Key Takeaways

The Effective Engineer by Edmond Lau

Time is our most finite asset, and leverage, the value we produce per unit time, allows us to direct our time toward what matters most.

$$\text{Leverage} = \frac{\text{Impact Produced}}{\text{Time Invested}}$$

Adopt the Right Mindsets

Focus on High Leverage Activities

- **Use leverage to measure your engineering effectiveness.** Focus on what generates the highest return on investment for your time spent.
- **Systematically increase the leverage of your time.** Find ways to get an activity done more quickly, to increase the impact of an activity, or to shift to activities with higher leverage.
- **Focus your effort on leverage points.** Time is your most limited asset. Identify the habits that produce disproportionately high impact for the time you invest.

Optimize for Learning

- **Own your story.** Focus on changes that are within your sphere of influence rather than wasting energy on blaming the parts that you can't control. View failures and challenges through a growth mindset, and see them as opportunities to learn.
- **Don't shortchange your learning rate.** Learning compounds like interest. The more you learn, the easier it is to apply prior insights and lessons to learn new things. Optimize for learning, particularly early in your career, and you'll be best prepared for the opportunities that come your way.
- **Find work environments that can sustain your growth.** Interview the people at the team or company you're considering. Find out what opportunities they provide for onboarding and mentoring, how transparent they are internally, how fast they move, what your prospective co-workers are like, and how much autonomy you'll have.
- **Capitalize on opportunities at work to improve your technical skills.** Learn from your best co-workers. Study their code and their code reviews. Dive into any available educational material provided by your company, and look into classes or books that your workplace might be willing to subsidize.
- **Locate learning opportunities outside of the workplace.** Challenge yourself to become better by just 1% a day. Not all of your learning will necessarily relate to engineering skills, but being a happier and better learner will help you become a more effective engineer in the long run.

Prioritize Regularly

- **Write down and review to-dos.** Spend your mental energy on prioritizing and processing your tasks rather than on trying to remember them. Treat your brain as a processor, not as a memory bank.
- **Work on what directly leads to value.** Don't try to do everything. Regularly ask yourself if there's something higher-leverage that you could be doing.
- **Work on the important and non-urgent.** Prioritize long-term investments that increase your effectiveness, even if they don't have a deadline.
- **Reduce context switches.** Protect large blocks of time for your creative output, and limit the number of ongoing projects so that you don't spend your cognitive energy actively juggling tasks.
- **Make if-then plans to combat procrastination.** Binding an intention to do something to a trigger significantly increases the likelihood that you'll get it done.
- **Make prioritization a habit.** Experiment to figure out a good workflow. Prioritize regularly, and it'll get easier to focus on and complete your highest-leverage activities.

Execute, Execute, Execute

Invest in Iteration Speed

- **The faster you can iterate, the more you can learn.** Conversely, when you move too slowly trying to avoid mistakes, you lose opportunities.
- **Invest in tooling.** Faster compile times, faster deployment cycles, and faster turnaround times for development all provide time-saving benefits that compound the more you use them.
- **Optimize your debugging workflow.** Don't underestimate how much time gets spent validating that your code works. Invest enough time to shorten those workflows.
- **Master the fundamentals of your craft.** Get comfortable and efficient with the development environment that you use on a daily basis. This will pay off dividends throughout your career.
- **Take a holistic view of your iteration loop.** Don't ignore any organizational and team-related bottlenecks that may be within your circle of influence.

Measure what you want to Improve

- **Measure your progress.** It's hard to improve what you don't measure. How would you know what types of effort are well spent?
- **Carefully choose your top-level metric.** Different metrics incentivize different behaviors. Figure out which behaviors you want.
- **Instrument your system.** The higher your system's complexity, the more you need instrumentation to ensure that you're not flying blind. The easier it is to instrument more metrics, the more often you'll do it.
- **Know your numbers.** Memorize or have easy access to numbers that can benchmark your progress or help with back-of-the-envelope calculations.

- **Prioritize data integrity.** Having bad data is worse than having no data, because you'll make the wrong decisions thinking that you're right.

