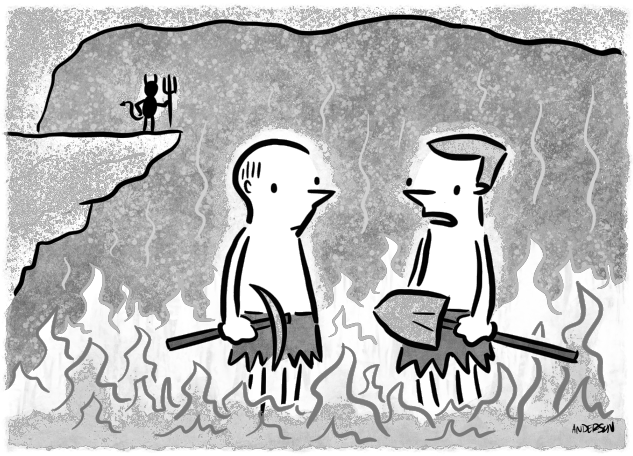
Individual efficiency vs administrative efficiency

by Jason Cohen on August 11, 2024



"Don't get me wrong, I'm no fan of the heat or the backbreaking toil. It's the corresponding paperwork that gets me."

Everyone has their own favorite note-taking app: Notion versus Google Docs versus Apple Notes versus OneNote versus Obsidian. Dropbox Paper shrieking "what about me?" into the void. I wrote this article in Bear. (No relation.)

From the individual's point of view, the optimal company policy is to allow everyone to use whatever app they like. It's both autonomy and mastery. Everyone's comfortable and efficient.

From the team's point of view, however, this is a bad policy. I don't want to learn how pseudo-database-table-thingies work in Notion, but Bear doesn't support sharing notes between people. Efficiency for the individual reduces efficiency for the team.

Beyond the team, there are company-wide administrative concerns. Once we're sharing notes, who has access to those notes? And who decides who has access to which notes? What if company secrets are in those notes? What if customer PII are in those notes? Is note-authentication linked to the corporate identity system, so that when someone leaves the company they automatically no longer have access to the notes? Can the central IT team back up the notes? Can the central security team audit the notes? Since the notes are shared, they probably bounce through a server somewhere; is that secure? Administrative requirements have a negative effect on the efficiency of the team, and certainly on the individual.

While you might be laughing because "this is why big companies are dumb and slow," and indeed startups often win exactly because they don't have to be "dumb" and "slow" like this, it's also why the big company will add more ARR today than the small company will add in the next few months. (A result that is neither dumb nor slow; both the benefits and drawbacks are due to scale.)

So, when should a policy optimize for the efficiency and happiness of the individual, and when should it optimize for the team, or the company?

In search of efficiency

Imagine a scenario in which 10 people each receive a meal. Each meal is unique, and each person has unique food preferences.

A maximally efficient way to *administer* the meals is to dole them out randomly. Nothing to manage or track. This is also *fair*—a desirable quality in policies—in that everyone is (mis)treated identically. Even so, this is obviously suboptimal for the recipients.

Next, everyone looks around the room. Some person `P` sees another person `Q` holding a plate that `P` would prefer to have. Perhaps `Q` is thinking the same thing about `P`! In that case, it's wise for them to trade plates; both are happier. Or the scenario where person `P` would be happier if they traded with `Q`, and `Q` doesn't care either way; they also should trade. Trades won't happen if one person would be less happy; that person would simply refuse the transaction.

If we allow these sorts of trades to proceed until no further trades are possible, the system reaches a state called "Pareto-Optimal." Formally, this is a state where any transaction would result in at least one party being worse off than they currently are.

"Pareto-Efficiency" is not the only kind of efficiency, nor does it necessarily reach a "maximally efficient" state by some measures. Consider the goal of "maximizing total happiness," a kind of utilitarianism, and apply it to the following scenario, where persons `P` and `Q` rate their plates on a 1-10 scale, where 10 is the best:

Person	`P`'s Plate	`Q`'s Plate
`P`	`1`	`10`
`Q`	`5`	`9`

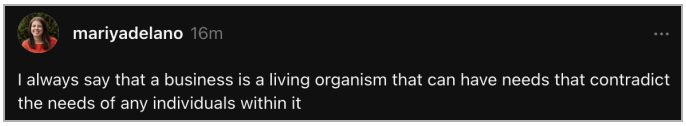
`P` would very much like to trade, but `Q` would not; this is Pareto-Optimal. But if they did trade, total happiness would increase, because currently total happiness is `1 + 9 = 10` whereas trading would result in `10 + 5 = 15`.

