Introduction

The “Insider’s Guide to Technical Management ” details some of the expectations we have of our technical managers that might be different than other companies. Whether new to Company or new to the manager role, the guidance that follows will help you understand what we value and how to be effective.

Company Mission

What is company mission?

Company Leadership Principles

Add company leadership principals

Guidance for New Tech Managers

Technical Managers at Company wear many hats, which is to say they are granted significant ownership over their technology area. Not only are they expected to drive their business like an owner, deliver exceptional solutions, and guide the careers of their staff; they are also expected to dive deep into their systems, be a voice for their customer, and focus on operational excellence. Before we get into all of these expectations, one thing must be stated upfront.

The quality of your engineering team will make or break you at Company.

Management in any area starts with your people. As this is not a management 101 guide, we are not going to get into *how* to manage people. There are plenty of books and training courses on that subject (see a list of recommendations in the appendix). We will say that the best managers at Company are deeply invested in their teams. They don’t just operate at high levels. They are connected to the details and *technically* interested in what their team is doing.

Hire the best. Get to know your team, their interests and career goals. Challenge and motivate them. Remove any obstacles that hinder their progress. Help your engineers become great.

Do this and embrace the expectations that follow and you will be a successful Technical Manager at Company.

Work backwards, starting with the customer

Means we prioritize what is most important to the customer; eliminating as waste all activities that the customer would not find value in or be willing to pay for. To help us do this, **we work backwards** – that is, we start with the customer and what they want and let that define and guide our efforts.

When deciding on a new product or system feature begin by writing a Press Release (PR). The goal of the PR is to describe - in a simple way - what the feature does and why it exists. If we were to announce to the press (and our customers) what would we say? What problem does it solve? What benefit does it provide? The process of writing the PR forces you and your team to focus on these questions. It is a mechanism that surfaces the most valuable customer requirements, ensuring that *before we build anything,* that what we are planning to release is something that really matters to our customers.

The next step in the Working Backwards process is to write a Frequently Asked Questions (FAQ) In this step, you clarify the customer experience as well as answer "*how* will this feature or service work?" The FAQ may include answers to questions:

- You might expect your customers to ask
- Internal teams (other service teams, business teams, finance, etc.) might ask
- Naturally arise when using the feature for the first time
- Are sent in response to the press release

With the press release and FAQ in place, now feature design and mock-ups can be constructed. What will the feature actually look like? How will customers interact with it? How does the feature impact the other aspects of the customer's experience (e.g., site latency, convenience, Prime membership, workflows, etc.)? What APIs are needed?

When all of the steps above are completed, development and testing can begin. The Working Backwards process works very well with Scrum and other agile methods, as the goal is the same: Deliver the most important features to the customer FIRST.

The primary benefit of the working backwards process is that once we get the deliverables crisp then we can let development teams go and build unimpeded.

When a team doesn't get the customer requirements clarified upfront they risk realizing well into the development process that they haven't defined a feature correctly (or worse, they are not building what actually matters). It is often this misstep that forces teams to go back to earlier stages and even restart– both of which are very frustrating.

Newcomers to the working backwards process often misperceive that iterating on the working backwards docs before writing code delays the project. However, experience has taught us, that we finish much more quickly and effectively when we get the working backwards docs right from the get-go.

For more detail, see

Be an owner

Company Technical Managers are given a business or problem space to own and just like a small business, they have a great deal of autonomy to make decisions and staff in any way they deem

appropriate to deliver exceptional value for their customers. Your ownership includes, but is not limited to:

- Team Mission, Tenets and Deliverables
- Product Definition/ Project Management
- System Design and Architecture
- Coding, Testing, Release and Deployment (Quality)
- Operational Excellence (Support/Maintenance)
- Availability and Latency
- Customer Experience, Business Impact, and Financial Metrics
- Infrastructure Scalability and Efficiency
- Staffing: Recruiting and Retention
- Employee Development, Performance Management, Promotions
- Budget

You own your customer experience.

Even if you have a Product Manager (PM/PMT) or Technical Program Manager (TPM), Technical Managers at Company are still expected to be a voice for their customers.

As Jeff Bezos says, "At every meeting and in the midst of every decision we make, one important person is not present: our customer. We have to vigorously advocate for him/her and earn their trust."

The best Company managers find a way to interact with their customers directly, even if they are downstream of your services. You can't delegate ownership of your customers. If you are building tools and services used internally, go talk to your customers. If you're building an external facing product, attend some of the user studies yourself. Whatever you do, don't be blind to the actual end customer. Create metrics that measure your customer's experience and then review them daily to make sure your systems (and team) are delivering an excellent experience.

You also own what you don't own.

Namely dependencies and escalations. If you depend on another team to deliver a feature, complete a project, or to keep your business running, **you own driving their delivery** and, if necessary, escalating appropriately in situations where your team is unable to get on their priority list. A service owner owns their service all the way to the customer. If the situation warrants it, you can work to remove them as a dependency. It's your call. As the owner, you should be looking around corners to foresee problems. At Company, excuses that another team is blocking you never go over well.

Giving you strong ownership is how we retain the agility of a start-up. This type of ownership might be more than you are used to if you came from a company where a product owner set your

strategy, handed you a product design, which you then worked and then moved onto the next when a feature was marked as complete.

At Company, this is your team to run. **Think long term**. Remember you are living with the design and quality decisions made by former owners so do your best to leave a great legacy for the next owner.

Focus on Operational Excellence

Prioritizing operational performance is tantamount to prioritizing the customer experience, which should always be one of our highest priorities as leaders and Companyians. Keeping a continued focus on operational excellence improves the quality of our technologies and, ultimately, reduces disruption to our customers. This also ensures our engineers are able to spend more time inventing and building new solutions which is key to developing their skills and improving their morale.