Validate Ideas Early and Often

- **Approach a problem iteratively to reduce wasted effort.** Each iteration provides opportunities to validate new ideas. Iterate quickly to learn quickly.
- **Reduce the risk of large implementations by using small validations.** Invest a little extra effort to figure out if the rest of your plan is worth doing.
- **Use A/B testing to continuously validate your product hypotheses.** By incrementally developing a product and identifying what does and doesn't work, you increase the probability that your efforts are aligned with what users actually want.
- **When working on a solo project, find ways of soliciting regular feedback.** It may be easy and comfortable to keep working in a silo, but you run the huge risk of overlooking something that, if spotted early, could save you lots of wasted effort.
- **Adopt a willingness to validate your decisions.** Rather than making an important decision and moving on, set up feedback loops that enable you to collect data and assess your work's value and effectiveness.

Improve Your Project Estimation Skills

- **Incorporate estimates into the project plan.** These estimates should be used as an input to decide whether delivering a set of features by a certain date is feasible. If it is not, they should lead to a conversation about whether to change the feature set or the delivery date. Don't let a desired target dictate the estimates.
- **Allow buffer room for the unknown in the schedule.** Take into account competing work obligations, holidays, illnesses, etc. The longer a project, the higher the probability that some of these will occur.
- **Define measurable milestones.** Clear milestones can alert you as to whether you're on track or falling behind. Use them as opportunities to revise your estimates.
- **Do the riskiest tasks first.** Reduce variance in your estimates and risk in your project by exploring the unknown early on. Don't give yourself the illusion of progress by focusing first on what's easy to do.
- **Know the limits of overtime.** Many teams burn out because they start sprinting before they're even close to the finish line. Don't sprint just because you're behind and don't know what else to do. Work overtime only if you're confident that it will enable you to finish on time.

Build Long-Term Value

Balance Quality with Pragmatism

- **Establish a culture of reviewing code.** Code reviews facilitate positive modeling of good coding practices. Find the right balance between code reviews and tooling to trade off code quality and development speed.
- **Invest in good software abstractions to simplify difficult problems.** Good abstractions

solve a hard problem once and for all, and significantly increase the productivity of those who use it. But if you try to build abstractions when you have incomplete information about use cases, you'll end up with something clunky and unusable.
- **Scale code quality with automated testing.** A suite of unit and integration tests can help alleviate the fear of modifying what might otherwise be brittle code. Focus on ones that save the most time first.
- **Manage your technical debt.** If you spend all your resources paying off interest on your debt, you won't have enough time left to work on new things. Focus on the debt that incurs the most interest.

Minimize Operational Burden

- **Do the simple thing first.** Simpler systems are easier to understand, extend, and maintain.
- **Fail fast to pinpoint the source of errors.** Make debugging easier by not masking your errors and by not deferring failures until later.
- **Automate mechanics over decision-making.** Aggressively automate manual tasks to save yourself time. At the same time, think twice before trying to automate decision-making, which tends to be hard to get correct.
- **Aim for idempotence and reentrancy.** These properties make it easier for you to retry actions in the face of failure.
- **Plan and practice failure modes.** Building confidence in your ability to recover lets you proceed more boldly.

Invest in Your Team's Growth

- **Help the people around you be successful.** The high rungs of an engineering ladder are reserved for those who make their co-workers more effective. Moreover, the success of those around you will also carry you along.
- **Make hiring a priority.** Keep a high hiring bar and play an active role in growing your team.
- **Invest in onboarding and mentoring.** The more quickly you can ramp up new team members, the more effective your team will be. The more effective your team, the more freedom you have to tackle different projects.
- **Build shared ownership of code.** Increase your bus factor to be greater than one so that you're not a bottleneck for development. This will give you the flexibility to focus on other high-leverage activities.
- **Debrief and document collective wisdom.** Reflect on projects with team members, learn what worked and what didn't work, and document and share the lessons so that valuable wisdom doesn't get lost.
- **Create a great engineering culture.** This will help you be more productive, streamline decisions, and recruit other strong engineers. You build a great culture by fostering the same habits you need to effectively deliver impact.