by their own experiences and perspectives. Even highly seasoned CTOs need help from time to time with the things that they are not familiar with.

Key Concept - *Even Well-Aligned,*
Equity-Based Partners Have Severe Limitations

THE “RUBBER BAND” PROBLEM

As severe as the Principal Agent Problem is, it is only one part of a larger problem.

The larger problem is that the cost of technical mistakes compounds with each day that passes. This is truly what makes software development so incredibly difficult to manage.

Key Concept - *It’s Already Too Late*
To Put Out The Fires By The Time You See The Smoke

Visualize software like a ball of rubber bands.

Each band that is added to the ball increases the difficulty of accessing the center of the ball because doing so means that each band must be unwound and then rewound again. This means mistakes made early on, at the core of the ball, are the most expensive to fix.

And, software developers need to access the core *often*. This means that every line of code that is added to your project makes every line of code that came before it more expensive to work with.

In turn, this means that you won't be able to just hire a new team when things get bad, like you can with most other things. You'll either have to start over, or spend years fixing the mistakes that you made before hiring your newest team.

I'm not exaggerating here, it often takes years to fix technical problems because of the intricacies involved.

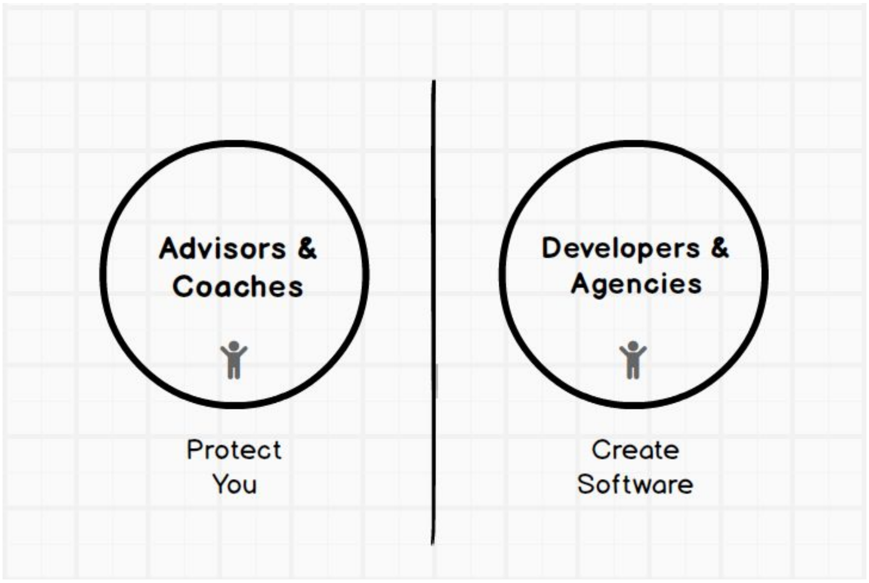
SOMEONE OUTSIDE YOUR ORGANIZATION HAS TO HELP

It comes down to this - software development is too important, and too costly, to learn how to manage as you go along.

Everyone that you work with is pitted against you (whether they want to be or not), you have no way to inspect their performance, and small mistakes fester undetected before blowing up in your face in a very expensive way.

This means that you must hire experienced people to help you inspect and support everyone else on your team.

Remember that there is a very good reason why the President of the United States has a Cabinet of Advisors.



SUMMARY

Never put all your faith in just one person… or firm.

It’s rarely a good idea to rely on a single person in any area of business, but it just so happens that software development is the first endeavor that many entrepreneurs face which is expensive, complex, and difficult to fix after problems become evident.

You can’t manage software development yourself, you can’t expect anyone that is responsible for building your software to guide you, and you can’t wait to hire seasoned leaders down the road.

You have to establish checks and balances early and often in order to build software quickly and easily.

***NOTE**: It's important that you hire Technical Advisors even when you hire experienced people to execute your vision because it's important that the most senior people on your team never stand to gain substantially from the recommendations that they make. <u>However</u>, you will need less help from your advisors when you hire truly experienced executors.*

CHAPTER II
FINDING YOUR FIRST ADVISOR

This chapter will help you get up and running with your first advisor, and understand the differences between paid and unpaid advisors.

THE MOST EFFECTIVE ADVISORS GET HANDS-ON WHEN NEEDED

The fact of the matter is that most organizations cannot afford to hire extremely capable technical partners (e.g. decorated agencies and CTOs)... and the few organizations that can hire seasoned technical partners usually find that their growth quickly outpaces their partner's ability to grow with the organization.

This means that most organizations need Technical Advisors who can get their hands dirty from time to time.

The sample job description below will help find the help that you need:

HANDS-ON TECHNICAL ADVISOR WANTED

<u>Responsibilities</u>

- Be available for conversations, brainstorming, client meetings, code reviews, and so on
- Review and improve business value of products and solutions
- Review and improve quality of software
- Review and improve cost-effectiveness of development team
- Review and improve performance of technical culture
- Review and improve technical roadmap, architecture and specifications
- Review and improve product roadmap, design and specifications
- Review and improve technology costs, including build vs. buy decisions
- Triage services, as needed, in the event of a technology crisis
- Translate business and technology language and jargon
- Support capital search in meetings with investors
- Support business development and sales in meetings with potential partners and clients
- Coach and mentor internal technology staff and/or CTO
- Recruit, vet and onboard team members
- Research and assist with special projects

THE MOST EFFECTIVE ADVISORS UNDERSTAND PRODUCT DESIGN

The most important thing to remember when developing software is that you need to focus on solving problems that your users actually <u>need</u> you to solve for them.

Otherwise, you are just wasting your time and money building software that no one will ever use.

This requirement confuses most non-technical leaders.

"Founders are supposed to be the visionaries behind their products, and Technical Advisors are supposed to be code wizards - right"?

<u>Wrong</u>.

Designing effective software products requires years of training, exposure to multiple markets, and a deep understanding of technology.

All of which are things that you probably don't have.

Key Concept *- If You Only Have One Advisor,*
That Advisor Must Understand Product Design

When bringing products to market, look for people that are experienced at bringing products to market. Likewise, when scaling established products, look for people that are experienced at scaling established products.

Ask candidates what makes them great at designing products. You're looking for broad experience in multiple disciplines (marketing, finance, etc.), as well as specific training in designing effective user experiences.

Ideally, candidates should also be able to architect complex software and build effective technical teams, but these are only secondary concerns when just starting out. Remember, you can always recruit more than one advisor to get access to all the skill sets that you need *(the next chapter will explain the different kinds of advisors that are available)*.

YOU'LL PROBABLY HAVE TO TRAIN SOMEONE

Product Design is a new discipline that emerged during the last decade, whereas Software Development (as a field) emerged over 30 years ago. This means that there are far more technically oriented CTOs available on the market than there are product focused CTOs - or technically focused CPOs (Chief Product Officers).

This also means there are VERY few consulting CPOs available. For example, in LA, where I live and work, I know of 2 qualified consulting CPOs. Myself and one other (amazing) colleague.

The bottom line is that you will probably need to recruit a CPO that is working somewhere else on a full-time basis, and you will probably have to show this person how to become an effective advisor for you.

Key Concept *- Look For CPOs, Not CTOs,*
When Looking To Cultivate Your First Advisor

Try searching LinkedIn and your local community for CTO/CPO, CTPO, "Technology and Product Officer", "technically focused CPO" and "product focused CTO".

Then ask them to read this guide to help them understand your expectations.

Or visit my blog at JayCrouch.com, or email me via connect@JayCrouch.com for quotes, referrals and introductions. #shamelessplug

YOU DON'T NEED THE MOST

PRESTIGIOUS ADVISOR ON THE PLANET

Investors will tell you that they don't care who your advisors are, and they'd be right.

Don't waste your time looking for advisors with notable brand experience or successful exits unless you are running a large company. These accomplishments usually correlate with "too busy to pay attention" and "out of touch with the challenges small teams face".

Key Concept *- You Want Practical Executors With Recent Experience Working At Your Current Stage And Scale*

Fill your advisory board with prominent, non-technical figures that expand your network while hiring practical Technical Advisors to assist your team in a hands-on capacity.

EQUITY ALONE IS USUALLY NOT ENOUGH

I strongly encourage you to recruit as many equity-based Technical Advisors as you can find. However, know that it's extremely unlikely that anyone with real experience will be willing to get hands-on in return for equity alone.

The reason is that experienced technologists are in high demand and typically value cash a lot more than they value equity.

Key Concept *- Expect Equity-Based Technical Advisors To Ask Great Questions, Not Provide Hands-On Guidance*

And, this is a serious problem because:

- CEOs typically need to demonstrate traction before anyone will pay attention to them on an equity basis - but most early stage founders need a lot of help from advisors *before* they demonstrate traction

- CEOs typically rely on equity-based technical advisors that don't have all of the skill sets that they need - because, at the end of the day, beggars can't be choosy

- CEOs typically waste a ton of time and energy trying to command the attention of their equity-based advisors who are focused elsewhere - and organizations are vulnerable when CEOs are distracted

Which means that you'll probably need to hire someone with cash to fill in the gaps that your equity-based advisors leave open.

PAID ADVISORS CAN'T BE TRUSTED

The thing to know about paid advisors is that you can't completely trust them. But, then again, you can't completely trust equity-based advisors either.

Advisors, just like everyone else, stand to gain something from the recommendations that they make in addition to the compensation you provide them directly. Sometimes they earn referral fees from the people they introduce you to, other times they benefit from representing themselves as your advisor in order to further their own career, or leverage some other kind of relationship.

Always remember that there are an infinite number of ways that both equity-based and cash-based advisors might benefit from a relationship with you.

***Key Concept** - Even I (The Author)*
Stand To Gain From My Own Recommendations

The bottom line is that it's unreasonable to expect anyone on your team to always act in your best interest, see all perspectives, and be experienced in all things - including both paid and unpaid advisors.

A BAD ADVISOR IS SIGNIFICANTLY BETTER THAN NO ADVISOR

Ironically, the fact that you can't completely trust your advisors is exactly what makes them so valuable.

It's nearly impossible to spot conflicts of interest when working with developers, CTOs, and firms, but it's relatively easy to do so when working with advisors.

Key Concept - *You Can't Manage Technologists, But You Can Manage Advisors*

Why? Because developers, CTOs, and firms all earn a living creating code that you'll never really understand. Advisors, on the other hand, only provide counsel and support which you can easily inspect and verify.

A PERSONAL EXAMPLE FOR ILLUSTRATION

As an Advising CTO/CPO, I am often asked to help find and recruit staff.

Sometimes I refer my clients to vendors, sometimes I help them to recruit their own staff, and sometimes I provide them with subcontractors. The important thing to note is that in each case I usually earn staffing fees or referral fees in addition to my advisory compensation.

And, while all of this is commonplace (and extremely helpful to my clients), it also creates a conflict of interest because I stand to benefit from the recommendations which increase my own staffing revenue.

This is why I always recommend that my clients find a friend or another advisor to look over my shoulder when my staffing income begins to become substantial relative to my advisory compensation. If the conflict of interest grows large enough, I'll even recuse myself as Advisor (or Advising CTO), and transition to CTO, Consultant, or Agency, in order to make it very clear where my responsibilities lie.

I do this because I'm deeply connected to the Los Angeles community of CEOs, investors, and influencers. This means I care about protecting my reputation and fulfilling my responsibility to serve the companies in my care, and this means I must go out of my way to disclose conflicts of interest.

But, even if I *wasn't* connected to the community, **I would still disclose these conflicts of interest**, because my clients (and their investors) can easily inspect the work that I perform as an Advisor without knowing much about technology.

And, **this,** is the true value of working with Technical Advisors… you don't need to know much about technology in order to manage them.

