Tech Lead Expectations for Engineering Projects (Gergely Orosz @Uber)

Notes from the author (Gergely Orosz):

- For context on this document, please see the post An Engineering Team where Everyone is a Leader
- This is an earlier version of project lead expectations my engineering team has been using at Uber, in Amsterdam. It is highly specific to my team, and the approach we took. Our team consists of software engineers, and we work with a product manager. We ship user-facing products.
- Internal links to Uber resources have been removed.
- Originally published on the Pragmatic Engineer blog. Want to read more content/resources like this? Sign up for the newsletter.

Team: this is a living document - please feel free to make or suggest changes to it.

As a project engineering lead, your role will be crucial for the team to succeed. Below is some guidance of what this role consists of, and what the expectations are.

Project Roles

- **Product Manager (PM)** owns product decisions ("what") & external stakeholder management
- **Engineering Manager (EM)** owns staffing decisions and is accountable* for execution ("who")
- **Project lead** is and engineer, who is responsible (but not accountable*) for execution ("how")
    - Setup a framework for collaboration
    - Break down project into milestones & provide estimates on these
    - Communicate project status to stakeholders
    - Manage and call out risks
    - Help the team ship and delegate (both to the team and upwards)
    - Motivate the team

* Responsible & accountable: if something goes really wrong, they get in trouble
Responsible, but not accountable: if something goes really wrong, they don't get in trouble

1. Setup a framework for collaboration

1.1. Initial setup

- **Create a list of project stakeholders** involved in this project. Get contact persons (EM/PM) and make sure they are being included in regular updates.
- **Schedule a kick off with the team** - with the PM, Design, DS and EM included. The kick off should address "the why" of your project first (this will be driven by the PM during the meeting), and include measurable milestones. Also look at "project charters" and best practices for charters which you can use for the kick off meeting. Make sure all stakeholders are involved in the kick off meeting.
- Ensure that feedback from stakeholders is given, and actively solicit it if needed
- Consult the payments stakeholder list for inspiration
- Aim for feedback/awareness from at least one person from each stakeholder group

1.2. Documentation & Collaboration

- **Setup and maintain the documentation** for the project. (See details in this section)
- **Add your project to the staffing doc & wiki**
    - Add it to the team staffing document (or update the details if already there)
    - Add to the Current Projects wiki page and to the Active projects section on the team page
- **Set up team collaboration tools**
    - *A JIRA/Phabricator board* - set one up for your team. Use of sprints/milestones is encouraged.
    - *A chat channel* - set one up for your team and you can set some of the quick links to the header. Invite all stakeholders (PM, DS, EM) to this channel.

    - *Email group* - set one up for your team to be able to send messages to all group members
    - *Google Drive Folder* - set one up for your project so all project-related documents can be found in one single place
    - Wiki page - link all the above on this page. A good example to follow for structure is the the Brazil Debit Cards project page.

1.3. Day to day project management

- **Find a good way to get frequent updates across the team,** making sure everyone has the context to do their work and help others on the team. For example different forms of standups or other ways of syncing up could work well. Do your research and choose a format you're comfortable with. A good tip is to track action items from updates and so a living doc with notes might be a good idea.
- **Demo progress** regularly. Your team is expected to show what progress you made regularly - even if it is not clearly visible progress, just under the hood. The bi-weekly team demo session is a great place to demo progress.
- **Follow project management** guidelines at a macro scale (high level) and decide on how you will manage the project on a micro scale (day to day).
- **Ensure good Phabricator usage** and basic "Phab hygiene". See details in this section. After a while this should be second nature on the team.
- **Encourage face to face information exchange** between teams during project ramp-up if there is collaboration across offices, or if working in a codebase that another team has familiarity with. Consider doing video kick off meetings, or whether in-person collaboration would help the project. Achieve velocity early, ideally in week one or two of the project.

2. Manage risks

- **Understand all parts of the project** at high level - from mobile to backend to business impact. If you're a mobile engineer, get familiar with the backend stack and the other way around and build a good relationship with the engineers on your "other" stack. If you are a backend engineer, get familiar with the release/build train process and mobile experiments. You will be the person with the most context on the project - make sure you know what the top issues on each stack are.
- **Create an inventory of knowledge** - List out the services and libraries that you'll touch, and make a note of team members' familiarity with them. If it's an unfamiliar codebase, *assume the worst* when giving time and risk estimates.
- **Call out risks** honestly as they arise - on standups, raising to your EM immediately as well as on the weekly meetings. Between project time, scope and people if one changes, we need to decide if and how we change the others. As a rule of thumb, the best time to call out a risk is when you first spot it. The second best time is now and not later.