Operational Excellence is making sure that your systems are running properly, your customers are taken care of, your team measures latency as well as availability, and your team completes and resolves the root cause of any defects in a timely manner. It is a consistent focus on “are we up” and “are we running extremely fast”. It is continuously working to improve your systems for the best customer experience. It is documenting, setting up monitoring, scaling for growth, managing server efficiency, lease replacements, migrations, game day events, and more...

As a manager, you should:

- Create operational metrics that determine whether your customers are having a good experience and that show how well your systems are running.
- Set Service Level Agreements (“SLAs”) on those metrics.
- Review those measurements every week.
- Meet as a team to review operational metrics and set the expectation that meeting SLAs (which are the proxy for the customer experience) is a top priority.

Where Operational Excellence is concerned, a Technical Manager must walk this talk (which means emphasizing the importance and taking concrete steps to construct and regularly review these measurements) for your team to prioritize operational performance and their customer experience appropriately.

With or without a support team, all engineering teams at Company face the same challenge: *How to balance the needs of a growing business with the requirement to maintain existing systems.*

Focus only on support and project delivery suffers (impacting customer delight and business growth). Focus only on project delivery and system stability, quality and security are at risk - consuming more and more engineer productivity and eventually impacting customer experience.

It’s so easy to de-prioritize operational work and “get to it later”

Unfortunately the work just piles up and the severity of problems deepen—making them harder to fix. Getting to problems later can actually end up costing you more as people lose familiarity with the issue or when the engineers, that did have intimate knowledge of the issue, leave the team.

There is no one-size-fits-all solution. Each team needs to ensure they create well-architected, well-written code, their customers have an excellent experience, that latency is low and availability is high and that their SLAs measuring all of these are met. Just remember, as the manager of the team, YOU have the ability to change your team's operational support situation.

You just need to decide it matters and prioritize operational excellence activities.

Making sure your team is evaluating operational performance, defining acceptable system performance before shipping, doing more rigorous QA and testing before releasing software, trading off features for fixing chronic defects or staffing operational support, etc.

For more information see:

Invent simple solutions.

Company celebrates and rewards automation and simplification. Not only is the simple solution usually the best for our customers it is almost always easier to maintain. As such, we are always looking for opportunities to remove complexity in our systems. We admire Technical Managers who work themselves and their teams out of a job using smart automation. *(Don't worry—we have no shortage of hard problems for you to solve!)*

As a Technical Manager, you are expected to judge new ideas from customers, stakeholders, and even your team, on several axes including how complex the idea is to implement and sustain over time. When evaluating changes to existing systems, be on guard against the "kitchen sink" approach of adding all features regardless of the impact to the system. Beware of bolt-ons and work-arounds as you seek to apply innovations to existing systems and services. Challenge your teams to "keep it simple" in all they do.

Be wary of big-bang projects with long release times. Often the landscape changes before the solution can be launched causing rework and lost effort. Better to parse a big system and build/replace manageable pieces than to flip one giant switch.

Iterate, release in smaller chunks, release often.

At Company, engineering teams are staffed on long-term strategic bets. Rather than focusing on a project, delivering it, and moving on to the next one. Instead, teams iterates – continually delivering value for their customers and to the business in an ongoing manner.

For almost all engineering teams, deployment is a lightweight process. The goal is to build in smaller chunks and release well-tested versions often (ideally weekly or monthly).

Whatever delivery schedule you choose, the operational performance of your systems and the customer experience must be stellar. It is not acceptable to deploy features of inadequate quality (or that break functionality) into production in the name of "release early and often". It is also NOT okay to launch with the idea that if something is wrong your customers will let you know. Unless you are running a customer focus group, your customers are NOT your testers so plan your launches accordingly.

The benefits of releasing early and often are:

- Your customer gets incremental improvements on a regular basis and can provide feedback on what you've done so far to improve the next version.
- You build a documented track record of success for your team.

As a Technical Manager you will need to think agile and be agile. Don't get too hung up on the decision-making process. **We value calculated risk-taking**. Design, build, test, and launch. And do it often. In the technology environment that we operate in, hesitation can mean missed opportunities.

This expands on one of our leadership principles "Bias for Action", which in Technology translates to delivering distinct improvements for your business **often**. Leaders are expected to use their judgment to enable Company to operate at a fast pace in an effective manner. This doesn't mean making bad decisions, it means **making good decisions faster**.

Apply the Company flywheel process to your organization. Get some wins early. Build up momentum and you will find your first year as an Company Technical Manager a success.

Data, data, data

We are a data-driven company. Data is the backbone of every decision we make. We don't just expect all of the smart people who work here to come up with really good ideas; we expect them to **back up proposals with data.**

As a Technical Manager, you are expected to build and instrument your systems in such a way as to allow your team to monitor and measure:

- The customer experience
- Business and financial impacts
- Test coverage and success (e.g., pipeline metrics, service availability in test environments, etc.)
- Operational metrics (e.g., tickets, pages, errors, availability, latency, uptime, etc.)

Figure out the data you need to understand if your products are successful and how to make them more useful. If necessary this can mean building the system that can produce these metrics.

Data doesn't lie. Data is not personal.

The best way to get something done at Company is to drive the right metric. Real data is one of the most effective ways to get other teams and senior leaders onboard with your proposals. Incorporating metrics into narratives ensures everyone at the decision table knows what is at stake. Below are some of the ways we like to measure data:

- **We like to see baselines with relative metrics**. It is always better to show the starting, lowest or highest point when referencing a current metric. This gives context and shows if we are trending in the right direction.
- **We prefer to see percentiles** (tp50, tp90, tp99) rather than "the average" for measuring central tendencies (i.e. median). This gives us a better *general* idea of the distribution of the data.
- **We analyze metrics at four 9s (99.99%) and [1]five 9s (99.999%)** because in the tail of a metric is where some interesting problems, and potentially our greatest opportunities lie.
- In addition to summary statistics, at times **we find it useful to look at the whole distribution** to see the ways in which data might be misbehaving. Often it is this misbehavior that will teach us something important about the business. For more on this, watch the Principals of Company (POA) Talk: Monitoring Large Scale Services
- **We log everything**. Sometimes, in an effort to understand a problem, it is important to go all the way to the bottom of the source data – the logs themselves. The best technical managers at Company study the logs of their services.