FREE UP CASH BY HIRING LESS EXPERIENCED DEVELOPMENT PARTNERS

Many founders believe that paid advisors are too expensive to engage until *after* raise significant capital.

This is just silly.

The reality is that you can not afford to *not* hire an advisor. Advisors help you save far more money than they require - that is their fundamental value proposition… unlike CTOs and agencies whose value proposition is to make software development simpler for you.

Remember that you're far better off with an experienced advisor and an inexperienced development partner than a semi-experienced advisor and a semi-experienced development partner - or no advisor at all *(you'll find more information about this topic in Part II - Executors).*

Use the ubiquitous 80/20 rule as a rough guideline to attract the experienced advisors that you need until your technical budget exceeds $500,000/yr.

Plan on allocating at least $2,000 to technical advisors when you only have $10,000 with which to build a prototype. Later on, allocate at least $100,000 for technical advisors as your technical budget approaches $1,000,000.

SUMMARY

You're probably going to need a hands-on advisor, and you'll probably need to pay your advisor with both cash and equity to keep their attention.

Hands-on advisors allow you manage an otherwise unmanageable situation because you can easily identify where their conflicts of interests lie.

You don't need people with notable brand experience or big exits. That said, experience obviously matters, and recent tactical experience matters most.

There is no school or organization that accredits advisors, which means you are going to direct your advisors how to help you best *(tip: ask your advisors to read this guide).*

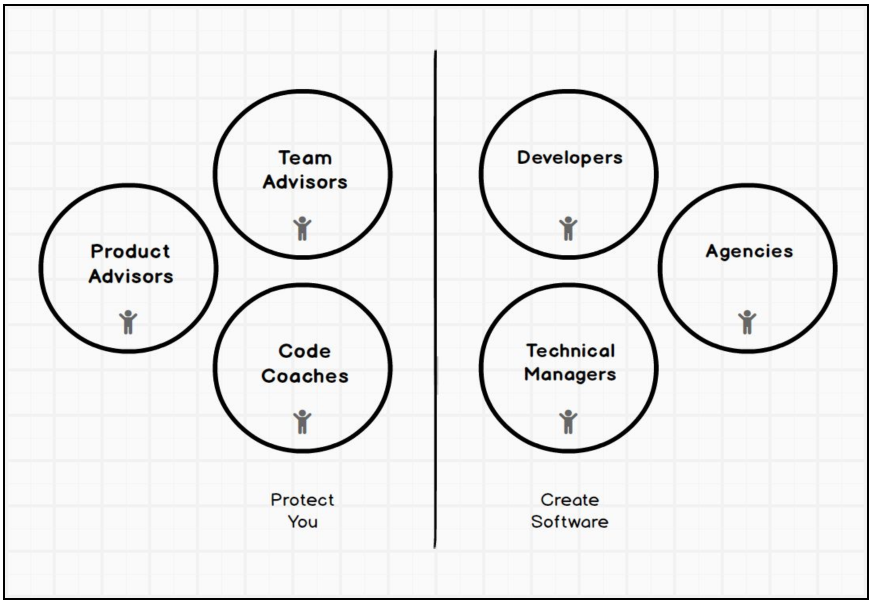
CHAPTER III
DIFFERENT KINDS OF TECHNICAL ADVISORS

This chapter will help you identify the three different kinds of Technical Advisors that are available, and know when to work with each.

SUCCESS = PRODUCT + PEOPLE + TECHNOLOGY

When just starting out, you will probably only need to work with a single Technical Advisor... but later on you will probably need to find two or three specialists because every Technical Advisor has strengths and weaknesses just like every other kind of technologist:

1. **Product Advisors** help you find ways to simplify your solutions
2. **Team Advisors** help you drive the performance of your team
3. **Code Coaches** help you ensure that your code is scalable



And, when the total cost for all your paid advisors starts to become significant, it's time to find yourself an equity-based Advisor (or friend) to keep tabs on all your paid advisors.

Understanding Product Advisors

As mentioned in the previous chapter, nothing else really matters when you are solving the wrong problems.

This means that your first advisor needs to be great at figuring out which problems are actually worth solving in order to help you to find the cheapest and simplest path forward.

CASE STUDY (2 of 8)
Jim Almost Lost Everything

Jim met me at a panel, much like Billy. But unlike Billy, Jim asked me *to review his MVP before starting to build it.*

Jim was already working with a firm when we started working together, and the problem, as usual, was that his firm was focused on their interests, not his. They had already convinced him to spend $50,000 on an MVP, and just like Billy's firm, had designed a product that made absolutely no sense for his users.

To illustrate, Jim was targeting millennial users who were renting apartments, but his firm had designed a prototype with a million features and an interface that felt like a 1990 corporate intranet. It was never going to work for millennial users, and Jim understood why as soon as I pointed out the flaws to him.

So, I helped Jim design a new product that borrowed from Instagram and respected the way that his users wanted to interact with their phones. Then, we simplified his go-to-market-strategy so that Jim could roll out one feature at a time, thus saving money by not building lots of unnecessary features until he could verify that his users liked his ideas.

And finally, I helped Jim find a new firm that was happy to help for $10,000.

In the end, he saved $40,000 and months of time.

—

Primarily, Product Advisors help you:

1. Ensure that the people designing your solutions do not overcomplicate things (*think Eric Ries's MVP methodology, but at an organizational level*)
2. Ensure that the people designing your solutions understand key principles like the importance of user acquisition and user engagement
3. Ensure that the people designing your solutions solve the right problems

And, if you are the person responsible for designing your solutions, your

Product Advisor's job is to keep *you* from becoming your own worst enemy.

***NOTE**: Few startups can afford to hire experienced CPOs, so Product Advisors tend to start off heavily involved, then taper off while MVPs get built, and then come back again with a vengeance once the product is up and running.*

Product Advisors need broad training and strong executive communication because they help shape the direction of the company.

This means that Product Managers are not qualified to be Product Advisors, and I mention this because many founders attempt to hire Product Managers in lieu of Product Advisors in order to save money.

That said, I *have* found that Product Managers can work cost-effectively when paired with senior Product Advisors who serve in a less hands-on capacity.

Also, because Product Advisors assist in driving revenue, they really need to understand WHY the business exists, what it's goals are, and where the industry is going.

Granted, it's ultimately the responsibility of the CEO to articulate the WHY behind the company, and ensure alignment across the executives of the company... but, if Product Advisors are out of step here, they can make recommendations that runs counter to the type of company that is being created.

UNDERSTANDING TEAM ADVISORS

Once you have established a product direction, it's time to put a team together to help you build the software that you need.

Primarily, Team Advisors help you:

- Locate cost-effective people (and agencies)
- Evaluate and improve the performance of the people on your team
- Implement best-practices and processes

NOTE: *Examples of powerful job ads and more information about hiring developers can be found on my blog: https://jaycrouch.com/posts/win-the-war-on-talent-stop-writing-job-ads-like-everyone-else.*

—

CASE STUDY (3 of 8)

Don't Be Fooled Into Expensive "Discovery" Projects

I recently worked with a pair of co-founders that were looking for someone to help them build an MVP.

They were interested in a few candidates, but one stood out above the rest. He was a long time friend of the founder, all the way back to college, and the founders felt very comfortable with him.

Let's call this guy "Bob".

Bob disliked working hourly because "there was no motivation to finish quickly". He felt it was better for everyone to work on a project basis, and because of that, he proposed a quick discovery phase for $20,000 to figure out what to build before starting to build anything.

When I pressed him for answers, he said that he guessed we were looking at a 6 month project based on the documentation that he had seen. He also said he didn't have an hourly rate, but if he did, it would be somewhere near $200/hr.

I have to admit, he was a good salesman. The founders were convinced

(they had raised several million and figured a $250,000 MVP sounded about right for everything they wanted to do.)

And… I advised the founders stop talking to Bob.

Here's why.

*Bob was the worst of two worlds. He wanted to build everything **his*** way - which locked the founders into working with him down the road. And, he wanted to work on a project basis which forced the founders to prematurely commit to a 6 month project - and allowed him to pocket an expensive discovery phase.*

At no point did Bob ever stop to consider what was best for the business.

Bob never bothered to challenge the founders to build a smaller MVP. And, his justification for not wanting to work on an hourly basis was a dead giveaway that he didn't think like an Advisor, or even a CTO. A large part of the CTOs job is to make sure everyone is working hard.

In the end, we created a very simple product roadmap and hired a full time developer for $50/hr. Our "discovery phase" lasted one day.

And, of course, the founders called me two weeks later and told me that they wanted to change the roadmap because of all the exciting things that they had just learned about their customers.

Had they chose to work with Bob, they would have had to pay him to change all the specifications that they had just paid him $20,000 to create. And most likely, they would never have been able to get rid of Bob because no other developers would have known how to work with his code.

—

Team Advisors should be very skilled at recruiting cost-effective resources and setting up processes *for organizations of your current size and scale*. There is no sense in hiring a Team Advisor with access to deep networks of enterprise developers if you aren't yet an enterprise level organization.

Also, it is usually most cost-effective to pair a less experienced technical manager (or Technical Co-founder) with a senior Team Advisor in the same way that it can be cost effective to pair a Product Manager with a senior Product Advisor.

When you go looking to hire your first Team Advisor, screen candidates with this criteria:

- Should be willing to test the waters with a low risk engagement
- Should be open to using any technology ("technology agnostic")
- Should be comfortable outsourcing, and cultivating internal teams
- Should **NOT** be comfortable writing code for you
- Should be extremely good at expectation management and communication

Start slowly and look for alignment.

And finally, save your equity pitch for developers and strategic ("hands-off") advisors. Most experienced CTOs and Team Advisors don't care much for relationships that are based on mostly equity.

UNDERSTANDING CODE COACHES

Once your product direction and team are set, the final step is to ensure that your software remains scalable as you build it, and make sure that your developers do not take shortcuts that might lock you into working with them down the road.

These guys are the "nerds", not the "leaders", and this means you usually don't need to talk with them directly. Instead, Code Coaches work at the lowest level to review your code and architectural patterns on a regular basis and report their findings to your Team Advisor (and CTO, if you have one).

I use the word "Coach", rather than "Advisor", because this type of Advisor operates at a tactical level rather than interacting with your executive officers and external stakeholders. It's a small but important distinction *(more on this in*

Chapter VIII – Structuring Your Team).

—

CASE STUDY (4 of 8)
The Quality Of Your Code Matters

Fred decided to manage a team of developers with the assistance of just one Technical Advisor.

Three years later, Fred had invested $7,000,000 dollars into his company, but had almost no users to show for it. Yup, 7 million and zero traction.

Most of his budget had been spent on building custom software, and the rest had been spent on ineffective marketing and operations because his software never worked.

In case you might be asking, "why did this happen?" The answer is that Fred never asked his Technical Advisor to review his code. If he had, his advisor would have told him that he didn't have the expertise required (I know this because I met him).

And, this mattered because the developers who wrote the code lacked the experience required to convince the founder of basic things, such as the need to invest time and money in automated testing.

Which meant that the lack of automated tests slowly created an unstable environment that stopped development in its tracks.

Long story short, I was able to fix things up when Fred brought me in, but it cost him several hundred thousand dollars to right the ship, and in the end, we ended up with a final product that pros could have built for less than $1,000,000 from the ground up. ***Ouch.***

—

***NOTE**: Don't worry about finding Code Coaches, your Team Advisor should help you find, hire and manage them. They should be experts with specific technologies, and have recent practice working with languages like Ruby on Rails, Python, Java, PHP, .NET, etc.*

For early stage teams, the best Code Coaches are typically working as Directors of Technology, or Software Architects, at medium sized businesses. For later stage teams, the best Code Coaches are typically working as Chief Architects, or Enterprise Architects, at very large companies.

