Meta E4 -> E5 Growth Plan

Expectations Breakdown

Keep in mind that **this is not a checklist**. The goal of this document is to provide levers for you to pull; every engineer has different strengths so they will get promoted in different ways. However, you will probably have to exhibit a good amount (50%+) of the behaviors outlined in this document in order to progress to the next level.

At high-level, going from E4 to E5 means transitioning from being a very competent IC leading small to medium sized tasks to becoming a strong leader, capable of pulling teams together and delivering end-to-end on entire projects, usually involving at least 7-10 people (XFN included). This is what it means to grow from a mid-level IC to a senior-level IC at a high-performance company like Meta.

Another attribute of E5 engineers is that they will start building an identity within their teams, teams being a mix of your product team and stack-specific team. E5 engineers will almost always have at least 1 thing (often 2-3) that they're uniquely good at, something that legitimately very, very few other engineers within their space can do. Part of your growth to E5 and beyond is figuring out what these special things are. What do you want to be known and respected for?

I want to call out that once you get to E5+, the picture gets more and more blurry as there's more paths to get to these levels and it takes much more to perform at this stage in your career. More specifically, the axes will start overlapping: For example, let's say you come up with your own project, get buy-in for it after a long slog, and then execute on it as a TL to great success.

1. The first part is Direction. However, you won't get points for Direction solely coming up with the idea (e.g. you'll see a lot of E3/E4s come up with great ideas in brainstorms, but they completely miss on the 2nd and 3rd part).
2. The second part is People.
3. The last part is Impact. You also got more impact as you exerted a lot of direction on the workstream.

Lastly, take any numbers mentioned here with a **huge** grain of salt. These ranges are very rough and will vary from org to org, manager to manager. The numbers are just there to make things clearer, especially as most E3/E4 engineers will need things to be a lot more concrete to really understand things from my experience.

Engineering Excellence

As an E4, you demonstrate very strong technical proficiency within your space (i.e. your stack and team). As an E5, you need to build a technical presence that's both incredibly deep and extends outside of your immediate scope.

E4

1. You build a reputation as someone who delivers a substantial amount of very high quality code. It is very difficult for folks to find problems in your implementation after it's launched.
2. You hold the quality bar for systems you have built. You make sure that the code you've written is always being extended in a responsible, scalable way instead of just "dropping and shipping". This largely manifests through going through relevant diff reviews touching your codebase like a hawk.
3. You continue delivering code in a timely, high-quality manner. For lower complexity projects, you build a reputation of delivering well ahead of time (25 - 50% faster than expected).
4. You care about test coverage. You may even lead small to medium-sized efforts (2 - 5 engineers) to add more test coverage to other parts of the codebase.
5. You have a strong presence in code review. You are extremely thorough catching issues within your own domain, finding architectural issues (sometimes challenging the entire diff/diff stack), performance problems, and several edge case misses. You are able to provide decent code review outside of your domain as well.
6. You have strong mastery over small (<1 month) and medium-sized (1-3 months) technical problems. You can deliver on larger technical efforts (3+ month time horizon), but you may need technical assistance scoping out these very large projects (3-6 month length or problems deep within the infra layer).
7. You have a good presence in SEVs that pertain directly to you (i.e. it's your written/owned code that's broken). You write the fix and participate in the post-mortem, pushing to understand the impact and helping define follow-up items.
8. You are working within low to medium technical complexity. Lower technical complexity is often defined by these factors:
    a. Distance from infra layer and its deeper issues like performance and reliability
    b. Proximity to the product layer. Putting pixels on screen is generally easier as teams generally use the "dumb client, smart server" model.
    c. Younger and smaller codebase - When you're working with hulking, legacy codebases powering code products, the code is often messy and hard to understand. This puts more burden on the engineer's learning fundamentals and technical depth as they try to extend the codebase without breaking it. Counterintuitively, adding a pretty separate new screen is one of the easier things an engineer can do (and E4s will often time be put on this), especially when compared to a seemingly smaller but fundamental change like making image-based Instagram stories take 3 seconds instead of 5.