So, should we *force* them to trade, pissing off `Q` for the greater good?

Clearly there won't be a single answer to defining "efficiency" or "fair policy." Nevertheless, with this backdrop as our guide, and taking up practical considerations arising in real companies, there are many actionable things we can do to make policies more fair, and people more efficient, and even more happy.

When to standardize

Under what conditions do the benefits of standardization outweigh the penalties on individual efficiency?



Maximize the individual; satisfy administration

Individual autonomy leads to everything good: The proverbial "win-win" of both happiness and efficiency. Therefore, we should maximize individual autonomy. Administrative requirements should take precedence only with reason, such as a current problem with harms we can easily identify, or because the law requires it, or because our values dictate it, or because important jobs

(like those of the executive team or the security and IT teams) cannot be done without it. Use the Satisficing vs Maximizing Framework to navigate this dynamic.

"I'm sorry, Jim, I'm afraid this is only a win win situation.

Minimize the scope of standardization

Most developers agree that code-formatting should be standardized, at least within a single team. One implementation is to require everyone to use the same IDE, which in turn enforces the formatting rules. But that's over-scoping the solution, because many IDEs are capable of enforcing the same formatting rules. The solution is to mandate the *rules*, and allow developers to use any system that enforces those rules. Sure, some IDEs might make that easier than others, but if a developer really wants to use a different editor, and will abide by the rules (perhaps with an external tool and custom automation that the developer maintains), then individual efficiency has been preserved, while the team enjoys the benefits of standardization.

Use standardization in one area to create individual autonomy in other areas

It would easier for the IT department if everyone used the same laptop, with the same operation system, and also used the same smart phone and same tablet. But people have their own smart phones, and would prefer not to carry two. So, as we create the "note-taking policy" mentioned earlier, we could constraint ourselves to consider only those note-taking apps that work well across *all* of Windows, Mac, iOS, and Android. Perhaps we also add the requirement that the note-taking app must have a fantastic web-UI (so that any laptop with any operating system has a good experience), and that it have an at-least-4-star app for both iOS and Android devices. That will narrow the field of possible note-taking systems, but in doing so, we preserve the individual's choice of device. While we're at it, perhaps we have requirements for supporting the visually-impaired and for a wide variety of languages; this additional constraint again increases happiness and effectiveness for individuals.

Standardize on outcomes, not on implementations

Goal-setting and metrics-reporting are common examples of tension between the needs of administration and the needs of operators. The only way for the CEO to keep track of the business is to have a consistent summary of activities, metrics, and how departments and major initiatives are pacing to expectation. But Sales operates very differently from Product; Marketing operates differently from Support. Not just in the obvious ways like which metrics are tracked, but in how work is scheduled, how impact is quantified, and planning cycles. Product might plan tactics every two weeks and strategies annually, yet metrics and goals are reported monthly. Sales typically runs on a monthly cadence and therefore has no problem reporting and reacting monthly. Product doesn't want to ship new things in December; Sales doesn't want any meetings in the final few days of the month. Everyone uses different systems-of-record—Jira vs Zendesk vs Hubspot vs Segment vs Salesforce; there is no natural place for goals and metrics to live. It would definitely be incorrect to force everyone to use a single tool to manage all their work. Therefore, the right solution is to standardize on how goals and metrics are *reported upward*, but explicitly *not* standardize on how each department *operates*. (And to implement a KPI philosophy accordingly.)

"I *am* multi-tasking! I've done this report three or four times already!"