And finally, remember that Team Coaches and CTOs should not serve as Code Coaches whenever possible, even if these people can serve you in both capacities.

Here's why:

First Code Coaches are almost always cheaper than Product and Team Advisors (and CTOs) on an hourly basis.

Second Product and Team Advisors can not serve you objectively when they limit the options that they consider by focusing on technologies that they are familiar with.

Third Code Coaches should be replaced if you change technologies, or add new technologies, and you'll lose your Advisors in the process if they are also serving as Product or Team Advisors.

IMPORTANT! YOUR ADVISOR IS NOT YOUR CTO (OR CPO, OR CSA)

If you're thinking that Team Advisors, Product Advisors, and Code Coaches sound like Chief Technology Officers (CTO), Chief Product Officers (CPO), and Chief Software Architects (CSA) - you'd be right.

Each of your advisors are probably CTOs, CPOs, and CSAs at organizations that are larger than your organization. However, the important thing to remember is that your advisors are never your CTO, CPO, or CSA. This means that you should never hold them responsible for deadlines - instead you should only expect them to help you manage and support the people on your team that are responsible for creating results.

Also, it's okay to turn an advisor into a CTO, CPO, or CSA, but know that the moment you do so, your advisor/coach becomes compromised and unable to assist you as an advisor. The fact is that no one can objectively help you assess their own performance, not even advisors.

Key Concept *- Advisors Are Not Responsible For Driving Results, Your Team Is (Or You Are)*

Think about it this way. You, being a good leader, need to push your team from time to time. This means that, at some point in the near future, you are going to tell the people building your software to move faster regardless of what it takes to make it happen.

In turn, a reasonable CTO might warn you that you are making a costly decision, but eventually back down and take the shortcuts necessary; because after all, you're the boss. And, this becomes a problem because you really have no way of understanding the long-term costs associated with your decisions.

But an advisor (or coach) can continue to push back. An advisor can remind you to repair the shortcuts that you have taken because they are not responsible for meeting specific deadlines or milestones.

SUMMARY

Three things matter far more that shipping code quickly and cheaply.

1. Your software must solve problems that actually need solving
2. You must be able to easily switch out your teammates at any time

3. Your solutions must be cost-effective and scalable

To realize the full power of coaches, work with specialists.

Your first advisor, the Product Advisor, needs to be great at figuring out which problems are really worth solving in order to help you to find the cheapest and simplest solutions to those problems.

Your second advisor, the Team Advisor, needs to show you how to locate people, evaluate their performance and implement your processes.

Your final advisor, the Code Coach, needs to work at the lowest level to review the code and architectural patterns that your software developers create.

Advisors and coaches aren't responsible for creating results - they only help you make sure that your team has the best chance of creating results.

PART II
EXECUTORS

Once you have found all the advisors and coaches that you need, it's time to select the people that you want to help you build your software.

This section will help you avoid mistakes by revealing the most commonly held misconceptions about agencies, software developers, technical co-founders, VPEs, and CTOs.

It also illustrates the most common pitfalls to watch out for when working with each.

CHAPTER IV
AGENCIES

There are two kinds of agencies.

Staffing agencies (aka "staffing firms"), and full-service agencies (aka "development firms").

As a group, staffing firms are cost-effective which development firms are not (but they are convenient).

Staffing firms typically markup their staff about 10-50%, and then provide these staff members to you directly so you can manage them yourself.

In contrast, development firms typically markup their staff about 100%-200% and then manage these staff members for you. That's right. **50-70%** of the money spent with a development firm does not go to the people doing the work!

WHEN IS IT A GOOD IDEA TO WORK WITH A STAFFING FIRM?

Staffing firms are a good option for organizations that:

a) Do not have the time and resources required to recruit quality staff on their own
b) Want to offload the HR costs associated with managing employees (especially offshore employees)
c) Or, need the ability to change up their staffing requirements on a regular basis

They are not a viable option for organizations that need to control the way their employees work, or reduce their employees' salaries with stock-option grants.

Staffing firms compete primarily with recruiters, and they usually win because staffing firms charge their fees over time, rather than upfront as recruiters do.

If you want to work with one, consider working with TechTeamExtension.com, a staffing firm that I manage when I am not advising founders.

TechTeamExtension provides nearshore and offshore technical staffing services, and occasionally provides development services for elite clients.

Which brings us to development firms.

WHEN IS IT A GOOD IDEA TO WORK WITH A DEVELOPMENT FIRM?

Development firms are a good option for teams with deep pockets and tight deadlines (e.g. the Boston Consulting Group, or GE).

They are usually not a viable option for early stage startups.
I believe this so strongly that I will refuse your business if you call me looking for a full-service development agency without having first found an advisor to

help you manage me and my team.

If that doesn't persuade you to fire your development firm tomorrow, *this will* – know that development firms are completely misaligned with the clients that they serve because they increase their margin by completing as <u>little</u> work as possible when engaged on a fixed-price basis; and they also increase their margin by completing as much <u>unnecessary</u> work as possible when engaged on an hourly basis.

This means that while development firms can *help you* design software and *help* you ensure its quality, they can not be *responsible* for delivering either of these things - no matter how you engage them. They can only be held responsible for creating code that meets the specifications given to them.

Do You Remember Billy?

Here is a quick refresher in case you missed his story: Billy paid $80,000 to a firm to design a mobile app which took over 8 months to build. But, Billy didn't need a mobile app because his users would never use it regularly. He only needed a website that could be easily embedded in social media, and the solution I gave him cost only $2,000 and 2 months to build.

That's a big difference. Why did this happen? Because the firm he was working with had no reason to help him save money. Most firms will happily build any solution they can which allows them to turn a profit.

And, in case you might be thinking Billy is alone - he isn't.

Billy is the rule, not the exception.

—

CASE STUDY (5 of 8)
Referrals Aren't Everything

I recently worked with a founding team who was interested in working with a development firm because their investors had successfully worked with the same agency previously.

By the time I was brought in, the agency owner had already convinced my clients to build a quick MVP using an exciting new technology that he was already using with another client. My client was very impressed when the owner talked about this technology because it sounded like "fancy jargon, fancy jargon, fancy jargon…".

But, when I asked the owner if anyone would be able to work with this exciting new technology, should anything happen to him and his team, he responded with "no, it was too new".

Frankly, I had already ruled out the agency based on this response, but I was curious to see a demo of the technology... and of course it turned out that it had absolutely nothing to do with the founders' needs.

This meant that the agency's proposal would have locked the founders into permanently working with his firm based on a sales pitch for technology they they had absolutely no use for.

Here's the upshot. Referrals and reputation really don't matter much.

Good developers show you how to build things simply. Bad developers make s#!t complicated.

―

Today, Billy has a thriving startup. But, how much farther ahead would Billy be if he hadn't worked with a firm that was eager to build the most expensive mobile application that he could envision?

All Billy needed was a little bit of time with an independent party before he started his journey (*e.g. me*).

Remember that development firms employ specific people who are trained to get you to spend more money with them. These people are called "Product Managers", but they usually double as sales reps for agencies because they are in a position to convince clients to build more features.

Also, remember that it's usually *very* difficult to get rid of a development agency once they are in the door unless you have a Code Coach watching closely to make sure that the code they create is built in ways that other developers can work with.

Many firms bank on this fact when offering to build small projects for "discounted" rates. These firms know that you probably won't be able to leave them once you start working with them. This applies to large agencies as much as it does small boutique agencies… the tech giant SAP once proposed free development for a year if my billion dollar client agreed to switch to their platform. *Why do you think they made this offer?*

The truth of the matter is that every technologist on the planet benefits by making you dependent on them.

<u>I'm not saying they will, and I'm not saying that technologists are evil</u>... but I <u>am</u> saying that technologists' interests do not line up with your interests.

And, this means you need to employ advisors and coaches in order to align your interests with the interests of your teammates… and to support them, because they need your support just as much as you need theirs.

—

CASE STUDY (6 of 8)
When Good Intentions Aren't Enough

Mark was $70,000 into his MVP when he learned what I do. He had never heard of a CPO before, but he was interested to hear my thoughts on the mobile app his firm was building for him.

Mark was a fitness buff who wanted to help people workout with personalized strength and conditioning programs designed by accredited trainers.

The problem was that Mark was a bit too much of a fitness expert himself. He had spent a year designing an extraordinarily complex app that was basically the app he personally wanted to use. And, when I say it was over complicated, I mean that he had shoved an encyclopedia into a mobile app.

To their credit, Mark's development firm had been telling him all along *that he needed to trim the features in his app. But their words fell on deaf ears. Why? Because they were software developers - not seasoned executives. They didn't know how to communicate in ways that he could understand. And, at the end of the day, they made more money doing it his way, so why fight him?*

His conversations with me were different. As an experienced CPO/CTO, Advisor and Coach, my counsel carried the weight needed to make him listen. I was able to share my experiences leading previous organizations, and communicate using metaphors that made sense to him. Also, I didn't earn money by implementing my designs - so my recommendations were easier to hear.

In the end, it was too late to help Mark spend less money on his MVP, but I was able to convince his firm to put his money towards a simpler solution without charging the founder more money. This saved Mark about $70,000 and a year of time.

SUMMARY

Staffing firms are usually cost-effective and low-risk options for teams of all sizes.

In contract, development firms are typically simple to work with, and should only be considered by teams with lots of money and limited time to execute.

Also, you must to be careful when working with development firms because they are motivated to advocate for solutions that earn them more money rather than create the best products, packages, and services for your needs.

CHAPTER V
EXECUTIVE CTOS

There are two different kinds of CTOs.

Those that write code, and those that don't.

This (very short) chapter will examine the pros and cons of working with CTOs that only manage other people, while the next chapter, Generalists, will illustrate the pros and cons of working with CTOs that code (and all other kinds of technical managers that code).

THE BOTTOM LINE

When it comes to working with a real, executive CTO, the truth is that you probably don't need a full-time leader until you have a multi-million dollar technical budget, or you need short-term help to fix a badly broken team.

Chapter VIII - Structuring Your Team will explain specifically why this is so.

Until you reach the point that you need a half-time CTO, you're probably better off paying a Technical Advisor to get a little more hands-on while less experienced leaders manage the internal team.

> ***NOTE****: Chapter VII - Working With Tiny Budgets will walk through multiple ways to compose your team so that you only need a little help from people who don't code for you.*

If you do choose to work with an executive CTO, you will probably need to hire someone in a part-time capacity; and, there are three different kinds of part time CTOs:

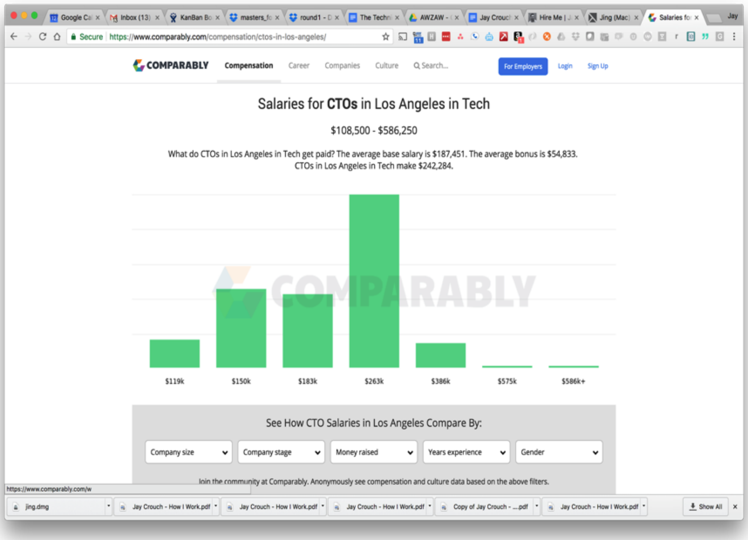
Consulting CTOs	Are professional consultants that only work in a freelance capacity
Interim/Acting CTOs	Are CTOs that are focused elsewhere, but are willing to help you on the side
Advising CTOs	Are Technical Advisors that get so hands-on that they end up actively manage the organizations that they serve *(but are still not held responsible for creating results)*

HERE ARE THE FACTS

Acting/Interim CTOs that moonlight on the side can be very cost-effective, but their primary gig always comes first. This means that their tools and processes aren't designed to serve you; instead their tools and processes are designed to serve the other guy.