Understand how we are frugal.

One of the Company core values is frugality. There are several good reasons for our frugality but the most important is to make sure that, as a company, we are effectively prioritizing our initiatives. Without a large budget, leaders at all levels are forced to choose which areas to invest in. This is one of the reasons we like real data. It provides a solid justification for our investments. Once a decision is made, managers need to be smart, creative and agile to deliver the features that add the most value with their precious resources (engineers).

This is a real challenge. You will not have enough resources. No one does.

One of our peculiar ways in Technology is that we have a tendency to lose respect for managers who say they don't have enough resources to get something done. That is not to say that there aren't times where a team is not resourced correctly. There are and if you feel strongly you are under-resourced, work with your management chain to resolve it. However be careful about using this as an excuse because it shows that you are missing the point.

We want to run lean.

Running lean allows us to work on the big ideas long-term without building too large a fixed cost structure. As a Technical Manager, you are expected to expertly prioritize so that the projects your engineers are working on add the most value to the business. So if you find yourself in a situation where someone is asking for something that is not on your roadmap, it is better to be

[1] *Not all systems need to operate at five 9s. You need to know if yours does.*

open about how you are prioritizing. This allows your stakeholders to assess whether their request should be higher or lower than what is currently on your list and together you can make the best decision.

(Note: If you really want more headcount, then deliver exceptional solutions with what you have. Delivering great results not only instills confidence in your leadership but it provokes the question "If this person can do this with this many people, what can they with more?")

Some other ways that our frugality impacts you as a Technical Manager:

Project Delivery: Carefully vet all projects to make sure you're **always** working on the highest priority items, be they strategic or tactical. Make sure your team is not over-building. Don't build the gold-plated version of a product that doesn't need gold plating. Setup mechanisms to ensure you are delivering the best customer experience.

For more on this watch this video of Jeff Bezos discussing "Why Good Intentions Don't Work"

TCO/TPO: TCO (total cost of ownership) and TPO (total price of ownership) refer to hardware costs. Be conscious of costs when designing new software projects. Be wary of costlier items (databases) and look to use more cost-effective solutions (e.g., AWS products like EC2 or s3). Be prepared to defend your team's hardware decisions, especially if the alternative solution is man-months of developer work or a less reliable or maintainable product.

Build vs. Buy: At Company we tend to build. Most off-the-shelf products provide 80% of what we need, however it's our experience that the other 20% are required features unique to Company. Not having those features have the potential to severely hinder employee productivity or worse bottleneck the business. So while you should always make sure you're not having your team build something that you could acquire (typically via Open Source or another Company team), be wary of buy situations. Many seasoned, senior IT leaders hired from the outside have challenged this, but most have found that they don't scale for what we need. Very few packaged software installations survive here.

Travel: Most teams have a limited travel budget. If you have engineers on other continents, you will have to be strategic about how often to meet. Engineers in remote sites benefit greatly from trips to Seattle. Not only do they get to meet folks here, but they also get a tangible sense of our culture which they in turn bring back with them. It is also important for you as a Manager to go to them and if they have team-mates in other regions to support face-to-face meetings whenever possible. Ultimately it is your job as a manager to ensure teams located in remote sites behave in an Companyian way and feel connected. A key goal of trips are to assess the culture at their site and through your presence, let your team know that the company cares about them. Whenever possible use video conferencing or Chime to get face to face time.

People: By far, your biggest asset as a manager is your team**.** The best way to be frugal is to make sure your engineers are as efficient as possible. Use development/project management processes that limit/reduce interruptions and task context switching. If an engineer needs a $150 piece of

software to optimize their time or another gigabyte of RAM, buy it – the time savings (and the happier developer) will be well worth it. Lastly, while it seems like an easy place to cut back, don't forget to spend a bit on fun activities. They build team camaraderie and morale. Have a discussion with your manager on how much you can spend on team events, etc. Take your team out to lunch once in a while and arrange activities that are inclusive and fun for all.

Communicating the Company Way

This section lists out effective means of communication at Company, whether your audience is the senior management team, your peers, or directs.

Written Communication

We communicate a lot in writing at Company, and we're not just talking email. We have several peculiar ways with regards to communication. The first of which is **we use narratives (6 page documents) rather than PowerPoint presentations** for several reasons:

- Written documentation forces you to organize your thoughts and this helps to avoid misinterpretation.
- We prefer the emphasis of *substance* over *style*.
- Slide-based presentations are linear and flow at the presenter's velocity; written documents allow readers to read at their pace and in any order.
- Questions that come after reading an entire written document are better informed than those that arise part of the way through a slide-based presentation.[2]
- Written documents last. They outlive their author and the moment in time they reference.

Important tips for writing papers at Company:

- Be sure to reference who wrote the document, the date it was written.
- Add "Company Confidential" and page numbers to the footer.
- Spelling and grammar count. Use spell check. Ask a peer to be your "editor". Generally take the time to make sure that the document has no errors. The last thing you want is for the message to be obscured by typos.
- Take the time to format the document properly. Look to see if the fonts, spacing and paragraphs are consistent. This helps the reader focus on your content.

[2] *For one perspective on this, read Edward Tufte's article in Wired on why PowerPoint Is Evil*

- Reduce wordiness. The best content is crisp, clear, and concise.

One-pager

One-pagers are effective in communicating the high-level goals, tenets and design of a project. Writing a one-pager is a great exercise as it forces the author to be very crisp about the value a project will add. A well-written one-pager will allow your audience to quickly understand the project, evaluate its benefits and risks, and make high-level decisions regarding it.