- **Accountable for keeping the project on track** - across all the stacks. You own communicating the status of the project proactively - may it be off track or everything going well.

3. Communicate project status to stakeholders

- **Keep the team and internal stakeholders aware** of how the project is progressing via the weekly team meeting and demos. You should also consider sending out a weekly update email to make sure everyone is on the same page. As the engineering lead you own the internal team updates (keeping all internal stakeholders up to date). The PM owns the external update and comms. Be sure to clarify with your PM who updates what groups to avoid overlap.
- **Keep status tracking documents up to date** that the team and other stakeholders can get an idea of project progress. This will usually be the rollout section of the staffing document, the feature readiness checklist and any other developer confidence tool you might choose to use.
- **Help the PM identify any other stakeholders** who need to be aware of the project. On our team, PMs own communicating with stakeholders outside our team. You will likely have additional context on people who should know about the project - let your PM know and help your PM with communicating as needed.

4. Help the team focus and don't be afraid to delegate

- **Help engineers focus** on the next important thing. We need to go one step at a time towards the next shippable thing. Remind the team what milestone you need to ship next.
- **Delegate** where it makes sense. You can both delegate upwards (e.g. to your manager or PM) as well as to team members. You own a lot of responsibilities - it can be a win-win to `share these with some of your team mates. For example, someone else can be in charge of running standups, ensuring the Phabricator is up to date or presenting the demo. Be clear that you're delegating or asking for help and what your expectations are for the person taking this on.

5. Motivate the team

- **Do something fun** with your team. Take them out for lunch or dinner or beers or toasters.
- **Celebrate** the small wins, hitting milestones and people who help others.
- **Involve team members** in running of the project and be open and welcoming to suggestions coming from them - the project will only be successful if everyone is involved!
- **Over communicate milestones** within the wider team team. A product launch email to the team, the wider org or small things like @all mentions in chat are all great ways to celebrate larger or smaller milestones.

Checklist for first time projects

Using the tools below greatly increases the likelihood of project success. Use these tools if it is your first time running a project.

- **Project setup**
  - **Lead a kickoff meeting** with an agenda and all stakeholders present. Prepare for this meeting, getting feedback from the EM and other, experiences project leads.
  - **Break down milestones, assign estimates** to them. Usually done on one or more planning meeting, with the engineering team.
  - **Follow the RFC process**. Send out the intent for RFC, do the planning, put the RFC together, send it for review and follow up with approvers.
- **Regularly**
  - **Weekly update emails**. Send out weekly update emails to internal stakeholders. This email should have milestones, estimates and their status the very least. This should be sent at the very least to all team members working on the project, team EM & PM.
  - **Daily standup with an agenda & notes.** Use something like the sync notes template. Make sure the whole team and internal stakeholders (like PM, design) are present.
  - **Set weekly team goals**. At the beginning of the week, agree with the team what you aim to get done this week.
  - **Demo progress (bi)weekly.** At the end of the week, check if you made this progress via a demo. Demo even if you don't have anything visible: show at least the code to each other.
  - **Do a retrospective** at the very least, at the end of the project. You can do it every few weeks as well.

Resources

Documentation

Area	Notes
Wiki page	Create one on the team wiki page & kept up to date with resources. A good example to follow is the Brazil debit cards project.
Feature Readiness Checklist	Make a copy of this template and keep it up to date. This is a living template, so feel free to make modifications for your project and the template.

[Suggested] **Weekly Update Email**	Many teams use this template to let internal stakeholders know about the status. This is a great way to be transparent about the status of the project.
[Optional] **Developer Confidence Tracker**	An idea that can be used for micro level project management - see this template.
[Optional] **Timeline**	A simplified version of a Gantt chart that gives a visual idea of the project schedule and how (un)likely it is to meet the deadlines. See this template.

See the Project Management and Templates page for more resources you can use on your project.

Good JIRA/Phabricator Usage

Area	Notes
Have a ticket for anything a team member works on	All diffs need a ticket to land. So let's make sure to have them in place before or when
Clear & easy to understand tickets	Any engineer - even one from not your team - should be able to pick up work from the Phab board. All tickets should have a basic acceptance criteria. Simple ticket? Then the acceptance criteria will take no more than 30 seconds to write.
JIRA/Phab reflects work that people are doing currently	When someone is working on something, the ticket should be in progress, with the ticket assigned to them.
[Tip] Use Phabricator Milestones or JIRA Sprints	Smaller boards are easier to oversee than larger ones. Consider dividing your projects into smaller sections. Milestones are basically sprints within Phabricator - feel free to experiment using them. Sprints are similarly efficient in JIRA.