And, because of this, it's very easy to believe that you have a part-time CTO, when in reality, all you really have is an overpaid advisor who isn't responsible for generating results.

Consulting CTOs aren't much better. While most consultants often help multiple clients at once, they are usually too expensive to create value for small teams on a long term basis.



Salaries for **CTOs** in Los Angeles in Tech

$108,500 - $586,250

What do CTOs in Los Angeles in Tech get paid? The average base salary is $187,451. The average bonus is $54,833. CTOs in Los Angeles in Tech make $242,284.

See How CTO Salaries in Los Angeles Compare By:

Join the community at Comparably. Anonymously see compensation and culture data based on the above filters.

To understand why, consider that in top-tier markets like LA, the average consulting client must be willing to pay a Consulting CTO at least $10,000/mo because Consulting CTOs need to earn at least $30,000/mo in order to justify not taking a full time role, and it's very difficult for them to effectively manage more than a few organizations at the same time.

Why $30,000/mo? *Because qualified CTOs who are employed full-time earn a minimum of $20,000/mo + bonuses + equity, and consultants have to make more than employees make in order to compensate for self-employment taxes, marketing, relationship management, and lack of stability.*

Sometimes qualified Consulting CTOs will accept lower retainers - but they usually aren't able to actively manage a team for less than $6,000/mo - and this means that they are usually not a viable long-term option for startups.

This also means that many Consulting CTOs resort to upselling early-stage clients services that they don't really need. Sure, Consulting CTOs can do other things to justify a large retainer, like project management and code reviews - but these activities can usually be accomplished by other people who are far more cost-effective under an Technical Advisor's supervision *(the next* *chapter to will explain this in more detail).*

That said, Consulting CTOs are a good short-term option for teams that need to fix things in a hurry, and a good long-term option for medium sized teams that operate within agencies and other types of organizations that don't care much about developing a strong internal technical team.

Chapter VIII - Structuring Your Team will explain this more deeply.

SUMMARY

Interim and acting CTOs are usually not focused on your business.

Consulting CTOs are usually not able to create sustainable value for small teams.

Advising CTOs are usually able to create sustainable value, just like Technical Advisors.

And, all three options can effectively bridge short-term gaps between capable leaders.

CHAPTER VI
GENERALISTS

"CTOs that code", VPs of Engineering, Technical Managers, and Technical Co-founders are usually incredibly cost-effective hires *(second only to advisors)*.

However, they should never, ***ever***, "handle everything".

A decade ago, the best way to build software was to hire a renegade nerd and hope that everything worked out. I know this because I was one of those renegade nerds.

But, today, one-man-bands are no longer a viable option.

The reason is that modern software development is extremely complex, even for bootstrapped founders - and responsibilities like <u>"building the team, designing the product, architecting the solution, and writing the code"</u> have all become separate disciplines that each require specific training and resources.

This means that it takes decades to master all of the skills required to build the right solutions, the right way, with the right people... and this means that no single person can actually handle all of these responsibilities at the same time while remaining cost-effective.

Said differently, software developers stop being cost-effective software developers the moment they stop developing software full-time.

Still don't believe me? Just take a look at the following list of responsibilities and tell me with a straight face that a single CTO, VP of Engineering, Technical Manager, or Technical Co-founder should handle all of them:

- External Relations (vendors, investors, partners)
- Internal Education (within dept. & cross dept.)
- Expectation Management (within dept. & cross dept.)
- ROI Assessment (build vs. buy)
- Resource Development (budget & personnel)
- Career Planning/Upward Mobility
- Team Composition & Role Definition
- Goal Identification & Communication
- Culture Definition
- Strategic Risk Identification & Mitigation
- Overall Process Specification (gap identification)
- Personnel Management (one-on-one's)
- Productivity/Velocity Management
- Conflict Resolution
- Resource Allocation
- Recruiting
- Onboarding
- Culture Implementation
- Automated Test Strategy & Architecture
- Overall Code Architecture (scalability & fault tolerance)
- Overall Code Quality (design patterns)

- Technical Component Selection ("tech stack")
- Technical Risk Identification & Mitigation
- DB Schema & Data Architecture
- Code Style Definition
- Product Roadmap Management (definition & prioritization)
- Customer Need Identification
- Network Performance
- Network Security
- Resource Budgeting & Planning
- CI/CD Implementation
- Code Style Enforcement
- Large Feature Architecture
- Day-to-day Code Reviews
- Test Implementation
- Feature Implementation
- A/B Test Implementation & Analysis
- Product Requirements Gathering
- Process Requirements Gathering
- Process Implementation & Documentation
- Release Management
- Wiki Management
- Sprint Management
- QA Testing
- UX Design
- UI Design

Yup, that's right. ALL of these skills, and more, are required to build even the smallest of projects!

Additionally, even *if* a single person was somehow able to cultivate strong leadership skills while managing to keep their coding practice in tip-top shape, it's actually impossible for them to effectively switch between producer and manager roles often.
This article illustrates the hidden cost of switching roles regularly (*https://www.petrikainulainen.net/software-development/processes/the-cost-of-context-switching/*).

Further, even *if* a single person could magically code, *and* lead, and be cost-effective at both at the same time; it's also impossible for a technical manager to build a cost-effective team when they can't consider hiring developers that work with technologies that they are not familiar with, or consider hiring developers that might do their job more effectively than they can.

All of this means that you must systematically strip responsibilities from generalists so that they can focus on doing more of whatever it is that they do well. Chapter VIII - Structuring Your Team will explain how to hire cost-effective specialists which support all of the generalists on your team so that everyone can focus on the things that they do best.

***Key Concept**: "CTOs that code" Accomplish Less Than Technical Co-founders Paired With Technical Advisors - And Build Products That Are Poorly Positioned For The Market*

Just to be clear, it is a GOOD IDEA to hire a CTO, VP of Engineering, Technical Manager, or a Technical Co-founder, and it is okay to provide your teammates with growth opportunities to learn how to do things that they have not done before… but it's important that you:

1. Expect generalists to have weakness
2. Support generalists by hiring specialists that can shore up their weaknesses
3. Do not overwhelm generalists will so many responsibilities that they are no longer able to be cost-effective at any of them

—

CASE STUDY (7 of 8)
Developers Usually Are Not Trained To Lead

A founder recently hired me to interview her CTO (Evan), who she had been struggling to manage for over six months.

And, when I spoke with Evan, he told me that he was frustrated with the founder because she wasn't sticking to a vision that he could follow.

That's when I asked him if he found it challenging to document their conversations, and he said "no, that's her job".

That response was all I needed to hear. It meant that Evan wasn't managing the founders' expectations - a primary responsibility of an an actual CTO.

Evan was clearly a developer, not an experienced leader.

—

NOTE: *This problem isn't just an early-stage problem. Chapter VIII - Structuring Your Team will look at another example of an overwhelmed CTO struggling to lead a Billion dollar company.*

SUMMARY

Technical Co-founders, "CTOs That Code", and other technical generalists lack the perspective and experience to shape your:

- Code
- Products

- Processes
- Team
- *And* culture

But, they can be very effective partners when supervised by an experienced CTO, Advising CTO, and/or Technical Advisor.

***NOTE**: Visit Bonus #1 :: Signs Of Trouble to identify the top signals which indicate that your technical leader is failing. Also, find a comprehensive self assessment framework with over 300 questions at https://www.jaycrouch.com/assessment which you can use to check how well your technical team is doing.*

PART III
GROWTH

And now, it’s time to put everything together to scale a technical project from start to finish.

We will begin by selecting the right technical partners to build an MVP given your budget and requirements, then work through how to structure your team, and then illustrate how to apply the formula we’ve learned to large organizations.

CHAPTER VII
WORKING WITH TINY BUDGETS

This chapter will describe five ways to work with internal staff, contractors, and agencies in order to maximize your ROI when working with very small budgets.

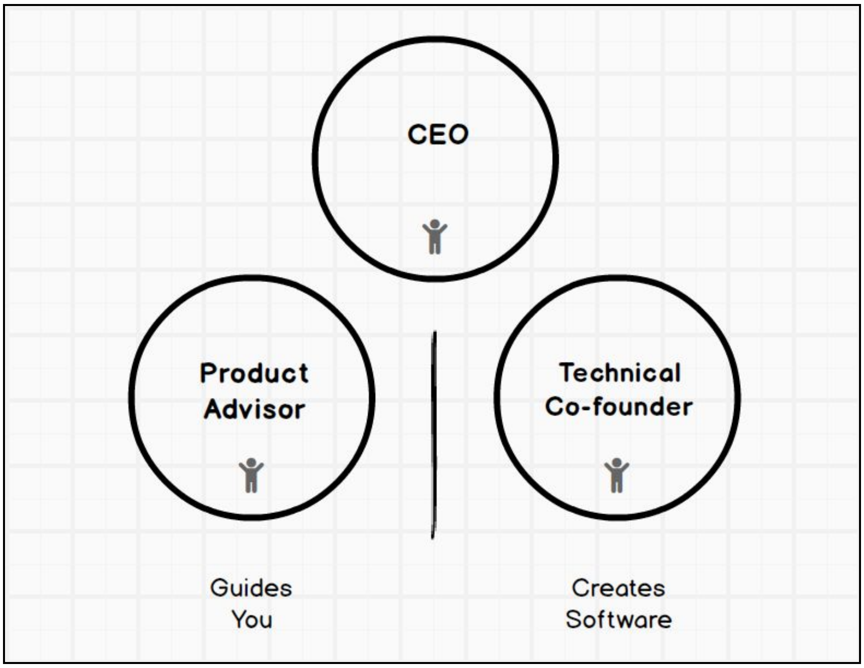
The most efficient configuration requires at least $12,000/mo - but there are many good options available for leaders who lack gobs of cash.

OPTION A) MINIMUM CASH BURN (~ $1,000/MO - $4,000/MO)

The smallest and cheapest team includes a CEO, a Product Advisor, and an equity-based developer.

Find $1,000/mo and give it to a Product Advisor, then recruit an equity partner to be your Technical Co-founder.

Also, don't bother trying to recruit a senior developer as a Technical Co-founder. Senior technologists don't need to work for equity. Focus on recruiting a Jr. to mid-level developer and support him/her with experienced advisors and coaches.



Keep in mind that $1,000/month won't buy you a lot of time with an expert advisor, but it will buy you all the time that you need. Remember that your only concern at this stage should be verifying that the product you want to build is worth building.

Don't build a mobile app when a web app will do, and don't build software at all if you can get away with using other mechanisms to gather feedback (like mockups).

Also, try to find someone that works in the Product department at a successful company to look over the recommendations that your Product

Advisor makes in order to ensure that their recommendations are sound.

OPTION B) OPTIMAL TEAM (~ $12,000/MO - $25,000/MO)

Build a team of disposable contractors around a full time Lead Developer once you have solid funding and are ready to get the most bang for your buck.