E5

1. You deliver code with extremely high-quality.

    a. You are often accounting for complex long-term issues that won't be a risk until 6-12 months out in your code. This shows from your reputation as someone whose code has very, very few SEVs.
    b. Your diffs are regularly referenced as a golden standard for what a great diff looks like, setting the diff culture for the team as a whole.
2. You are able to deliver code with extremely high velocity, even with medium to high complexity projects. You are often put on urgent, mission-critical projects, those with deadlines that are far more aggressive (30%+ less time) than the average project in your org.
3. You regularly solve large-sized technical problems (3 - 9 month projects). You almost never need hands-on technical assistance scoping out your work as you can thoroughly break it down and scope it out yourself.
4. You are working within medium to high technical complexity. This can mean:
    a. Working closer to the infra layer, solving problems like cold start and frame drops if you work on the mobile side for example
    b. Being further away from the product layer. When working on the product-side, you are often building common components with multiplicative impact (e.g. a button that's used everywhere) or on the hardest UI problems (e.g. manually drawing an animation frame-by-frame in Android, which is how the polling sticker in Instagram works).
    c. Working with massive, legacy codebases while still maintaining high-velocity and code quality.
5. You have a very broad and powerful presence in code review.
    a. Instead of mainly reviewing stack-specific diffs within product domains you have worked on, you review diffs across the entire stack for your org (M2+ scope). You are able to review diffs for areas that you have never worked with before, quickly absorbing the context of the diff and its surrounding ecosystem to leave genuinely insightful feedback in a relatively short period of time.
    b. You catch large structural and architectural issues in diffs. You are able to suggest entirely different approaches in diffs, often majorly changing the technical direction of the entire diff. When you see trends here, you proactively pull together efforts to have solid code quality and architecture up-front instead of catching these massive problems at diff review time.
    c. You build a reputation as a respected gatekeeper. You are able to preserve the quality and integrity of entire systems through your high-quality, high-frequency diff review.
    d. You are able to catch issues that very few engineers on your team can catch, resulting from your incredibly deep understanding of tech ecosystems. For example, as an Android engineer, this can be catching some sort of niche performance issue that only occurs on certain devices when you do multi-threading incorrectly (i.e. something that only someone with an incredibly strong understanding of Android can catch).

    e. Your diffs are regularly cited as great learning resources among the team. Engineers say in their feedback that having their diffs reviewed by you substantially helped them grow on the technical front.
6. You are able to fix issues that very few other engineers on your team can.
    a. This is quite similar to catching deep issues in diff review in that it's the "writing code" version of it.
    b. We all know those issues that take 50+ hours to solve where the root cause is buried in some bizarre collection of strange circumstances across 3 separate tech stacks and 5 different eng stacks (and they're often SEVs to boot). You are able to fix these issues and create a strategy to prevent them from happening again or at least mitigate them if they do happen again.
    c. You are able to make the entire team more fluent in fixing these bugs. This often manifests itself as creating detailed write-ups that break down your incredibly complex fixes. A lot of these writeups have higher-visibility and impact as they come from SEV review/oncall notes.

Impact

What a lot of engineers don't realize about impact is that there's 2 dimensions:

1. The team/business impact of your projects
2. The depth of your contribution

You need to take a nuanced dot product of these 2 in order to figure out how many "impact points" you get. A lot of people think that only #1 matters and your goal, even as an E3/E4 engineer, is to choose the best team/projects with the highest impact.

However, E4, similar to E3, is not really held responsible for impact for the simple reason that they generally don't have much control over the projects they're put on. Let's clarify this with an example with 2 different standard E4s:

1. The first E4 is put on a project, builds their stack's piece with high quality and on time (and not much else), and the project is a success, contributing 50% to the team's goal.
2. The second E4 is put on a project, builds their stack's piece with high quality and on time (and not much else), and the project is a failure. The experiment is metrics-negative and the entire feature is killed.

These 2 E4s will be given the same amount of credit on the impact axis.

However, this promotion is where things get interesting: As E4s grow to E5, they need a *much* stronger depth of contribution in order to get more "points" from their projects. This section will cover how that mindset and behavior shift happens from E4 to E5.

E4

1. You are able to resolve ambiguity at the project level for your stack. For example, if you're an Android engineer: Your lead/manager should be able to go to you with fairly

rough Android designs and have you deliver a quality, stable, and on-time Android implementation with good communication and planning.
2. You are able to come up with granular, high-fidelity timelines for your stack on small to mid-sized projects (1 - 3 months). For very long-term projects (3 months+), you may need assistance from more senior engineers to thoroughly scope things out.
3. You play a big role clarifying requirements and designs, working often with PMs/designers to tie up loose ends and align the team on exactly what needs to be delivered.
4. You proactively clear out gray areas instead of addressing them reactively. One core way of doing this is thoroughly spec-ing out projects before execution and front-loading the ambiguity. This thorough planning should be reflected in feedback from leads (either product leads within your team or tech pillar leads).
5. You play a larger role in status checking and project management. You make sure that the work is chugging along at a healthy rate and it's properly tracked. You look at the work tracker (tasks, Quip, etc) holistically and call out issues if things are falling behind.