Also see:

Narrative

The narrative is a written document specific to Company. A typical narrative consists of three pieces:

- 6-page narrative with tenets
- 1-2 page FAQ (accompanying the narrative)
- A set of appendices supporting the narrative is optional

Every narrative must:

Be no more than 6 pages long[3]. The author can provide as much supporting data as needed in an appendix, but there is no guarantee that anyone will read it. It is not recommended to move text into the appendix that must be read for the narrative or tenets to make sense. The appendices are there for reference only.

Be formatted and readable. There is (currently) no strict rule on font size and borders but the author must ensure that this freedom is not abused. There are no extra points for fitting more words in 6 pages.

- **(Ideally) Start with tenets.**
- **Clearly state the objective in the first paragraph.**
- **Make a recommendation or call out next steps at the end.**

When followed, the above requirements are a highly effective mechanism that drives upfront thinking, clarity and precision on outcomes, and alignment between the problem and the solution. In regular reviews of an area, it is advised that the author maintain the same high level format for the narrative. The predictable and consistent format makes it easier to read the narrative over time.

When writing a narrative, keep in mind that your audience may not have much prior context so it's important to edit these into a clear, crisp set of documents. Generally the first part of the meeting will be spent reading the document, and then discussion and Q&A will follow.

[3] *Longer isn't necessarily better. If you can get your point across in 3 pages, do it.*

Tenets

Tenets are a few, carefully articulated guiding principles for a program or business area. They simplify decision making and act as an effective guide for the team and senior leaders to align on a vision. Narratives often refer back to tenets and use them as tie breakers when making tough judgment calls. Teams are encouraged to improve their tenets, perfecting them over time, if a narrative or learning from past decisions exposes an opportunity for improvement.

Lastly, in documents, tenets usually have the moniker of "unless you know better ones". This is because they are always evolving. Tenets at their best are living, breathing principles. Leaders are encouraged to welcome input from others to help refine and improve them.

COE (Correction of Errors)

The intent of the Correction of Errors (COE) process is to improve the overall customer experience and operational performance of our services. COEs drive accountability and enable us all to learn from problems experienced by others. As part of the resolution of most Severity One tickets (and some Severity Twos), the owner is expected to fill out a COE document (ideally within 48 hours of the incident). Owners may also be called upon to present their COE at operations reviews and/or post-mortems.

We expect managers to answer the "5 Whys" in their COE. The "5 Whys" is simply, ask "why did this happen" 5 times (or more…) until you get to the root cause. (For more on the 5 Whys watch this short video:

It is not enough to say that a problem is due to a "known issue". As the owner, you should know why it happened, what it would take to resolve the root cause, and, if you are not prioritizing that resolution, why and what is taking precedence. We expect leaders to look inward and be transparent about the issues that led to significant incidents. At their best, COEs educate us all about areas for improvement. As a company, we should always be thinking about how we can build our products and systems to have the best customer experience and be resilient to components and dependencies failing.

Technical Promotion Documents

The objective is to present a balanced and factual account of the promotion candidate's accomplishments, capabilities, and leadership behaviors. Despite the overall goal to obtain the promotion, you are also expected to articulate in writing why the candidate may not be ready.

360 degree feedback is a critical component of a promotion. Managers should not edit or cherry-pick actual feedback. Solicit and include development areas for that employee, which includes super powers and development areas.

A promotion justification that is too long (or too short) can be ineffective. Documents that are too long can obscure a candidate's accomplishments. Too short and there isn't enough information for reviewers to properly assess.

Be honest, do not embellish, do not oversell. Let your employee's accomplishments speak for them.

Provide Details. The majority of questions you are asked during a promotion review are either based on our Leadership Principles, how expectations match the next level, and for more senior roles, Principal Tenets Your goal is to answer those questions *in the promotion justification* so that reviewers don't need to ask them.

Provide context to help reviewers understand the employee's work. Always focus on the employee's contributions first, and the background of the project second. A sentence or two on each product, system or project describing what it is at a high level and why it is important is generally sufficient.

Answer hard questions up front. For example, if the candidate has been in the current level for a short or long period of time, answer the question 'Why now? Why not wait until the next promotion cycle?' or 'Why has it taken so long for this candidate to get to the next level?' Be prepared to answer these questions.

Identify development areas. Everyone has areas they need to work on. As a promotion is a 360 review of the employee, the document should be clear what those areas are and how they are working to address them.

Verbal Communication

Engineering teams at Company have an open communication style. We prefer communication that is honest, direct, and crisp. We also like the effectiveness of hashing out decisions in-person. It is not uncommon for managers and engineers to openly share their ideas in a meeting, even if those ideas are contrary. We find it efficient to simply and quickly get to the point. For those coming to Company from other company cultures, this can seem a bit intense. It's important to know that all involved are passionate and have the best interests in mind for our customers, business and company. Your goal as a manager is to make sure discussions are inclusive, so all voices and perspectives are heard.

Tips for Effective Meetings

Be prepared. The number one cause of a disastrous meeting is a lack of preparation. It is counterproductive to "wing it". If you are presenting, make sure the document is clear and to the point. Take steps to make sure it has had a wide review from peers and select individuals in your management chain prior to the meeting. Be prepared to answer deep questions in detail. Don't focus on trying to guess what you'll be asked about; focus on knowing the facts and data in detail. If you don't know the details, ensure that the right people are invited to the meeting. Be prepared, but don't spend too much time practicing. Spend your time getting the right data and answers.

In the interests of keeping high-level meetings on track, it is often best to speak only about your areas of responsibility (OR when asked to speak). If you chime in on other topics, do your best not to derail the meeting. In meetings we usually cover a lot of ground and so it is important to

keep things moving. Some folks take this too far… speaking only when "directly" spoken to and the person has made eye contact with them and specifically asked them "what do you think?" (We don't recommend this type of communication *avoidance*.)