Your Lead Developer (or Technical Co-founder) doesn't *have* to sit in the office with you, but it helps. Also, it's best to work with contractors because your needs are likely to change often, and swapping out W2 based engineers is very costly from both a financial and cultural perspective.

There are some very capable people in Tier 3 markets like Florida, Utah, and Illinois; and the cost savings can be substantial when compared to Tier 1 markets like San Francisco and New York (as well as Tier 2 markets like Austin, Los Angeles, and Seattle.)

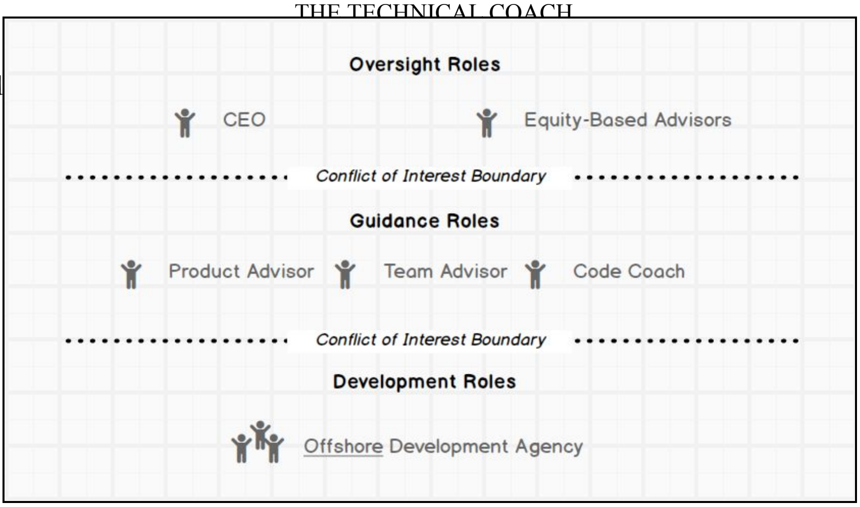
Just so you understand, the same developer costs 30% more in a Tier 2 market than a Tier 3 market, and an additional 30% more in a Tier 1 market. Here's a breakdown of rates in several software developer markets *(http://www.businessinsider.com/cities-us-software-engineer-ranked-salary-cost-living-2015-6)*.

Keep in mind the numbers in this report aren't reflective of the full range of salaries in these areas - top developers make top dollar in all markets.

OPTION C) OFFSHORE AGENCY (~ $2,000/MO - 5,000/MO)

Skip hiring any and all employees by working with an offshore staffing agency when you only have a small MVP to build and don't want to recruit an equity-based Technical Co-founder.

Offshore staffing agencies typically cost 30% less than Tier 3 domestic d



Remember, your Product and Team Advisors should work together to help you reduce the size and cost of your MVP, and a domestic Code Coach should supervise the work that your offshore firm creates.

Alternatively, you can hire a local Project Manager and work with independent offshore developers… but this is more complicated.

The cheapest Ukrainian firms will supply "senior developers" for $25/hr, but you'll pay separately for project management and other support, and "senior" means different things to different people… I generally consider an offshore senior developer equivalent to a domestic entry-level developer in a Tier 1 market.

I usually recommend paying top dollar for the best engineers you can find (~$40-50/hr) because they tend to be much stronger with spoken english and far better at expectation management.

Alternatively, many offshore development firms have management offices in the US which can make communication and time zone differences easier to manage - but you'll pay for this service. Offshore firms with domestic offices average $80-$120/hr -- which is basically equivalent to the rates of most domestic mid-level contractors.

Also, understand that people who live in other countries have different cultural values. For example, many cultures value "projecting confidence" much more than people in the United States. Because of this, you typically have to micromanage offshore resources.

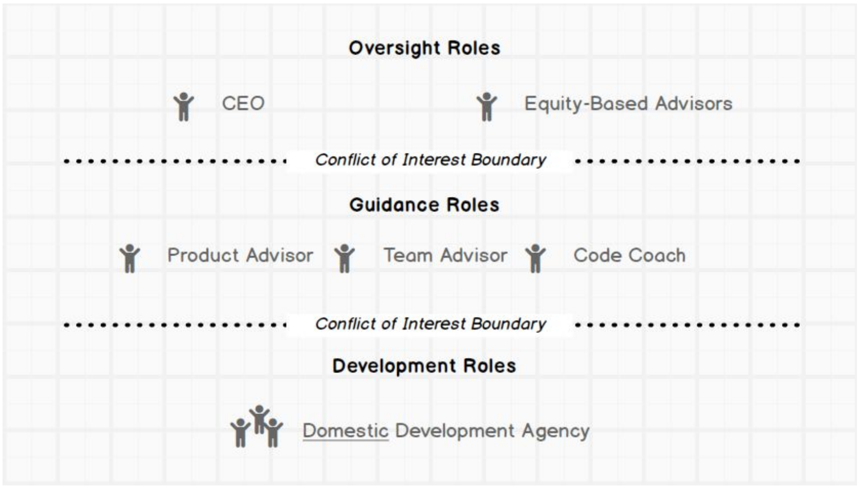
I'm making a wide sweeping generalization here, but if you ask the typical Indian developer if they can do something, the answer will usually be "yes". And, if you ask them to do it faster, the answer will usually be "OK". And, then they will fail silently until the deadline passes and you go looking for answers.

In contrast, Eastern Europeans will be much more likely to push back, work hard, and raise a white flag when they get into trouble. South Americans tend to be the easiest to communicate with because the time zones are closest to US time zones - but they also tend to not work as hard as Eastern Europeans (again, generally speaking).

***NOTE**: Never attempt to work with a firm (offshore or domestic) without the support of an advisor. This is the fasted way to torch your cash and get locked into a development firm because no one else will be able to work the code that they produce.*

OPTION D) DOMESTIC FIRM (~ $25,000/MO+)

Work directly with an domestic firm when you need to build a really big project, really fast - and you don't want to retain an internal team afterwards.



Remember, domestic firms are never cost effective, but they are easy.

Hourly equivalents for domestic development firms vary *widely*. Firms in Tier 3 markets often compete with offshore firms using a blended rate ~ $100/hr (+/-). Rates for Tier 1 firms typically jump up somewhere between $150 and $300/hr.

That said, you will recover a small amount of this money because you will allocate less to independent advisors and coaches when working with experienced domestic firms who have experienced Software Architects and Product Managers on staff.

Remember, you will still need all the independent parties described in this guide to keep everyone in check, but you won't need to spend as much time with your advisors and coaches when working with domestic firms.

OPTION E) INTERNAL TEAM (~ $25,000/MO+)

Recruit an army of software developers to support your Lead Developer when you absolutely *know* that you have a long term project ahead of you and enough budget to retain your team.

Internal teams are significantly more complex to build than all other options. You'll need Project Managers, Product Managers, QA Engineers, Solutions Architects, Devops Engineers, and Designers.

Chapter VIII - Structuring Your Team will explain how to build an internal team, and Bonus #2 :: Relating To The People On Your Team will explain what all these people do.

***NOTE**: Never attempt to build an internal team until you have established product-market fit and have a stable cash supply (see https://en.wikipedia.org/wiki/Product/market_fit).*

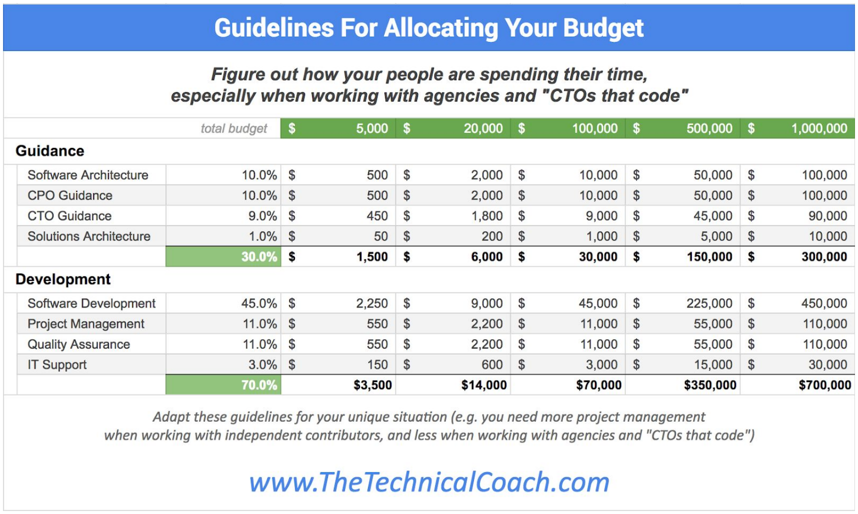
BY THE NUMBERS - WHAT EVERYTHING SHOULD COST

No matter how you configure your team, your budget should look something like the figure below. Try to get your total spend for advisors and coaches under 30% of your total budget, if possible.

This is only a very rough guideline, please keep in mind that every organization has different needs.

It's perfectly okay to ask a developer, development firm, or team of developers, to project manage things themselves and do their own testing. These activities do not create any conflicts of interest.

You may need to spend more on you coaches if you need a lot of help refining your idea, or fixing a team that wasn't set up well to begin with, or want things handled for you.



Guidelines For Allocating Your Budget

Figure out how your people are spending their time, especially when working with agencies and "CTOs that code"

	total budget	$ 5,000	$ 20,000	$ 100,000	$ 500,000	$ 1,000,000
Guidance						
Software Architecture	10.0%	$ 500	$ 2,000	$ 10,000	$ 50,000	$ 100,000
CPO Guidance	10.0%	$ 500	$ 2,000	$ 10,000	$ 50,000	$ 100,000
CTO Guidance	9.0%	$ 450	$ 1,800	$ 9,000	$ 45,000	$ 90,000
Solutions Architecture	1.0%	$ 50	$ 200	$ 1,000	$ 5,000	$ 10,000
	30.0%	**$ 1,500**	**$ 6,000**	**$ 30,000**	**$ 150,000**	**$ 300,000**
Development						
Software Development	45.0%	$ 2,250	$ 9,000	$ 45,000	$ 225,000	$ 450,000
Project Management	11.0%	$ 550	$ 2,200	$ 11,000	$ 55,000	$ 110,000
Quality Assurance	11.0%	$ 550	$ 2,200	$ 11,000	$ 55,000	$ 110,000
IT Support	3.0%	$ 150	$ 600	$ 3,000	$ 15,000	$ 30,000
	70.0%	**$3,500**	**$14,000**	**$70,000**	**$350,000**	**$700,000**

Adapt these guidelines for your unique situation (e.g. you need more project management when working with independent contributors, and less when working with agencies and "CTOs that code")

www.TheTechnicalCoach.com

SUMMARY

There are five ways to configure configure small teams with small budgets.

The most cost effective is not the cheapest, and it is comprised of a single Lead Developer that is surrounded by disposable contractors.

The cheapest option involves an equity-based developer.

CHAPTER VIII
STRUCTURING YOUR TEAM

This chapter will show you how to organize a powerful and cost-effective team; no matter how small it is, or how big it becomes.