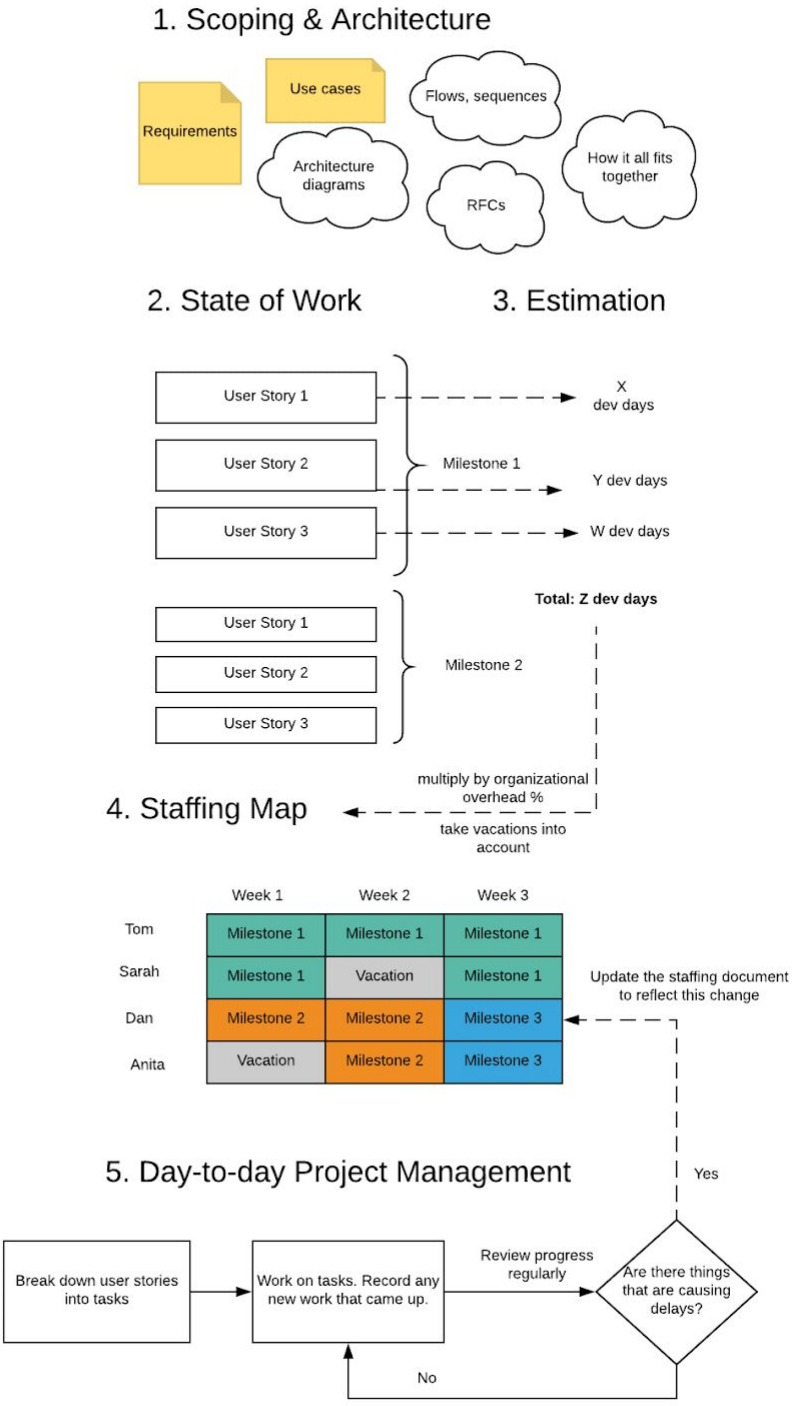
Project Management Methodology

We manage projects using a combination of two techniques.

- **Macro level project management** (high level) using the "Jellybean" tracker in the staffing doc. We scope, breaking into milestones, estimate using "dev days" and fill out the resource map (staffing doc) and keep it up to date.

- **Micro level project management** (day to day level). We do more granular breakdown and estimation & regularly check if we are on track and update the staffing plan (the macro level) anytime we see things that would impact the allocated time.

