CASE STUDY

How We Built a Clear Framework for Evaluating Developer Performance in Three Months — After 10 Years Without Formal Processes

My role: Ghostwrote the complete article for REES46 CEO Mikhail Kechinov.

Original publication: vc.ru, Development | July 18, 2024

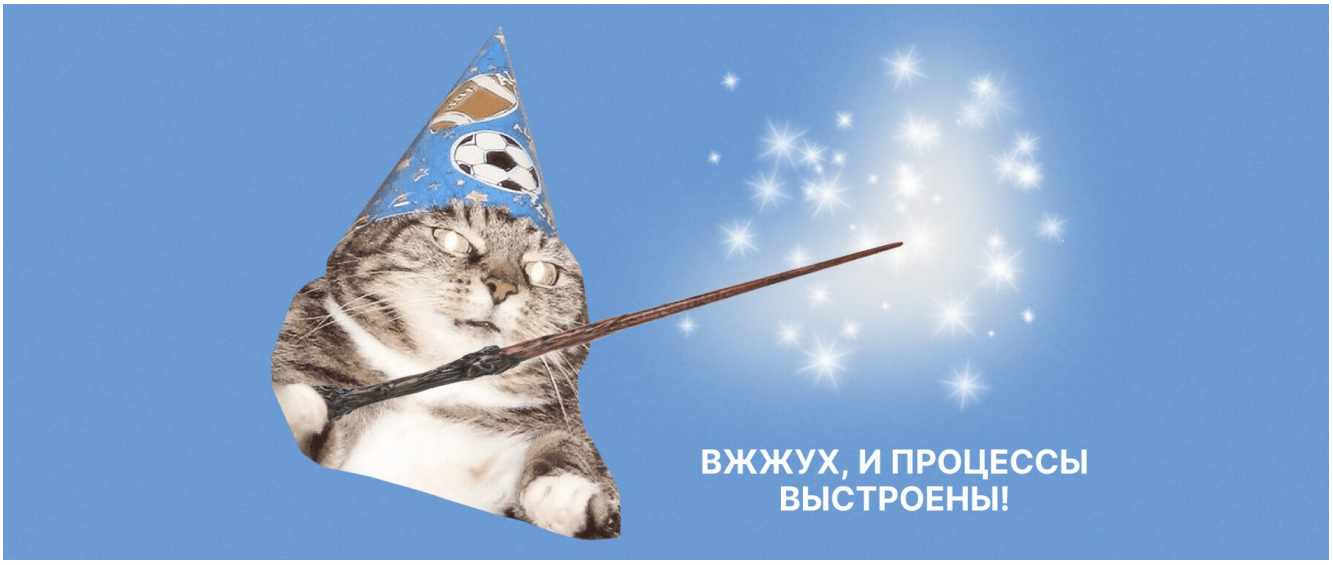
Original byline: Mikhail Kechinov, CEO of REES46

Russian original | Portfolio overview

English portfolio adaptation of the July 2024 Russian article. The narrative reflects the company's processes at that time. Selected original illustrations are included with English captions.

Hi, I'm Mikhail Kechinov, CEO of REES46, a tech company that builds marketing automation software. We work with any online business that has more than a hundred customers.

I recently came across a heated discussion here on vc.ru: is it okay for someone to hold down two or more full-time jobs? Here's why I think it isn't — and how we now measure our developers' performance without OKRs, PIPs, or even an HR department. We simply put proper processes in place. Finally.



Original Russian illustration. English text: "A wave of the wand — and the processes are in place!"

Why "results, not hours" doesn't work for us

A few months ago there was a lively thread here. It was about an account manager in the US who held two jobs while working just 40 hours a week — half a day for each employer — and earned $200K a year. This wasn't an arrangement either employer had agreed to; it was time management designed so that nobody suspected there were two jobs. Is that okay?

It's okay only if both the employee and the employer are just going through the motions eight hours a day. But jobs like that barely exist anymore — especially in tech. People can take on side projects, sure, but on their own time. Nights, evenings — go for it. Call me old-school, but the working day is ours: we pay a salary for those eight

hours. You promise to deliver eight hours' worth of results in exchange for money, and we count on all eight of those hours.

Finished a task early? Great — grab another one. We never run short of tasks.