Here's a simple way to go about organizing your technical team:

First	Identify every single kind of activity that your team needs to complete in order to be successful.
Second	Assign responsibility for each of these activities to the people on your team that are both cost-effective and qualified to handle them. *This concept is commonly referred to as "Role Optimization".*
Third	Hire part-time staff-members (or contractors) to handle all the activities that your existing staff members can't handle cost-effectively. *This concept is commonly referred to as "Fractional Role Coverage".*

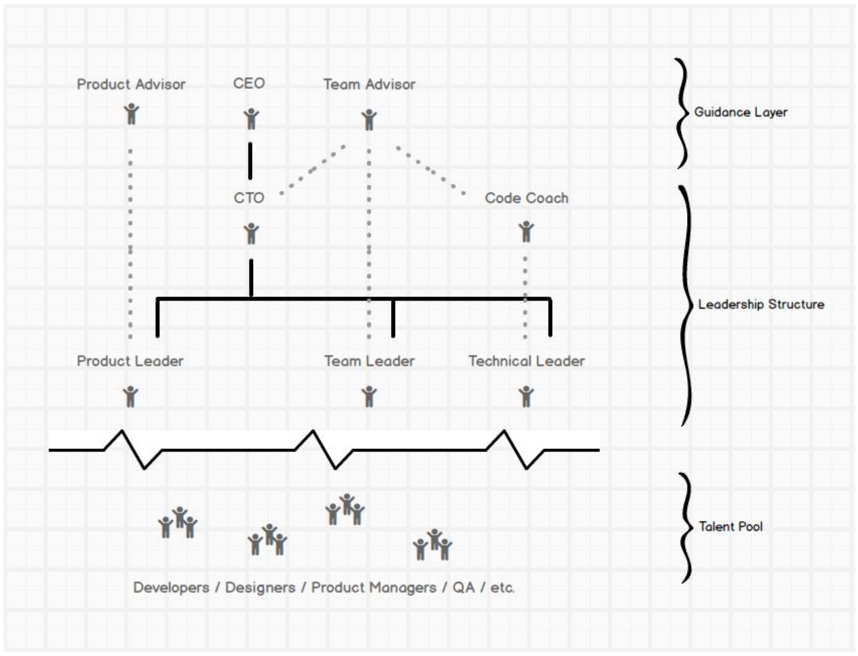
That it! That's all you have to do.

Hint: I guarantee that you'll end up defining at least four leadership rules when you perform this exercise.

THE ORG CHART FOR VISUAL THINKERS

The following diagram will help you understand the rest of this chapter.

It represents the basic leadership structure that almost *every* technical team should follow (*we'll cover the Talent Pool later in this chapter).*



> ***NOTE**: At a minimum, four leadership roles are required to run effective teams, each leadership role should be held by a single person, and no single person should hold more than one leadership role at the same time.*

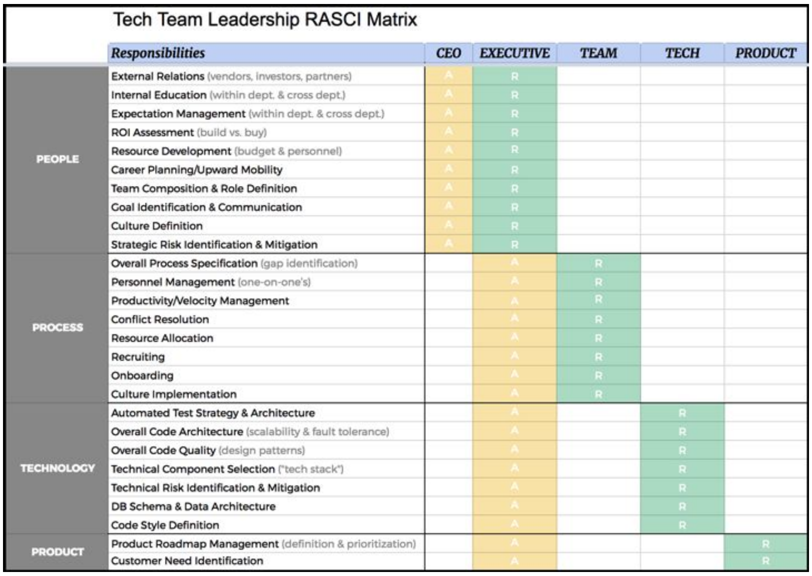
Remember – everyone on your team can be a part-time resource, if needed.

THE RASCI

To understand why at least four leadership roles are required, take a look at the result of a RASCI exercise that I recently performed for a client.

A RASCI is a simple exercise that helps you assign responsibilities to staff members. (see https://en.wikipedia.org/wiki/Responsibility_assignment_matrix - RASCI)

To perform a RASCI, just draw out a grid and list all the important activities that you need to handle on one side of a grid, then list all the roles that you have defined on the other side, and then fill in the blanks to designate which roles are (R)esponsible, (A)ccountable, (S)upporting, (C)onsulted, and (I)nformed for each activity.


Tech Team Leadership RASCI Matrix

	Responsibilities	CEO	EXECUTIVE	TEAM	TECH	PRODUCT
PEOPLE	**External Relations** (vendors, investors, partners)	A	R			
	Internal Education (within dept. & cross dept.)	A	R			
	Expectation Management (within dept. & cross dept.)	A	R			
	ROI Assessment (build vs. buy)	A	R			
	Resource Development (budget & personnel)	A	R			
	Career Planning/Upward Mobility	A	R			
	Team Composition & Role Definition	A	R			
	Goal Identification & Communication	A	R			
	Culture Definition	A	R			
	Strategic Risk Identification & Mitigation	A	R			
PROCESS	**Overall Process Specification** (gap identification)		A	R		
	Personnel Management (one-on-one's)		A	R		
	Productivity/Velocity Management		A	R		
	Conflict Resolution		A	R		
	Resource Allocation		A	R		
	Recruiting		A	R		
	Onboarding		A	R		
	Culture Implementation		A	R		
TECHNOLOGY	**Automated Test Strategy & Architecture**		A		R	
	Overall Code Architecture (scalability & fault tolerance)		A		R	
	Overall Code Quality (design patterns)		A		R	
	Technical Component Selection ("tech stack")		A		R	
	Technical Risk Identification & Mitigation		A		R	
	DB Schema & Data Architecture		A		R	
	Code Style Definition		A		R	
PRODUCT	**Product Roadmap Management** (definition & prioritization)		A			R
	Customer Need Identification		A			R

When you look at the sample above it should be clear that certain types of activities naturally group together:

- **Team Leaders** drive the performance of the entire team
- **Technology Leaders** architect the software
- **Product Leaders** optimize the ROI of the features which are built
- **Executive Leaders** manage relationships with external parties and internal stakeholders

It should also be clear that people who excel in each of these particular roles will be unqualified and costly when placed in any of the other roles because the skill sets required are so vastly different.

Also, know that the titles that you give your leaders are not as important as the roles that you define for them. That said, when possible, try to give your leaders titles which reflect their role, experience, and the number of direct reports that they manage. This helps everyone in the organization to know what they can expect from their peers without referring to the org chart on a daily basis.

For organizations with less than fifty engineers, Team Leaders are typically referred to as VPs of Engineering (or Directors), Product Leaders are typically referred to as VPs of Product (or Directors), and Technology Leaders are generally referred to as Software Architects, Chief Software Architects, or Enterprise Architects.

—

CASE STUDY (8 of 8)

CTOs Need To Scale With The Companies That They Support

Most engineers aspire to be Rudy one day.

He is a great CTO, and he's one of the most caring and dedicated technologists that I've ever known.

But, what makes Rudy interesting is that he recently led a startup that ended up blowing up all over the news in a great way… one of those super successful companies that we all hope to be part of. And, while leading this team, some of Rudy's decisions fell short.

Despite being extremely dedicated and capable, Rudy struggled with a couple of critical things.

First, Rudy struggled with the selection of a critical technology component and ended up caving in to pressure from another department to pick an older 'proven' product instead of using newer platforms which offered easier integration and less need for external consultants.

Unfortunately, the older platform ended up costing at least double the original estimates, both in terms of time & money. This was a ***multi-million*** *dollar mistake that could have been easily avoided.*

Second, Rudy routinely prioritized new feature development over code quality, because the company was growing at a breakneck pace. However, this allowed a significant amount of technical debt to build up, and this eventually slowed down progress significantly.

The end of this story is not pretty: the delays and increased costs were so significant that the business shifted its focus from subscription based ecommerce to retail distribution… and dismantled it's technical team of 60 people.

But through all of this, Rudy and his team were not the problem.

The problem was that Rudy had not previously been the CTO of a company growing so fast; and the CEO never appointed a Technical Advisor, Advising CTO, or Coach to help him shore up his weaknesses.

—

***NOTE**: Most small teams struggle to assign appropriate titles because they give the first engineer that joins the team the title of CTO. This is usually okay until the team raises a bit amount of money and begins to hire engineers.*

That's when it usually becomes evident that the CTO is not ready to handle executive leadership responsibilities, and really needs to be transitioned to VPP, VPE, or CSA -- or at the very least, actively supported by an Advising CTO -- a delicate issue that is often difficult to navigate. Sometimes, teams handle this transition by changing the role and responsibilities without changing the title, and I usually recommend against this because it causes a great deal of confusion within the team.

I recommend educating your early stage CTO that they will most likely need to transition to Chief Architect, or VP of Product, when the team grows.

After the initial shock to their ego, these engineers usually realize that they would actually prefer either of these roles because they don't involve active personnel management, and are usually surprised to find that they can actually earn just as much personal income as an Architect as they can as a CTO (sometimes even more).

UNDERSTANDING THE EXECUTIVE LEADERSHIP ROLE

Let's stop and take a special look at the Executive Leader role before examining the labor pool structure.

The executive leadership role is a special role because it doesn't *have* to be filled by a highly technical person, unlike the other three leadership roles.

For example, CEOs of highly technical startups typically decide to manage the Product, Team, and Technology leaders themselves because the entire business is built around the technology team. These CEOs typically represent the technical team to investors and clients, cultivate vendor relationships, and manage expectations with all the other stakeholders in the organization. This means that these *CEOs* typically act as the Executive Leader for the technical team as defined above (otherwise known as "CTO").

It's extremely critical that these types of CEOs hire Advising CTOs (extremely hands-on Technical Advisors) to teach them how to be effective CTOs on an ongoing basis… because everything rides on them.

On the other hand, CEOs of non-technical organizations typically want as little interaction with the technology department as little as possible. This means that non-technical organizations typically need to take one of the following approaches:

a) Ask the Product, Team, and Technology leaders to report to another member of the executive team (President/COO/CMO), and hire an Advising CTO to help that person inspect all the things that they can not inspect themselves, and support all the people they can not support themselves...

or

b) Hire an experienced Consulting, Acting, or Interim CTO to manage the technical leaders under the supervision of a hands-off Technical Advisor. This is one of the few situations where Consulting CTOs actually make a lot of sense.

THE TALENT POOL

The best structure for your Talent Pool is, well, left up to you to decide. The fact is that there are many, many different way to put developers, quality assurance engineers, product managers, and designers together to create software.

That said, most organizations today are using a concept called a "pod".

Pods are cross-functional teams that typically include:

1. A Lead Developer

2. Three to five additional Developers
3. A UI (or UX) Designer
4. A Product Manager
5. And, a QA Engineer

***NOTE**: Visit* Bonus #2 :: Relating To The People On Your Team *to understand more about what developers, quality assurance engineers, product managers, and designers do for you.*

The key thing to understand about pods is that pod members typically don't report to a single manager. Rather, each pod member reports to a functional manager. Engineers typically report to Team Leaders, UI/UX Designers and Product Managers report to Product Leaders, and QA specialists report to either the Team or Product Leader.

However, occasionally, Product Leaders and/or Technical leaders also directly manage entire pods. These teams are typically responsible for innovation and infrastructure, respectively.

Also, of note, very few people other than the infrastructure architects ever report to the Technical Leader. Instead, the Technical Leader influences everyone indirectly. This is why they are typically referred to as "Architects". More information about pods can be found here: *(https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/)*.

THE GLUE LAYER

Mid-level managers are also typically required for teams that have grown so large that VPs are no longer able to interact with all of their reports directly.

Generally speaking, most leaders can only effectively manage five to ten people, so mid-level managers start showing up when teams start exceeding this size.