E5

1. You regularly own entire projects and are held accountable for all of its execution. You have a reputation as someone who is able to deliver projects on-time with high quality and minimal burnout among the team. You are a "magical black box" with projects: You can be given any large project and just emerge with a result after 3-6 months with minimal supervision.
2. You are able to resolve ambiguities across your entire project. Not only are you still held responsible for scoping out and delivering a quality implementation in your primary stack, but you are responsible for tying up loose threads anywhere, whether it be in a different stack (back-end, iOS) or outside of engineering entirely (product requirement question, unclear design).
    a. 1 way to concretely imagine this is that an E5 needs to be able to take just a single sentence for a project (e.g. "I want to make settings server-driven") and then deliver a result after 3-6 months for it.
3. You drive milestones and planning
    a. You create granular, high-fidelity timelines for entire projects within a half scope (i.e. projects that can be completed within 6 months).
    b. You are able to maintain this timeline amidst chaotic, quickly-changing circumstances, filling in the proper gaps as they come along. This can involve bringing in additional resources, descoping the project, and escalating issues.
    c. You create clear resources centralizing information about the project so everyone can easily know what we need to accomplish next and whether we're on track.
4. You have a large role clarifying product requirements
    a. You work closely with product and design very early on to figure out what we're even trying to build on the team to begin with. This can be through helping with the initial PRD and extremely rough, initial designs. You carefully observe and learn from UXR sessions to help you craft the product direction with XFN.

    b. After the initial product requirements and designs are out, you continue working with product and design to make sure they're thorough and 100% ready to be executed by engineering.
    c. You "hold the fort" with the product and design requirements, minimizing thrash to them to keep the team on track.
5. You are a leader in getting rid of gray areas. This is covered extensively in the previous 3 sections: If something is unclear, you are always at the forefront of the effort to make it clear. This is evidenced across multitudes of docs, chat threads, and more.
6. You are an expert communicator, making sure that the team is always perfectly aligned on what's going on. You do this through:
    a. Clear, regularly maintained specs
    b. Driving efficient sync meetings (weeklies, stand-up, etc)
    c. Communicating broader status updates (e.g. a weekly Workplace post in a large working group)
    d. Always making sure that new alignments are quickly absorbed across the entire team

Direction

At a high-level, direction works as follows: As your team executes, it will hit X amount of decision points. You get more "Direction" points by being relevant for more of those decisions and having a deeper influence on the ones that you're on, especially the more complex ones.

E4s are largely relegated to decisions local to the project and stack they're working on. For example, if you're an back-end engineer building some API for a new project, you'll work with your other back-end engineers to figure out the overall strategy/architecture and then with front-end engineers to clarify the protocol. The path to E5 requires being a wise guiding force on decisions relevant to entire projects and zooming out, on decisions dictating what projects the team even works on to begin with.

E4

1. You continue to be held responsible for your part of the stack, and your piece starts being regarded as lower risk due to your increased quality and proactive scoping work.
2. You act as the "glue" between all the different members of any project team, having/driving the conversations to produce the necessary alignment to move the team forward. For example, as a front-end engineer, this often involves:
    a. XFN - You work with designers to finalize the designs, clarify ambiguities, make sure they are technically feasible across all of mobile at least and ideally web as well. You also work with PMs to make sure that the PRD is crystal clear. If you want to go even further, you can work with parties like legal/comms to make sure the project launch is 100% safe.
    b. Eng - You work with the other stacks to make sure that all the tech comes together. For example, if you're a web engineer, this often means aligning with

mobile engineers to make sure the experiences/overall technical approaches are consistent alongside server-side to make sure that the client-server protocol makes sense.

3. You are able to independently come up with and drive new efforts spanning across 1 - 2 months.
4. You feel a strong sense of ownership over your own code. On top of being reactive through fixing bugs and answering questions, you start drawing themes among problems/support requests and creating systematic, proactive solutions to address them (wikis, a more robust monitoring system, etc).
5. You care about the project after it's been launched with a focus on your stack. You look at metrics, monitor feedback, and fix bugs without someone needing to kick a task over to you beforehand.
6. You are a primary point of contact for the systems you have built, answering questions and defining clear collaboration models so that others can extend/plug into your system smoothly. You present and evangelize your system if applicable, establishing yourself as a public face for it.