When you do speak, be brief and concise. Give a crisp answer then stop. Do not go into excessive detail.

Listen and answer the question. If you are asked a question – answer the question asked. If you don't understand the question, ask for clarification. Then extrapolate if needed. If you don't know the answer, say "I don't know." *Don't guess.* Answer "who" questions with a name. Answer "when" questions with a date.

Admit mistakes. One of the worst things you can do is try to minimize or hide a problem. It will negatively impact the decisions we make on behalf of our customers and it will hurt the credibility of your team. Just as we would expect you to own your successes, you need to own your problems. **Don't make excuses and don't guess.** In situations where you are asked something and do not know the answer, simply say "I will find out and get back to you."

When you are in a hole, stop digging. In situations where someone expresses public frustration with you, a natural reaction is to want to defend yourself. Before reacting, consider carefully *what* they are saying. Are they right? If so, take ownership of the problem and any resolution. If however you do not agree or you believe your team's work is being unfairly characterized, then of course speak up. In either situation, consider if the meeting itself is the best time to speak up or if it would be more effective to discuss offline.

(Note for those new to management: If you are feeling defensive then the best advice is to avoid raising the issue again during the meeting – odds are you will say something you will regret.)

We expect resolution or a plan for resolution by the *end of the day*. In some meetings, issues arise that are considered urgent and leaders in those cases will expect immediate action from the owners of the problem. If you are the owner and the problem cannot be solved that day, commit to submitting a resolution plan by the **end of the day**. If the end of day comes and you don't have a plan ready, send an update to the meeting attendees with your timeline for getting the plan.

Take criticism as a learning opportunity. If you are criticized in a meeting, don't take it personally. Their passion will sometimes make it feel personal. Generally, there will be good lessons in the feedback provided. Listen to them. Take notes, and read them the next day, when you can distance yourself from the meeting itself.

Stick to the facts and the data. The best arguments are data driven.

It's never "somebody else's problem". Being blocked by another team is never a good excuse. Saying "it's not in my charter" isn't either. Everything in your way is your problem. This is not a license to be a maverick and storm a solution without communicating to the teams around you. It's always best to find simple solutions that benefit everyone but when you're truly blocked, it is

ultimately your responsibility to figure out a way to get around the problem and/or escalate -- firmly.

Tips for Effective Emails

Email is how Company communicates. Whether via computer or smart phone, everyone reads and responds to emails. There are thousands of mailing lists; and both your email and team's email will be subscribed to many of them. The amount of email an engineer gets can be overwhelming. It's a personal choice whether to filter to a folder or manage them all in your inbox. Whichever you choose, just stay on top of it all.

Always consider your audience. Our open communication style applies to email…up to a point. How you write an emails can make the difference between someone reading them or not. It can also impact their perception of you.

Answer in a timely manner. Almost everyone at Company expects a quick response to their email. This can be challenging if the subject is complicated or requires troubleshooting. If you receive an email directly sent to you or your team and you cannot respond within a day, do your best to reply to the sender with some idea of when you will be able to respond. This simple act sends a message that you are on top of things.

Be concise and to the point. Clear communication helps everybody. *With email, the most effective style is simple, direct, and crisp*. Do not make an email longer than it needs to be. Assume folks will not scroll. **A great best practice is to get the *most important* part of your message across in the first paragraph**. Short paragraphs with blank lines between them are easier to digest. If someone receives an email that looks like a dissertation, chances are they will not even attempt to read it!

Avoid the personal and emotional. A bad mood often shows in writing. If you are frustrated about something, write the email and then **hold off on sending it for a while**. Go to lunch; sleep on it. Wait until the emotion has passed, then re-read it. Often you will see where your message was overly negative and could be worded better. In some cases you may realize that you don't want to send it at all.

Use proper spelling, grammar & punctuation. Improper spelling, grammar, or punctuation can have an adverse effect of causing readers to downgrade or outright dismiss the content of the email. Spell check is your friend, use it.

Read the email before you send it. It is a good idea to always read your email before hitting send. Try to read it through the eyes of the recipient. This will help you send a more effective message and avoid misunderstandings.

Do not overuse "Reply to All." Only use Reply to All if you really need your message to be seen by each person who received the original message. This is especially important when replying to messages sent to majordomo lists. In those cases it's important to keep in mind that if you reply all you might be spamming hundreds of people.

Don’t overuse the High Priority flag. Only flag an email as High Priority if it needs immediate attention or is otherwise so important enough that it really needs to stand out in the recipient’s inbox. Many people will set special rules for High Priority emails, including alerts on their mobile devices.

Don't encourage/allow discussion on important announcement lists. Announcement lists carry information that is critical for subscribers to do their jobs. Frequent, lengthy conversations on these lists are bad, because people can't just choose unsubscribe if they aren't interested.

What it means to “Disagree and Commit”

At some point you will be placed in a situation where you don’t fully agree with the proposed path. Think like an owner. The first step is to voice your concerns. If you are unable to affect the decision, the next step is to meet with other stakeholders.

Ask questions. Be sure you understand the problem. Be open to the possibility that you may not have all of the information that went into the decision (some data and strategic initiatives are highly confidential). Listen to their point of view and share yours openly and *respectfully*.

After considering all of the inputs, if you still have concerns, the last step is escalation. Only escalate if you truly feel the decision is not in the best interests of the company. Company is a data-driven company so get the data and the facts you need to support your opinion before escalating and have a proposal for an alternate plan of action.

Once a decision is made by the group, you are expected to support it wholeheartedly and commit 100%.