However, there is a huge difference mid-level managers (Directors or Technical Managers) and VPs. Directors typically have very little experience managing other managers (indirect management), and it typically takes people years to figure out how to do this effectively. In fact, many mid-level manager never figure out how to do this well.

This is why most organizational designers recommend that you should hire VPs from outside the organization, rather than attempting to cultivate them from within.

***NOTE**: Some very sophisticated organizations have started moving away from strictly hierarchical structures because the Glue Layer can become extremely costly, and inefficient.*
However, these organizations aggressively use data driven toolkits to manage their advanced structures. I do not recommend attempting to deviate from a hierarchical structure unless your organization is truly dedicated to investing millions of dollars into the development of new organizational structures.

THE FINAL PIECES

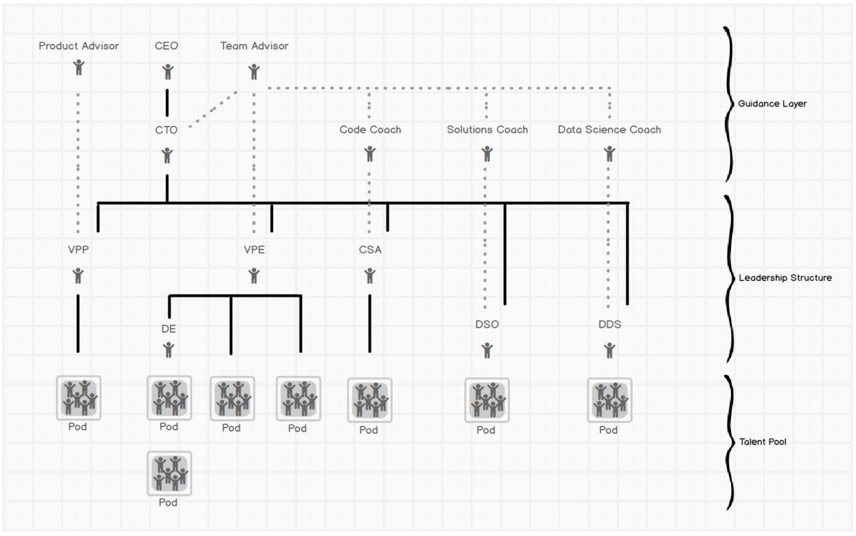
While building a healthy technical organization, you'll probably identify several miscellaneous activities that otherwise support the main mission but don't contribute directly to the engineering effort.

For example, most organizations need security, IT and "devops" support, and these functions typically need to fall under a fifth leader that I have omitted up to this point for simplicity - the Solutions Leader.

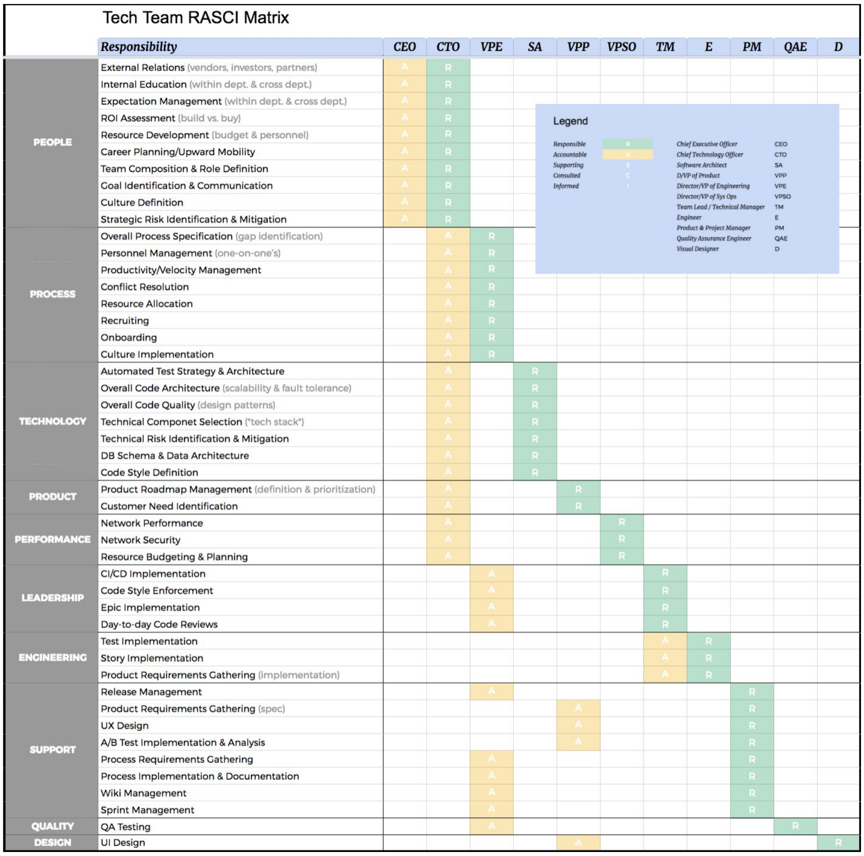
Additionally, many organizations are finding that they need to develop a few other functions, such as data science, graphics, and hardware. Each of these disciplines require leadership from a dedicated leader with domain expertise, just like the others we have talked about in this chapter.

And, of course, I recommend hiring specialized advisors and coaches to help you inspect and support every additional leader that you hire if that function is critical to your business… because you won't know what they do on a day-to-day basis any more than you do anyone else on your team.

Here is a real Org Chart for a company with about 50 engineers:



… and the corresponding RASCI sample:



Tech Team RASCI Matrix

	Responsibility	*CEO*	*CTO*	*VPE*	*SA*	*VPP*	*VPSO*	*TM*	*E*	*PM*	*QAE*	*D*
PEOPLE	External Relations (vendors, investors, partners)	A	R									
	Internal Education (within dept. & cross dept.)	A	R									
	Expectation Management (within dept. & cross dept.)	A	R									
	ROI Assessment (build vs. buy)	A	R									
	Resource Development (budget & personnel)	A	R									
	Career Planning/Upward Mobility	A	R									
	Team Composition & Role Definition	A	R									
	Goal Identification & Communication	A	R									
	Culture Definition	A	R									
	Strategic Risk Identification & Mitigation	A	R									
PROCESS	Overall Process Specification (gap identification)		A	R								
	Personnel Management (one-on-one's)		A	R								
	Productivity/Velocity Management		A	R								
	Conflict Resolution		A	R								
	Resource Allocation		A	R								
	Recruiting		A	R								
	Onboarding		A	R								
	Culture Implementation		A	R								
TECHNOLOGY	Automated Test Strategy & Architecture		A		R							
	Overall Code Architecture (scalability & fault tolerance)		A		R							
	Overall Code Quality (design patterns)		A		R							
	Technical Componet Selection ("tech stack")		A		R							
	Technical Risk Identification & Mitigation		A		R							
	DB Schema & Data Architecture		A		R							
	Code Style Definition		A		R							
PRODUCT	Product Roadmap Management (definition & prioritization)		A			R						
	Customer Need Identification		A			R						
PERFORMANCE	Network Performance		A				R					
	Network Security		A				R					
	Resource Budgeting & Planning		A				R					
LEADERSHIP	CI/CD Implementation			A				R				
	Code Style Enforcement			A				R				
	Epic Implementation			A				R				
	Day-to-day Code Reviews			A				R				
ENGINEERING	Test Implementation							A	R			
	Story Implementation							A	R			
	Product Requirements Gathering (implementation)							A	R			
SUPPORT	Release Management			A						R		
	Product Requirements Gathering (spec)					A				R		
	UX Design					A				R		
	A/B Test Implementation & Analysis					A				R		
	Process Requirements Gathering			A						R		
	Process Implementation & Documentation			A						R		
	Wiki Management			A						R		
	Sprint Management			A						R		
QUALITY	QA Testing			A							R	
DESIGN	UI Design					A						R

Legend

Responsible	R	*Chief Executive Officer*	CEO
Accountable	A	*Chief Technology Officer*	CTO
Supporting	S	*Software Architect*	SA
Consulted	C	*D/VP of Product*	VPP
Informed	I	*Director/VP of Engineering*	VPE
		Director/VP of Sys Ops	VPSO
		Team Lead / Technical Manager	TM
		Engineer	E
		Product & Project Manager	PM
		Quality Assurance Engineer	QAE
		Visual Designer	D

SUMMARY

All organizations need at least four leaders, although each can be a part-time staff member, or contractor… and the executive leadership role can be executed by a hands-on CEO.

Most organizations have special needs, and require additional leaders (and additional advisors to oversee these leaders).

Advising CTOs can be powerful and cost-effective allies for CEOs that want to stay close to their teams, or want to avoid the overhead associated with hiring full-time executive leaders.

The first technical partner that you hire is extremely unlikely to be capable of handling the requirements of the Team Leader role once the team size grows beyond approximately ten engineers, and this person is almost guaranteed to be unable to handle the requirements of the Executive Leader role.

Listen to your people and help them to organize themselves under qualified and focused leadership.

CHAPTER IX
DRIVING ENTERPRISE PERFORMANCE

This final chapter is essentially a call to action for senior managers of large organizations that are looking for ways to holistically improve the performance of their organizations.

Consider hiring 3rd party coaches to assist the staff who work at all levels of your organization, in all departments of your organization, not just your highest ranking technology officers.

I'm not talking about generic leadership or personal coaching, I mean domain specific skill coaching. Use coaches to free up both your mid-level managers and your senior executives to drive performance by tasking coaches with difficult assignments like developing your people and ensuring that the best ideas receive the attention that they deserve.

At the very least, hire coaches to support every function that drives a KPI.

COACHES ARE NOT A NEW IDEA

Fortune 500 CEOs often hire consulting firms like Mckinsey, Bain, and BCG because they know that the people on their payroll are mired in political battles and have limited perspectives about the overall business. These firms work to check and balance the organizations that they serve, and they often make billion dollar improvements.

But, the experience can be jarring when consulting firms are brought in to fix problems after they have already erupted. Staff are often “let go”, and this creates a hostile relationship between staff and consultants, to say the least.

If a $1,000/mo Code Coach can save a founder hundreds of thousands of dollars each and every year by offering guidance on a long-term basis, just imagine what coaches could accomplish when applied across your entire organization.

COACHES EXTEND THE LONGEVITY OF EVERYONE ON YOUR PAYROLL

Fast growing organizations often outpace their best leaders.

Coaches can help alleviate this problem by helping your staff to grow alongside your organization - increasing their performance and longevity.

By way of example, CTOs of high growth companies often lose effectiveness after 18 months. The reason is that great CTOs tragically help grow organizations to a size they no longer know how to effectively manage. But, advisors and coaches can look ahead and help CTOs to identify personal weaknesses and opportunities to grow with the company.

If they can do this for CTOs, think about what they can do for mid-level managers.

Remember that if your CTO doesn't have the time or skills to cultivate the staff working for him/her, your middle managers probably don't have time or skills to cultivate their staff either.

COACHES HELP MITIGATE POLITICAL ENVIRONMENTS

Many executives assume checks and balances naturally emerge as their organizations grow, especially if they make themselves highly available and check in with their teams often.

In reality, this is only partially true.

Yes, your staff members can help each other avoid mistakes and call attention to bad decisions. But, every organization I've ever encountered has both a chain of command and politics to contend with.

This means you can't rely on your staff to tell you the whole truth because everyone that works for you has their own agendas, limitations, and myopic views of the business. You can't rely on your equity advisors, because equity advisors of large companies sure as hell don't have time for hands-on work. And, you can't rely on your own inspections because you don't know which tires to kick.

This is why advisors and coaches matter from the day you start your business to the day you sell it. They provide you an independent voice free of political considerations by which to inspect and respond with.