E5

1. You build a reputation as someone who delivers incredibly high-quality systems with adept technical foresight. The projects you ship are built to last: Teams are able to easily interface with it in the future without breaking it and extending it is easy. Your systems are rarely a culprit when it comes to SEVs.
2. You often play the largest role when it comes to project management:
    a. You make sure that the work is well planned out across every stack with clear milestones and reasonable timelines.
    b. You regularly check in to make sure that the project is on track, and everyone is properly aligned on the progress and what's next.
    c. You are able to spot issues proactively and resolve them.
    d. You play a leading role when it comes to escalation, efficiently bringing all hands on deck when necessary to clear the nastiest blockers.
3. You are the "glue" for every facet of your projects, filling in the gaps no matter where they are. You work closely with other engineering teams to produce extremely optimized and future-proof technical designs. You work closely with designers and PMs to solidify requirements and scope products in a way that's harmonious across all of eng, PM, and design, guiding the team out of many misalignments along the way. For more clarity on the specifics, check out the "People" section.
4. You are able to independently come up with and drive large initiatives, creating additional scope for you and your team and adding to the roadmap.
    a. This can either be a project for your product team or a large portion of your stack-specific team (e.g. Messenger Android engineers working on group chats).
    b. You often have to get buy-in from other engineers to get these projects off the ground (otherwise, why would they work on this new thing with you?).

    c. You coordinate with managers to make sure that every person that contributes to your project is properly recognized and rewarded, further strengthening the case and profile of your new efforts.
5. You have a large influence in every stack of the project you're working on (you don't have to write code in the other stacks, but you may end up doing this) operating as a full tech lead vs. just a stack-specific lead. You have a substantial presence in tech reviews all over the stack, spec documents, and planning in general. You expand across stacks that are more local/connected to you. For example, Android engineers will generally expand their technical influence through the following:
    a. Their mobile counterpart, iOS
    b. Back-end (API and client-server protocol design)
    c. Logging (schema definition, monitoring strategy, etc)
6. You look 3 - 6 months ahead with every project you're on. This means always looking at the systems you're building, asking yourself "What happens with this in the next half?", and proactively evolving it in a way that accounts for that.
7. You have both a broad and strong presence in SEVs, guiding your team/org's culture and process around fire-fighting to a better place.
    a. You regularly play a large role in SEVs that touch your domain in any way, not just directly. For example, as an Android engineer, you can increase your ownership by having a broader sense of responsibility for all of RH Android, sitting on any SEV that touches Android.
    b. You drive resolutions, lead the post-mortem, push to understand the impact, and make sure that the follow-up items are actually delivered.
    c. You make improvements to the SEV process itself.
8. You make sure that every facet of your releases is healthy. This involves:
    a. Defining success metrics. If the logging doesn't exist to produce your success metrics, you figure out the necessary logging to do so and drive the addition of that.
    b. Performing really deep data analysis to fully understand the results of your product. This often involves really complicated SQL queries and creating detailed but easy-to-understand presentations breaking down the core takeaways from the data.
    c. Building fast, easy-to-understand, and thorough dashboards so the entire team can easily understand the overall health of their products, often needing to make custom dashboards in order to produce new insights and process new metrics.
    d. Making sure that your project's signals truly are healthy and high-quality by accounting for issues like novelty effect and exposure imbalances.
    e. Defining an oncall system for the product if it doesn't fit into an existing one. You design oncalls that are effective and fair while accounting for team health.

People

Similar to almost everything else, an E4's presence in this axis is more local and isolated. You work quite well with parties that are very directly involved in whatever you're working on (e.g. collaborating with a designer to clear up an issue with the design you're currently coding out).

Your mentorship skills, if present, are budding. Growing to E5 on this axis means that you greatly expand your network and become incredibly connected throughout Meta's rich people ecosystem. You regularly work with folks well outside your immediate domain, and you improve the very processes that govern how people collaborate at Meta.

As a side note, here's some rough interview PSC math/guidelines I got. I don't know how this applies across different levels, so I'm putting it here. Like all numbers in this document, treat this with a **huge** grain of salt:

1. It takes ~50 interviews (2/week) per half to go from one sub-band to the next:
    a. If you're a regular MA on People without interviews and you do 50 interviews, you will go up to MA+.
2. Interview volume **cannot** shift you up an entire band.
    a. If you're MA+ on People without interviews and do 250 interviews, you will stay at MA+. Your very impressive interview count doesn't take you to EE-.
    b. The idea is that simply doing a lot of interviews is fairly straightforward and mainly a time commitment: There's no deeper behavior being developed here. And when it comes to promotions, especially going past E4, it's all about behavior.