Present the decision as a *group decision* to your team and stakeholders and then align your resources with the decision. Facilitate an open discussion about other alternatives that were considered and the reasoning that led to the final decision. Avoid statements like “I did not agree with the decision and tried my best to convince folks otherwise” or “I was overruled and now we unfortunately need to do this”. You want your team aligned with the decision.

Encourage your team to speak up, even when they have contrarian opinions. Foster an environment of open communication where team members can voice their opinions or concerns freely. At the same time, reinforce the idea that the team can collectively evaluate the different options and reach a collective decision. Once the decision has been made, the team needs to rally behind the decision as a cohesive unit. Don’t allow disagreements to linger and slow the group down.

Disagree and Commit is strongly encouraged in all job families and at all levels. However this becomes particularly important if you have just transitioned from an individual contributor to a role as a Technical Manager. You are now responsible for effectively communicating and streamlining information from your team to upper management and vice versa. As a result, you need to be cognizant of how you communicate these decisions to your team; this includes

communications that directly impact your people such as performance ratings or compensation decisions. If you do not exhibit conviction in your communication, what you say will be interpreted and/or executed poorly by your team and your credibility as a manager could erode.

Hiring the Company Way

Let's face it: if you don't have good people it doesn't matter how good of a manager you are, your team can't deliver. Hiring is one of the most important things we do at Company and (like everything else) we have our own peculiar ways of going about it. This section summarizes our hiring process and provides some additional insight into areas that are particularly Companyian.

It's all about making the right hire!

The Company hiring process is designed to ensure that we hire *only* the very best people. In order to maintain our high standards, we tolerate a high rate of false negatives. This means that it is expected that we will pass on marginal candidates and sometimes on candidates who could do a great job but did not successfully demonstrate it in the interview process.

You and other interviewers must assess whether a candidate is someone you *want* to work with, not someone you would merely tolerate. It's not ok to hire *just ok* people. Be careful in times of extreme need—when your team is very understaffed, it's tempting to let through a candidate whom you otherwise wouldn't. Don't do it. Don't compromise!

An engineer that performs below our bar has a higher potential of producing mediocre work. The result of which may include a poor customer experience, slower development cycles, increased operational issues due to poor code quality, time spent managing and dealing with a poor performer and eventually time spent rehiring for the role. This is the fallout from a bad hiring decision.

Your hiring decisions are your legacy.

At Company, when we talk about the "Bar" we are referring to the expected level of competency for our employees. Company has a very high hiring bar and our goal as a company is for the bar to increase continuously over time.

You are responsible for ensuring that every new employee is better than at least 50% of employees currently in that role.

To help us in this effort, Company nominates and trains "Interviewer for the company s". Interviewer for the company s facilitate the process of hiring and keep us focused on making the right hire, every single time. For more information, see:

The Process

The process for hiring an external employee is not rocket science. You open a position, find candidates, interview them, and make an offer. But we like to put our own little twist on it. The

time and success rates vary somewhat. It is not uncommon to do 30 phone screens to get one candidate. As the hiring manager, you own the process and you need to be good at it. Even if you don't have any open positions, we recommend that all new technical managers seek out and "shadow" both hiring manager and technical interviews so they are prepared.

The Work Available Must Match the Candidate's Level

Hiring candidates at the right level to match the work that is available is critical for a Manager. *Under* or *over* leveling the work you have can be an issue to attracting or retaining the right kind of talent. This occurs more often in desiring a more senior contributor than the work you have to give them. An example is requesting a SDE III when you really have SDE II work. Not only will this be problematic in retaining the SDE III who is used to doing more senior work, the employee will be judged in their contribution against other SDE III's who are working at that level.

A best practice is to ask yourself what you really need to accomplish and what level employee do you need. If you need an SDE III but would take an SDE II, consider the possibility that you just need the lower level (or have under leveled the work).

There are no perfect answers to these questions nor is there a checklist to make a decision. When in doubt, work with your HR Business Partner, Interviewer for the company , and your recruiter to make the right decision.

Opening a Position

To open a position, you will need to be ready to submit a job description that will be posted both on our internal and external sites. To attract candidates, make the descriptions as exciting and specific as possible (without disclosing anything confidential like future products or services), but do make sure that it remains realistic. Otherwise, you will be interviewing candidates who may not want to accept the position once they've gone through the process. Remember to use inclusive language. Some words and statements can turn off certain types of people.

In some organizations, there is an additional requirement that the job descriptions be approved by Public Relations (we don't want to inadvertently use job postings as a press release) - check with your manager.

Identifying Candidates

Employee Referrals and **Internal Transfers** have proven to be the most effective way to hire the best Companyians. Continuously encourage your team to talk to friends and previous colleagues about opportunities. Proceed carefully when you are looking to attract internal candidates. If you believe that you have a position that is a good fit for an engineer from another team, talk to their manager before talking to the candidate. For all the rules of engagement around these processes, see the following pages:

Some candidates may come in through candidate pipelines in our recruiting system. In this case, it is up to the pipeline manager to determine how they are assigned (tagged) to positions. When

it comes to tagging, you and the recruiter will tag candidates for your position. If a resume is tagged, it must be dispositioned[4].

Interacting with the Candidate

You are directly representing Company in all your conversations with candidates. Whether or not you want to hire them, conduct yourself professionally. Treat them well and let them ask questions. **Every candidate is a potential Company customer and ambassador**. They should have a positive experience with the company irrespective of our hiring decision. We want them to keep shopping at Company and keep using Company services. We also might want to hire one of their friends.

Before conducting any interviews, make sure you:

- Read and understand the Company PR guidelines
- Take Making Great Hiring Decisions (training)
- Talk to your manager about what you can and cannot speak about with potential candidates (i.e. technology/product information that could compromise Company's strategic advantage)

With all interviews, you should review the candidate's resume and prepare the questions you will ask. It reflects poorly upon you and Company when you are unprepared for the interview.