And that means the people in your organization will be more likely to focus on achieving results than promoting themselves.

SUMMARY

Consider working with 3rd party advisors and coaches across all high risk, entrepreneurial, and growth arms of your business.

Consider hiring coaches to assist your staff at all levels of the organization, not just your highest ranking officers.

Consider that political pressures might be preventing your well-meaning staff from being as effective as they could be.

BONUS #1
SIGNS OF TROUBLE

A young entrepreneur recently asked me a question that caught me off guard:

"When do I need a CTO? What signals tell me something is wrong?"

It wasn't so much that he didn't yet have a CTO as it was that he didn't know how to tell the difference between great performance and poor performance.

NOTE: *Find a comprehensive self assessment framework with over 300 questions at www.jaycrouch.com/assessment to see how well your technical team is doing.*

Take the following checklist with you as you build your business. Every business has flaws, but a storm is brewing when you notice the following signs:

Executive Incompetence

Your team doesn't communicate, manage expectations or make themselves available to speak as often as you need

1. Your technical staff are unable to meet deadlines
2. Your non-tech staff are increasingly frustrated with tech staff
3. Your IT department is gripped with low moral & political struggles
4. You are struggling to recruit quality developers
5. Your tech staff have never been surveyed for engagement, or lack it

Technical Mistakes

Your code is preventing progress

6. You do not have automated tests that cover at least 70% of your codebase
7. You are locked into working with the developers who built your system
8. You don't know if you have bad developers and poor architecture
9. Your developers struggle to balance cost, speed and quality trade-offs
10. More (or less) than 20% of your budget is consistently allocated to resolving tech debt
11. You have old and dated infrastructure

Product Mistakes

Your team isn't building features that users and investors care about

12. Your consumers fail to engage with your products after signing up
13. Your costs for user acquisition and technology prevent growth

14. Your sales and marketing teams struggle to interest and retain customers

Organizational Mistakes

Your processes are wonky or you aren't able to deploy features quickly

15. You core systems are missing: product roadmap, project management system and product management system
16. Your core processes are not captured with documentation
17. Your technical staff fail to provide estimates or irregularly measure velocity (week-over-week productivity for small, medium and large features)
18. Your technical team, product and codebase haven't received an external audit in two years or longer

Get Senior Help Onboard Before S#!t Hits the Fan

Would you open a restaurant without knowing if the food tastes good?

Of course not.

In the end, the issues that create the most cause for concern are those you won't see until it's too late:

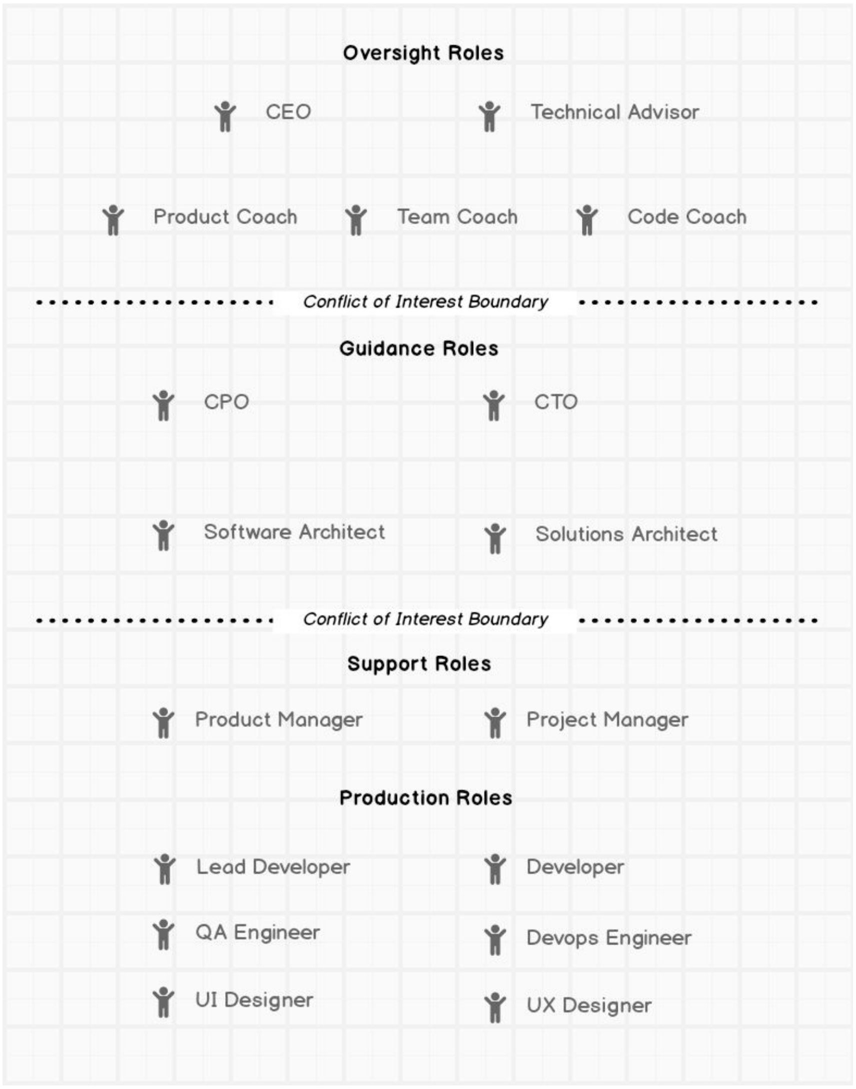
Lack of confidence in what's under the hood
- and -
Lack of progress while your competition surges ahead

BONUS #2
RELATING TO THE PEOPLE ON YOUR TEAM

The roles we've talked about in this guide are only some of the roles in modern software teams. Nowadays, there is a whole cast of supporting characters that keep everyone honest and performing at their best.

As your team grows it will begin to look like a fully functional technical department. At first, many of the roles will be filled by contractors, or handled by firms.

Pain and disaster await tech teams without full role coverage. Every team needs someone in each of the following roles:


Oversight Roles
CEO
Technical Advisor
Product Coach
Team Coach
Code Coach
Conflict of Interest Boundary
Guidance Roles
CPO
CTO
Software Architect
Solutions Architect
Conflict of Interest Boundary
Support Roles
Product Manager
Project Manager
Production Roles
Lead Developer
Developer
QA Engineer
Devops Engineer
UI Designer
UX Designer

Roles In Modern Tech Teams (*and their football team equivalents*)

1.	CTO	*Coach*
2.	CPO / VP Product	*Offensive Coordinator*
3.	Software Architect	*Defensive Coordinator*
4.	Solutions Architect	*Special Teams Coordinator*
5.	Product Manager	*Scout*
6.	Project Manager	*Medic*
7.	Lead Developer	*Quarterback*
8.	Developer	*Linemen*
9.	DevOps Engineer	*Kicker*
10.	QA Engineer	*Fullback*
11.	UI Designer	*Waterboy*
12.	UX Designer	*Powerade-Sponsored Waterboy*

And yes, ***you can afford everyone you need regardless of your budget***.

Role #1 - CTO (aka Coach - *not to be confused with Technical Coach*)

CTOs build and maintain the team. They work with people and processes. CTOs have deep technical knowledge but should NOT be the most technical person in the room.

Works with: everyone

Primary Duties/Skills:

- Budgeting
- Recruiting
- Educating
- Negotiating
- Mitigate risk
- Refine Processes
- Implementing Culture
- Managing Expectations

Role #2 - CPO / VP Product (aka Offensive Coordinator)

CPOs research, analyze and design: products, markets, user behavior and features. The CPO has decades of experience in Sales, Marketing, Technology and Operations.

Works with: C-Suite, Product Manager, Project Manager, Lead Developer

Primary Duties/Skills:

- Ensuring Alignment Company-Wide
- Applying The WHY That Drives The Company
- Attracting Users
- Driving Revenue
- Maximizing ROI
- Increasing User Engagement

Role #3 - Software Architect (aka Defensive Coordinator)

Architects have decades of experience with all the technologies used; are very expensive, and should be used sparingly.

Works with: CTO, Lead Developer, Developers

Primary Duties/Skills: Conduct code reviews to ensure code quality is high, design choices utilize best practices and the overall system is robust and scalable.

Role #4 - Solutions Architect (aka Special Teams Coordinator)

Unlike Architects, Solutions Architects deal with servers and "IT", not code.

Solutions Architects also have decades of experience; are very expensive, and should be used sparingly.

Works with: CTO, Lead Developer, Devops developers

Primary Duties/Skills: Audit infrastructure built by devops developers to ensure security, redundancy and availability.

Role #5 - Product Manager (aka Scout)

Project Managers are extensions of the CPO. They coordinate research projects and the design of the UI/UX.

Works with: everyone

Primary Duties/Skills: Manage the day-to-day prioritization of the product roadmap and user experience

Role #6 - Project Manager (aka Medic)

Project Managers are extensions of the CTO. They ensure that all stakeholders have all the information necessary to do their jobs.

UNLIKE PROJECT MANAGERS IN OTHER DISCIPLINES, PROJECT MANAGERS IN SOFTWARE DEVELOPMENT SHOULD NOT BE HELD RESPONSIBLE FOR ON TIME DELIVERY - THAT IS THE CTO'S JOB.

Works with: everyone

Primary Duties/Skills: Document requirements and decisions, steer meetings, and track down information

Role #7 - Lead Developer (aka Quarterback)

Lead Developer turns specifications and requirements into software. They are able to tackle any programming challenge, and are effective at delegating programming tasks to other developers.

Leads have less experience than Architects (5-7 years), but are more effective at working with people, and are more cost effective when implementing code.

Works with: CTO, Architect, Project Manager, Product Manager, Developers

Primary Duties/Skills: Code and stitch other developer's code together

Role #8 - Developer (aka Linemen)

Developers are day to day programmers with limited experience (< 5 years).

Works with: Lead Developer, Project Manager

Primary Duties/Skills: Code tasks delegated by the Lead Developer

Role #9 - DevOps Engineer (aka Kicker)

Devops engineers are your "IT" staff.

Works with: everyone

Primary Duties/Skills: Keep servers and services online, secure and available 24/7

Role #10 - QA Engineer (aka Fullback)

QA engineers test developers' code. They absorb repetitive and mundane tasks so that developers can focus on coding.

Works with: Developers, Lead Developer

Primary Duties/Skills: Test releases for bugs and consistency

Role #11 - UI Designer (aka Waterboy)

UI Designers create the colorful and appealing designs that bring to life the message and brand defined by your CMO which developers then build.

Works with: Product Manager, Project Manager, CPO

Primary Duties/Skills: Create designs

Role #12 - UX Designer (aka Powerade-Sponsored Waterboy)

UX Designers (often hybrid UX/UI Designers) also create designs that developers build, but do with concern for *how* users will interact with the designs. UX Designers are usually more expensive/experienced than UI designers, and not all designs require the UX skill set.

Works with: Product Manager, Project Manager, CPO

Primary Duties/Skills: Create designs that perform

SPREAD THE WORD

All boats will rise if leaders have the tools that they need to make better choices.

Please share this message if you agree with it.

Challenge your friends, investors and coworkers to rethink their approach.

And, if you have something that you'd like to add - please email me at or connect@JayCrouch.com or contact me through my blog: JayCrouch.com.

ACKNOWLEDGMENTS

Dedicated to: John, Sue, Sami, Casey, Mazal, Dariush, Justin, Salar, Norma, Joel, Ariel, Chris, Dean, Nick, Sean, Ben, Ava, Michael, Meg, Espree, Anthony, Daniel, Tony, Oleg, Amy,

and

my friends in the LA CTO Forum
-- thank you for supporting me.