E4

1. You are able to drive small to medium-sized alignments (2 - 5 people) consistently. You have a track record of bringing core stakeholders into meetings/docs to discuss tricky issues and emerging with a decision. This should be reflected with a couple non-local eng stakeholders (engineers from other teams within your stack or engineers outside of your stack) or XFN (design, PM, CS, etc).
2. You are able to mentor/lead E3s, producing clear, positive changes in their output and behavior. This needs to be reflected in their peer feedback.
3. You take on an intern and are able to get them a return offer. Feedback is very positive around your ability to grow them at a fast pace while accounting for their well-being.
4. You have a solid impact on interviewing, either by clearing packed interviewing pipelines and simply doing a bunch of interviews or being a lead on recruiting events, like representing the Meta booth at a university career fair.
5. You act as a strong bridge between your team and others. Other teams immediately think of you as a core POC when they want to interface and collaborate with your slice of the team's codebase (e.g. Messenger group chat creation on Android).

E5

1. You effectively mentor E4 engineers, growing them to E4 EE and eventually to E5. This means that you are able to concretely improve their productivity and change their behavior; this should be reflected in the feedback. On average, E4 engineers progress to E5 in ~2 years. As an effective mentor, you must be able to substantially reduce that for many of your mentees.

    a. This will often involve “creating scope” for them, either through coming up with new projects for them or carving out places where they can show E4 EE and beyond behaviors on the projects that you’re leading. In general, you push to leave teams (not just the codebase/product) in a better place after execution is done, bringing people up as the project happens.
2. You own a clear majority, if not all, of the XFN alignment issues for your projects, evidenced by a massive presence in Workplace, Work Chat, tech spec docs, and workstream meetings.
    a. For every project you own (and you should always be owning a project as an E5 engineer), you become its face as everyone naturally goes to you to figure things out.
    b. You are able to drive decisions with XFN efficiently in a way where they feel incredibly valued by you and your team and you’re augmenting their inputs (decisions often involve taking the best part of multiple viewpoints).
    c. When you’re driving a project, it has minimal XFN thrash, leading to a predictable, minimal-stress execution experience for everyone.
3. You regularly work with other engineering teams, particularly leads, to make sure that you are building and extending systems in a way that works for everybody. This manifests through driving tech reviews, doc collaboration, and frequent communication between eng pillars in general.
4. You have a very strong presence when it comes to interviews for the company.
    a. You do a good amount of interviews (>1 per week).
    b. You make improvements to the interview process as a whole, deprecating bad questions and coming up with new ones.
    c. You lead broad recruiting events for Meta, giving huge talks or working with recruiting to expand into previously untapped hiring regions.
5. If you’re a front-end engineer, you closely and proactively work with designers to make sure their designs are engineering feasible while still meeting your org’s bar for quality and craft.
    a. You are able to build an understanding of which design components are higher priority vs. not, allowing the team to intelligently make design descoping decisions if necessary.
    b. You thoroughly call out edge cases not addressed by the design, and you make sure that designs respect the platforms they’re on. For example, if you’re an Android engineer, you should have a deep understanding of Apple’s design guidelines and leave feedback accordingly on iOS as well.
    c. You build a good understanding of your org’s design principles (these will vary a lot across Big Blue, WhatsApp, Instagram, etc), which allows you to leave insightful feedback that respects both design and engineering.
6. You closely and proactively work with product managers to make sure that the product direction of the team is clear and impactful.
    a. You lead high-yield brainstorms that produce a lot of ideas that are on-topic and make it into the final roadmap.

    b. You regularly sync with product managers to make sure that product requirements are extremely clear. The other engineers on your team always know exactly the scope of what they're building thanks to you.
    c. You work with leads in planning to figure out your team's roadmap. You are able to make strong cases for the ideas you're passionate about and get buy-in. You do data analysis to identify clear user trends and needs, better informing product prioritization decisions.
7. You are a key player in onboarding.
    a. You create evergreen, staple onboarding resources for your teams. These resources are high-quality, clear, and you're able to galvanize the team around maintaining them.
    b. You are the onboarding mentor for several, if not all, of new incoming teammates. You are able to get them to peak productivity in a very short period of time while making sure they aren't overwhelmed. This leads to you becoming a face of the team.
8. You are a key player in recruiting.
    a. You're cited as a large reason, often an inspiration, for bootcampers that choose to join your team, especially more senior ones.