Be honest with a candidate about what will be expected of them, and don't make promises you can't keep. In Tech, it is a good idea to inform engineer candidates about:

- Company Open Source Policy
  ""/""/Open_Source
- Company External Communications Policy
  ""/""/en/About/Companypr/Pages/commprotocol.aspx
- Any operations or oncall expectations that require working outside of business hours

Additionally, if a candidate is from out of town, they may be on their own in an unknown city. Try to suggest places for them to visit or things to see, or better yet, consider taking them out to dinner. Ensure their experience with Company is positive.

Phone Screens

When you request a phone screen, you are committing the team/company resources to interviewing the candidate and ensuring they have a good experience. Use these resources wisely.

[4] *See Company OFCCP legal requirements.*

In Tech, a candidate will go through two technical phone screens to help us determine their suitability for Company. As a manager, you may have to perform some of these screens, and there is plenty of material about conducting and writing feedback about phone screens.

Pay attention to the interviewing effectiveness of your team members. Discuss this with recruiters and Interviewer for the company s, and use their feedback to improve their effectiveness. For more tips see:

Constructing an Interview Schedule (Loop)

As the Hiring Manager, you are responsible for selecting interviewers for a candidate's interview loop and assigning areas (competencies) for each interviewer to focus on. Consider the feedback from the phone screens and the expected level of the candidate when constructing the loop.

The goal of the competency assignment is to make sure that the interview team collectively has enough data to make a hiring decision.

In Tech, it is strongly encouraged that interviewers focus on core technical strengths as well as Leadership Principles during in-house interviews. For an SDE role, for example, you may have some interviewers focus on core technical skills (e.g., coding, design, algorithms, problem solving), while also focusing on "Ownership", "Bias for action", or "Customer focus". These in particular have proven to be particularly useful in identifying successful candidates for Company. Pick the leadership principles that matter most for the role and assign competencies to the interviewers in that fashion.

In-House Interview

As a manager, you will often be the "Hiring Manager" on the interview schedule, meaning that your interview should assess whether or not you want to hire a candidate for your team *and* for Company. Do not dismiss your Hiring Manager interview by giving the candidate a "free pass." Use the time to determine whether they:

- Understand the position and its expectations
- Will raise the bar at Company
- Are a good fit for your team.
- Would be a good fit for Company (they won't be on your team forever)

Feedback

When you have to make a decision about a candidate, make sure you are able to make an informed one. In other words, ask questions of the candidate that allow you to form an opinion either positively or negatively rather than indifferently. It is your responsibility to enter feedback into the recruiting system *before* the hiring debrief so that everyone can review it easily.

Your feedback should include your position on the candidate and supporting evidence. Write enough so that any person reading it can understand why you came to the decision you did. Be

honest and be clear when you editorialize. For example: “I asked the candidate how to tie a shoelace. I expected them to show me one way, but they showed me three ways – one of which is completely brand new. This was an excellent answer and shows they know how to innovate.”

In technical interviews, include code samples from the candidate to demonstrate their coding/thinking style and skills. This is valuable, though it often requires comments from the interviewer to identify mistakes and problems.

The Debrief

After the interviews, all the interviewers convene to discuss the results of the interview – this is called the “debrief”. The **Interviewer for the company** conducts this meeting and holds ultimate veto power, and the Hiring Manager decides whether to extend an offer to the candidate.

The **Interviewer for the company** is a person who participates on an interview loop as an objective third party. Interviewer for the company s assure the best long term hires are made for the company. In order for an offer to be extended, a Interviewer for the company and Hiring Manager need to agree to hire the candidate as well as on the job level to extend. This provides a balance of power in hiring decisions and a higher level of accountability to the High Hiring Bar.

You own closing your candidate. Do what it takes.

Some candidates respond well to “book bombs” (if your org uses them). A book bomb is where the manager selects a few books for the candidate and sends them as a gift prior to their arrival. Usually spending about $100, they send 1 or 2 technical and if they are relocating a fun local guide like “The Pacific Northwest Trail Guide” This is a great way to welcome someone to Company.

Many candidates want additional conversations about the role and the team before accepting their offer. Reach out to them personally to help them make a decision. Call them and tell them how they could be successful at Company, what the long-term opportunities might be, and why they would be great on your team.

Working with Recruiters

It is critical to establish a good working relationship with your recruiters. Don’t make their jobs more difficult than they already are. If you commit to an interview, *do it* or take it upon yourself to find a replacement. Don’t cancel the day of the interview and leave the candidate and recruiter hanging. Remember, hiring is one your top priorities.

Keep your engineers happy… and around!

As a manager you have a huge impact on whether your engineers choose to stay with your team. We do hire the best and the best want to be challenged, to work on innovative solutions, to be productive and ultimately to be recognized for good work.

It may sound cliché but if you don’t get the basic needs covered, you will not have a productive team. Two of the most common reasons employees leave their jobs are due to poor managers

and a lack of career development. Take care of your people and they will reward you with good work. It *is* as simple as that!

Below are some guidelines on what you can do to foster a great team environment.

Ask and Listen

One of the most important things you can do as a manager is set up regular 1:1s with each of your direct reports. While it is tempting to just focus on tactical projects, make sure to set aside time to discuss what motivates them, what their career aspirations are and to have candid discussions about their performance. Be prepared to listen to their feedback, whether it's about your management style or the work they are doing.

Regular 1:1s can provide you with tremendous insight into the pulse of your organization and clue you in to your employees' intentions. Waiting to talk about things that upset or de-motivate your employees once they've made the decision to leave doesn't help anybody. Stay connected.

Keep them motivated

There are many ways to motivate engineers. These are some of our best practices:

Give them the opportunity to innovate. We hire the best and the best want to invent. Encourage new ways of looking at old problems.

Make sure they understand the benefit their work provides to customers and the business. If a project moves the needle on key metrics, invite them to the leadership meeting (if appropriate) where those metrics are discussed. Involve them in the business so they are bought into the projects they are working on. Make sure they get recognition when their work moves a metric.