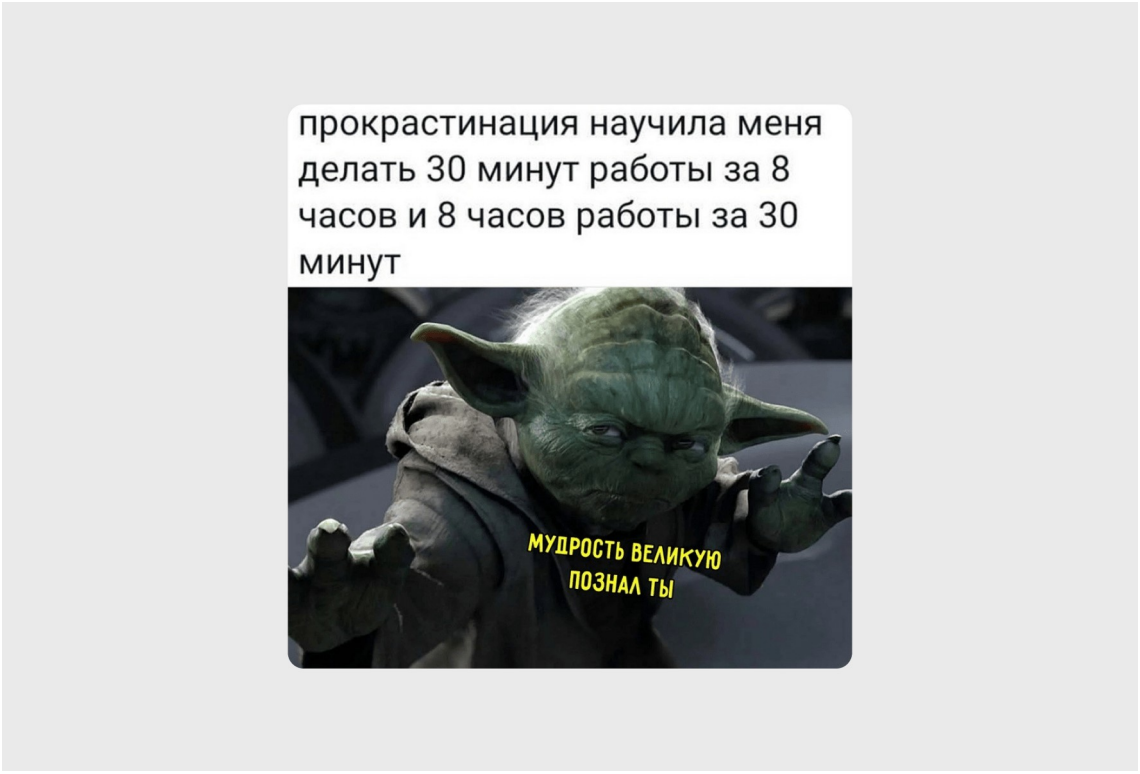
I'm sure most of my people do some freelancing on the side. Good for them — variety keeps things interesting. As long as it doesn't come at the expense of our projects. But holding more than one full-time job? That means your employers are either clueless and can't tell you're putting in three hours instead of eight, or they just don't care. Either way, you won't build a career with management like that. And as employers, we're not comfortable with that.

So what does "normal" look like?

When there are no KPIs and nobody is keeping track of what you do, it's tempting to do nothing — for that particular employer — and work on your pet projects instead. Or whatever else. Which raises the question: how do you measure what a normal pace of work looks like?

Large companies usually have full-blown performance management systems. Google has OKRs (Objectives and Key Results): inspiring objectives plus measurable key results that show how well those objectives are being met. Amazon has PIPs (Performance Improvement Plans) for underperformers — people whose output raises questions. A PIP comes with measurable goals, extra resources such as training or mentoring, and regular check-ins.

But rolling all that out at a 40-person tech company isn't easy. First, it takes serious time and resources. It's an entire system run by people whose job is to motivate the team and keep the culture healthy. That's at least ten more hires — 25% of our headcount. We're not there yet. We don't even have an HR person! Second, small companies often need more flexible, adaptive ways to evaluate performance.



Original Russian meme. English text: "Procrastination taught me to do 30 minutes of work in eight hours, and eight hours of work in 30 minutes." Yoda: "Great wisdom, you have learned."

