

Codility_

Codility Survey Findings

Developers that code together feel success more often at work



Overview


Working in silos in engineering teams can be detrimental to knowledge-sharing, productivity, and even feelings of job satisfaction. This is because teams that are structured to be agile, collaborative, and communicative can impact how often a developer feels success — which has a domino effect on business performance. The quicker and more often a team is able to work together to ship new code, the more a company can use new features to advance a product or service within the marketplace.

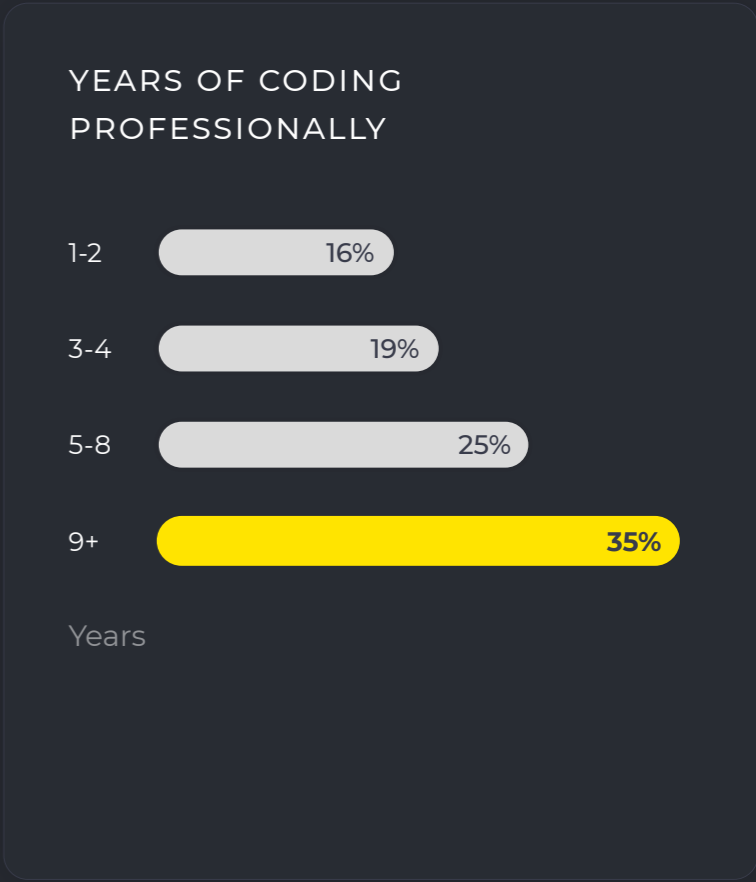
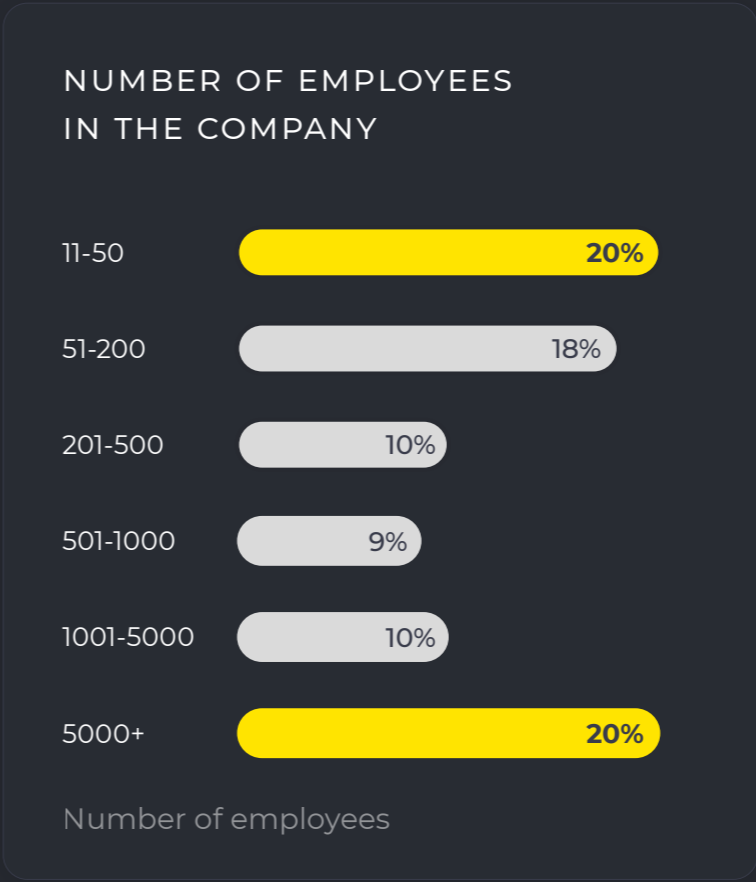
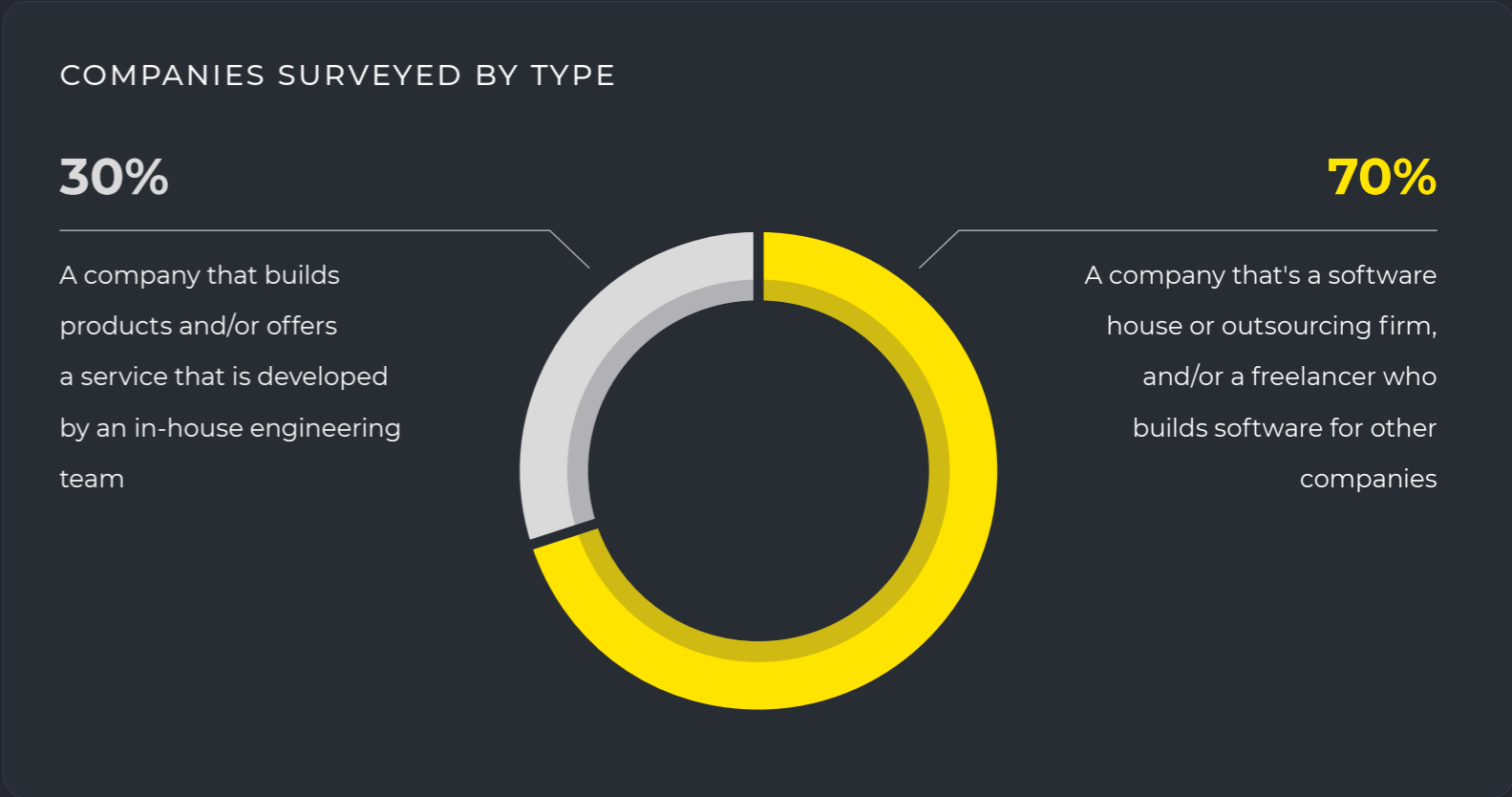
Findings from [Codility's Developer Survey](#) suggest that teams that work in test driven development (TDD) and pair programming see higher rates of feelings of success — and they deliver ideas to implementation faster.