Take the time to understand what challenges your team. Engineers like to work on the things that push their brains outside their comfort zone. Learn where each engineer needs to grow, as well as what technologies and innovations in the industry interest them. Then make sure that their work is enabling their growth.

Offer talented engineers the opportunity to lead or even *own* a project or part of a system. Help them get visibility and support them as the owner by putting them in front of their customers and stakeholders.

Remove barriers. As a manager, your primary role is to support your team by removing any obstacles that impact their productivity. Work to actively minimize the team's operational load, provide your oncall rotation with a wireless access card, and try to give the team quiet time to code without interruption.

Have fun! Take the time to celebrate successes, recognize your employees for a job well done. Arrange a team outing every now and then. This doesn't have to cost anything. Be creative. Sometimes just having lunch together is enough!

Developing the best…how do you do it?

Company culture expects each of us to drive our own career growth. We figure out what we want to do, when we want to do it, and then actively pursue the opportunities that are available to us.

This however cannot happen without a little help from the manager.

As a manager, you directly influence the most effective drivers of high performance: *providing fair and accurate feedback* and *clarifying expectations*. So, in support of being that catalyst, make sure to set clear goals with your employees that align appropriately to the business objective, and provide ongoing feedback. Support employee development by encouraging them to attend talks and conferences, recommending training, and providing stretch opportunities, etc. Development doesn't necessarily mean a promotion; it can be as simple as enhancing one's current position or realigning goals to better match aspirations.

At some point you may realize that you do not have the right opportunities for someone on your team. This is often the case for Managers with engineers looking to achieve the Principal level. In these situations, you are expected to support an internal transfer to another team that will provide the engineer with the opportunities they need. While it is tempting to try to hold on to a great engineer, it is more important that Company retain them.

Your engineer will appreciate efforts to help them identify opportunities, network and connect to the right individuals. Help them early on in the process as opposed to once your employee has announced they are unhappy and/or are leaving. This in turn will allow you to proactively manage your own staff while demonstrating that you have your employees' aspirations at heart.

Manager's Guide to Company Development Tools

Over half of Company Technical Managers have developers on their teams. While managers themselves are not expected to write and release software, it is valuable for you to understand the tools they use on a regular basis. This section provides a high-level overview. The steps documented should not dictate how your team develops software.

If you want to dive in a little deeper, check out: [“““/””/EE]

Concepts

The following concepts are used throughout this document.

- **“”** – the name of our deployment system.
- **“”** – a collective name for the tools and services used to build and release software.
- **Developer Desktop** – the Linux-based computer provided for all developers to use as their primary development computer.
- **Environment** – an environment maps a collection of software (configuration, scripts, and executables) necessary for a service or website to a set of hosts.

- **Package** – the logical grouping of software intended to be built and deployed together. For example all the code that would go into a single JAR file or EXE.
- **Platforms** – in its simplest form a platform is the name given to a specific host configuration (operating system, library and compiler versions) that packages can be built on.
- **Version set** – an Company-invented term for a revision controlled dependency closure, analogous to a branch in a revision control system.
- **Version filter** – a runtime view of a version set. Used in the deployment system to ensure that you are deploying the versions of packages that were built and tested together.
- **Workspace** – a directory where a developer does their work.

Finding what code to work on

A lot of the work that your developers do involves making changes and enhancements to existing code. Several tools exist to help developers figure out where to begin. If they know what package they want to review they can use the **"" Package Directory** to locate packages by name. This is also the place you can look for packages that you own and are responsible for. The most efficient way to find code is to use the Company code search tool.

Setting up your workspace

Developers typically start their work by setting up a workspace, or sandbox, where they edit source files needing to be changed, deleted, or created. Workspaces are created and managed using the command-line interface (CLI). The purpose of the command line tool is to hide the details behind the scenes and allow developers to work with packages and workspaces directly.

Checking out code

Editing

There are several editors available and the choice of which one to use is religious for most developers. In general most developers use one of two categories of editors: command-line editors or integrated development environments (IDE).

The two most popular editors are vim and emacs. If your team is writing Java code, or wants to develop on a machine other than their desktop, they should be using Eclipse since it includes powerful refactoring and productivity tools and runs on Linux, Windows, and Mac OS. However, any editor can be used to modify and write code.

Compiling code

During the development process engineers will compile code on their desktop using the **""-build** command This command automatically executes the build tool that is appropriate for the language being compiled (e.g., make for C or C++ code and Ant for Java code). In addition, this

command will run any unit tests that have been included in the software package and report the results.

Code reviews

All teams inside Company must require software to go through a peer review prior to checking in. CRUX (“““/””/BuilderTools/Product/CodeBrowser/CRUX) is the tool provided by Company to perform code reviews. Developers push their changes and then request a review from a peer on their team. Comments are tracked along with any subsequent changes made to the code. Once the team approves the code, the developer uses the repository-specific commands to check-in the code.

Building code

Once code has been checked in a developer is able to build the official version of the code. This is primarily done via ““code.””. Developers navigate a wizard-like interface to define what packages to build and the platforms for which they want the software built. They also choose whether or not the built software should be imported into Company’s deployment system.

Deploying the software

Almost all software is deployed using ““ (“““/””/””). This system allows any developer to deploy (install, configure, startup, shutdown, etc.) software they own to any server located on the Company network around the world. It also maintains a history of any changes made to the software and provides a means to quickly rollback a change in the event there is a problem with the software.

Testing software

Ensuring that your developers are writing high quality code is important as a development manager. How you go about doing that is up to you. Some teams follow a test-driven-development process – also called TDD. Other teams have a dedicated quality assurance team. There are numerous testing tools available for use inside Company ranging from Junit or TestNG for unit testing to Selenium for functional testing.