Policies should describe explicit benefits for individuals, not just for administrators

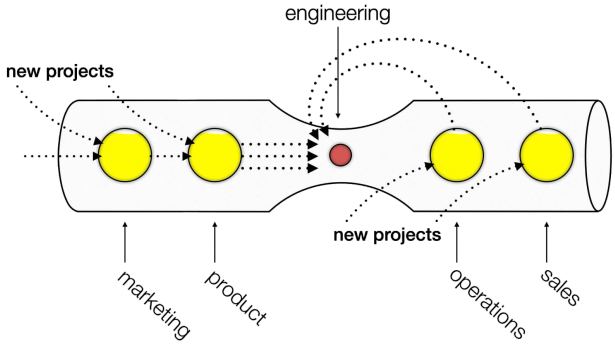
No one creates a policy with an explicit intent of causing pain; it happens by accident. If the policy dictates only "what must be done," and not "how it benefits everyone, in different ways," it is likely that the latter wasn't sufficiently considered while the policy was being made. Require policies to have sections that detail how this is beneficial for various parties; if any piece of that section is found wanting, that means our policy isn't good enough yet.

Look for cases where local optimization creates problems that global optimization can solve

Sometimes, optimizing of one component of a system harms overall system performance. This often happens in marketing funnels. One team is responsible for bringing more traffic to the site, so they optimize and succeed. But this new traffic turns out to be low-quality—perhaps that's why it was so easy to generate—so the "home page → purchase page" conversion rate plummets, harming another team's numbers, and making it harder for them to improve, because now they're wading through garbage traffic rather than improving engagement with quality traffic. This is a moment when global optimization should take precedence over local "efficiency." Another example is in assigning tasks, where people tasked at 90% utilization leads to catastrophic failures, to say nothing of unhappiness and burn-out.

Accept sub-optimization in areas that are not the primary bottleneck

In the Theory of Constraints, a system's throughput is limited by a single component (the "bottleneck"), and therefore optimizing other components does not increase total throughput (and in fact can ironically decrease total throughput). Among the many techniques for solving the bottleneck is that we should use people who are not part of the bottleneck to unblock or delegate tasks from within the bottleneck, even if these new recruits are far less efficient at executing those tasks, or if the tasks are menial. This is definitionally sub-optimal from the point of view of all teams who aren't the bottleneck, but the result is higher throughput for the entire organization, and therefore it is the right choice. Encourage everyone to participate in diagnosing the problem as well as inventing the solution, so they realize they're working for the greater good, not working for an ignorant organization.



Yellow activities should be subordinated to help with flow in the red activity.

Consider whether local inefficiency is temporary or permanent

All change creates temporary inefficiency, as people and systems acclimate to the change. People often dislike change. It's a well-documented rule of design that any time you change the UX of software, many existing customers will complain simply because it's different. When imposing standardization, we nearly always create tem-

porary inefficiency; assuming the standardization is valuable, we should just accept this cost. But when the inefficiency is permanent, we need correspondingly higher conviction that the standardization is worthwhile.

Document the cases where it's clearly better to (not) standardize

Product-line strategy is shared across teams, so it should be centralized. Communication across the entire company is better as a single report than as five disjointed reports. Conversely, a team's inside jokes only work when they stay inside the team, and no one is harmed when someone wants to use a family photo as their laptop's desktop background. Writing these categories down keeps us honest—not allowing standardization to encroach where is has negative value, and also agreeing that administrative needs should be paramount in certain areas.

Proactively encourage Pareto-Efficiency

In the "meals" example, we claimed that "random assignment" was efficient for management, but also that we could arrive at a better result if people are allowed to trade. Furthermore, trading does not create any problems for management. Therefore, not only should trading be *allowed*, it should be *encouraged*; everyone is better off, even management. Get creative about how people can self-organize within the constraints of the policy. For example: Trade time; some people help another team accomplish one of their goals faster now, and later the reverse happens, all without "management" getting involved. (Also "trading" isn't the only technique.)

Use global-optimization explicitly, and sparingly

In the "meals" example, we saw that global-optimization can be worse for individuals. It's unclear whether that's "more fair" or "better." And yet, global-optimization *sounds* like the smartest thing to do, and indeed it often is. When it really is smartest, we should be explicit about why that is, what global effect we are maximizing, and why it's so worthwhile for the collective good. If you're going to give me a meal I don't want, at least tell me why the higher purpose makes it a worthy sacrifice.

All these are variations on the key idea: Individual autonomy should be our paramount goal, and thus our default. But often local optimization does not lead to global optimization, and the latter is what we should all want for our organization.

Since there are many legitimate times when administrative needs *should* supersede the individual, we should always be open to them, but be explicit, be thoughtful, justify the decision, and keep the individual in mind.