Here is a visual representation on how these two methodologies work together:



Macro level project management (high level)

The goal of this section is to make sure the team is not over committed and “book” time that will be spent on working on this project at the very least. This is a best effort call - and it’s expected it won’t be spot on, so don’t take too much time with this phase. We use a time based estimation on larger parts of the projects, apply the organizational overhead to (“fudge factor”) and map out time needed in the staffing doc, taking vacations, on-call and other project work into account.

For this first estimation, have the relevant members of the team in the room, as well as experienced engineers who can help estimate the effort better.

Step	Details
Break up the work into shippable milestones	If the project “feels” longer than two weeks, define some milestones we could ship in between. These will serve as checkpoints we are aiming to hit.
Define major work items within the milestones (user stories)	List all major work items that make sense to do by themselves. These tasks will be the user stories.
Estimate work items in ideal dev days	For each work item, estimate how many ideal dev days it would take to complete. An ideal dev day is a day without meetings or other interruptions, assuming all pre-requisites for the work are done. Do this estimation for each milestone.
Apply organizational overhead and map onto staffing doc	Apply the “fudge factor” or organizational overhead to the estimates. This is usually somewhere to 50% to 100%. Decide with the team how much non “pure” dev time you’re expecting to encounter. Now that we know how many days each part should take, given the team, book time in the staffing spreadsheet, marking different milestones with different colors. Here we can take vacations and oncall into account.

Micro level project management (day to day)

How you manage the project at a micro scale is more open ended. However, whatever methodology you use, be sure it fulfills the following:

1. **Breaks down user stories to smaller tasks.** With these tasks you and the team can see the day to day progress, and have a high confidence that you're working on the right thing.
2. **Feeds back risks and delays** to the macro level ("jellybean") planning. Simply put, if a milestone is at risk (either due to more work or slower progress), then you should be able to detect and flag this. Do some sort of check on this regularly - at least on a weekly basis. Anytime we think the current milestone could be delayed, we should shift the allocated time for the project at the macro level
3. **Uses Phabricator** as the source of task truth.

An example framework for Micro level project management

Here is a framework based on agile practices to use for the micro level. You are free to use another methodology or change parts of these - be sure to document it first though.

Setting up the project

Step	Details
Define reference tickets for estimation	We do estimations using story points that indicate complexity. As a first step, define reference tickets that we can relate to when estimating. If there is no mobile reference tickets list, please create one and link here. For backend, use the backend reference tickets list.
Create a Phab milestone for each project milestone	In the previous step we've defined the milestones. Open a Phab milestone subproject for each of these.
Add user stories to the milestone	For each user story that we've previously defined, create a task with a clear description and acceptance criteria. Note that user stories will not be estimated, only their subtasks. Mark the user stories in some distinct way.
Define tasks needed to achieve the user stories.	For each user story, list all the standalone things we need to do as Phab tickets. Use a whiteboard, UI designs etc here - make sure everyone has a clear understanding of what each task is. The title and description should be clear enough so that anyone (even outside the team) could pick it up.
Estimate all tickets and break up ones that are too big	Using planning poker, estimate all tickets as a team. Any ticket that takes more than 2 minutes, park it and pick it up later. Decide as a team what size of tickets should be

	broken up - if the estimate is too high, park those tickets as well and break them up into smaller pieces.
Setup a dev confidence tracker document	Create a document to track completed work, leftover work and new work added based on this template. This will be used to answer the question “are we on track to finishing by the time we estimated for?”

Day to day project management

Step	Details
Estimate an ETA for the milestone	Seeing the work broken down and the complexity, the team needs to commit to how many weeks they think the work is realistic. This will be the internal ETA of the milestone.
Organize a short, weekly progress review meeting with the team, PM & EM (e.g. after the standup).	**1. Record and estimate any new work** On this meeting, ask people what other work might need to be done to complete the current milestone. Add this work. Estimate all unestimated tickets at this point with the team. **2. Track the past week’s velocity and update the staffing doc** to reflect the new ETA based on progress. On this grooming, record what work has been done the past week in the burndown. Also record any new work that came up. If the progress indicates that the current milestone will take longer, adjust this in the staffing spreadsheet - just shift everything over.
New work discovered? Add a ticket, estimate it & track the added work.	Estimation can be done e.g. at the end of the standups. Don’t forget to add this new work to the developer confidence tracker document.
Ready to roll out?	Make sure that there is an experimentation and rollout plan in place. DS & PM own this; ping them in case there isn’t a written plan. Once a feature is shipped to the App Store, update the project tracker with the app versions, so that it’s easier to trace.

Further Resources

- Agile Project Estimating - blog post by Steve Thomas