When the team is small, you can generally see who's doing what and how long it should take. But once you're closer to 50 people than to 10, you need some way to measure productivity. And there are two things to keep in mind:

- The bigger the team, the longer any single task takes. A task one developer can do in two hours will take three people — a CTO, a PM and a developer — two days. I once built the website for one of our Hackdays (I run hackathons in St. Petersburg from time to time) in six hours. A team of three developers, a designer, a copywriter, a QA engineer and a project manager would have needed about two weeks for exactly the same job. Flip it around, though, and when you're dealing with a hundred tasks instead of one, processes and delegation speed things up dramatically.
- Two developers can have equally deep experience solving very different problems. What's a five-minute job for one is unfamiliar territory for the other. That said, if a five-minute task takes two weeks and the person never once asked anyone for help, the task isn't the problem.

What worked for us

We went through every stage of grief before accepting a simple fact: you do need KPIs. We trusted, we waited, we helped people get to know the product, we held endless meetings explaining how the system works, we delegated responsibility — and we got good results. Just very, very slowly.



Having arrived here, we decided to try building an actual system. Original Russian meme: "How about introducing KPIs?"

In 2024, we decided to change everything and pick up the pace — significantly. Here's what we found works for speeding up delivery and measuring results.

Agile in practice

Tasks are now grouped by context into milestones, and each milestone gets an engineer who owns it.

We chose Kanban for task management, because we always have an order of magnitude more tasks than developers.

At the end of every week we put together a sprint for each engineer, based on feedback from the product team. In other words, we always settle priorities — new features or bug fixes — before anyone starts writing code.

One Agile practice that surprisingly few people know about (or remember) is Extreme Programming. We've adopted it and use it quite actively. Instead of layering on more quick fixes, we refactor the relevant part of the codebase. It ends up cheaper, if not always faster, than a batch of new patches — and it always pays off in the long run.

> Our marketing team asked us to translate this bit into plain language — just in case. Here's what they came up with: "Instead of shipping more quick fixes, we refactor (improve the internal structure of the code) in a specific area. It costs less in the end, even if it isn't always faster than fixing bugs one by one. And over time, this approach always turns out to be the better investment."

Regular feedback

To collect feedback, we use automated daily check-ins instead of calls, so developers can report what they worked on yesterday and what they're tackling today.

At the end of every working day, the team also pushes their branches and opens draft pull requests, committing daily until the task is done.

We also teach the team to be self-sufficient and show them how — and in what format — to communicate where a task stands. Some tasks are especially urgent, and to avoid stressing out other departments (whose clients are waiting for updates), we came up with a simple rule: at the end of each day, post the current status and what's left to do right in the task. It's convenient and easy to follow.

Any engineer, product manager, account manager — any employee, really — can book a one-on-one with any of the leadership team, whether it's the company owner or the CTO. That keeps information flowing freely across the company and helps us head off breakdowns before they happen.

Transparent processes

Development lives in GitHub, while the business side lives in JetBrains Space, so we needed a convenient — and above all transparent — way for departments to work together. We built a few microservices and a couple of bots that sync data from Space to GitHub and back.

Every task has a status, such as To Do, In Progress, Review or Docs. Once a task passes review and testing, it moves to Docs, meaning our technical writer picks it up and updates the documentation so it's always current.

Here's how task sign-off works:

1. Before submitting a task for review, the developer must post proof in the pull request that it was completed exactly as the business specified — a screenshot or a screen recording.
2. Reviews move through three stages: a senior developer, the team lead, and then the CTO. This keeps release incidents to an absolute minimum.
3. Once the task passes review, it moves to Testing, where the changes are checked in a staging environment isolated from production.

Tasks often get stuck in Review or come back from testing marked Returned. Those get top priority, because letting them drag on inevitably means losing context — and a basic disconnect between the business and the developers, where nobody remembers what the task was or how it was supposed to work.



Original Russian meme. The ship represents how much knowledge of the REES46 platform a developer needed to complete tasks in 2021; the small excavator represents how much was needed in 2024.

It’s already clear these simple rules have moved us forward a lot in terms of speed. Onboarding used to take months, and some tasks took weeks to get done (or never got done at all). Now new hires pick up tasks on their very first day — and start adding value right away. And yes, they really do work eight hours a day — just like it says in their employment contract.