*TDD is a programming practice that requires unit tests to be written before the code they are supposed to validate.
* Pair programming is an Agile technique in which two programmers work together at one workstation. One writes code while the other reviews each line of code as it's typed in.

- IN THIS REPORT WE'LL COVER:**
- A good approach to testing code elevates team performance
 - Pair programming accelerates delivery without sacrificing quality
 - Best practices for structuring your team

“Everything you do should reflect the kind of culture that you agree that you want.”

Adam Bermingham
Engineering Lead, 



Developers that code together feel success more often at work

A good approach to testing code elevates team performance

34%

Only 34% of developers test first even though they declare shipping code faster than devs that test after, or don't test at all.

Not only are they delivering code faster, but they also report feelings of success more often: 53% of developers who experience feelings of success on a daily basis are also developers working in TDD.

An explanation is that there's a quick feedback loop when code is immediately tested by previously developed tests. If the test fails, you can easily go back and improve the code — this is a good safety net for teams going back and fixing older pieces of code. If it passes, the “green light” often prompts positive reinforcement.

What this means for business leaders is that there's a huge opportunity to optimize feedback loops at a team level. According to [Marek Kirejczyk, CEO of EthWorks](#), “short feedback loops benefit both the company and the individuals at hand. This is the best way to improve job satisfaction and reduce burnout. When teams have less deployment issues, then companies can build so much more.”

Interestingly, industry experts from DevOps Research and Assessment (DORA) [found](#) that the fastest teams to deploy to production are also bringing the best quality code, and that teams should practice TDD to improve overall code design and readability.

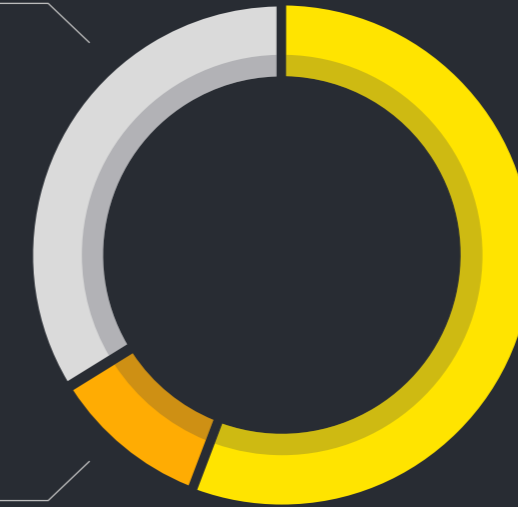
APPROACH TO TESTING

34%

We use TDD

10%

We don't really test



56%
We test after

APPROACH TO TESTING VS. TIME TO IMPLEMENTATION

How long does it take your team to take something from an idea to ready for implementation?

Approach to testing

Test first (TDD)



● Up to a week

We test after



● Around 2 weeks

● A month or longer

We don't really test



Proportion of respondents (%)

Benefits of TDD



Delivery of ideas to the implementation phase is faster, and can be executed within a few days



Anyone can adopt TDD as a mode of programming because it is not specific to any industry or company size



There's a chance that individuals feel success more often as a result of a quicker feedback loop



Sense of success contributes to overall well-being and quality of code, which can really shape an engineering team's dynamics

Challenges of TDD



Not widely adopted, which makes some teams skeptical to implement the technique



More typical for smaller teams (<3 developers) which means that it could be more challenging to adopt this practice with larger teams



There's little existing research available on the impact of TDD and how it might impact team performance



Quick feedback loops can lead to negative reinforcement because developers might get discouraged if they see that their code does not pass the test

Pair programming accelerates delivery without sacrificing quality

15% Only 15% of developers will pair program at least once a week even though these teams also boast faster delivery, with 60% of users declaring that they move to the implementation phase within a week.

Similar to devs working in TDD, those that pair program also report feelings of success daily (45%) compared to non-users (17%). It could be that when developers code together to achieve tasks, there's a heightened sense of progress.

Survey results show that pair programming users were more aware of their promotion requirements. An explanation is that these developers are already working

on soft skills development, transferring knowledge, and supporting team members — which are all behaviors typical for an Engineering Manager.

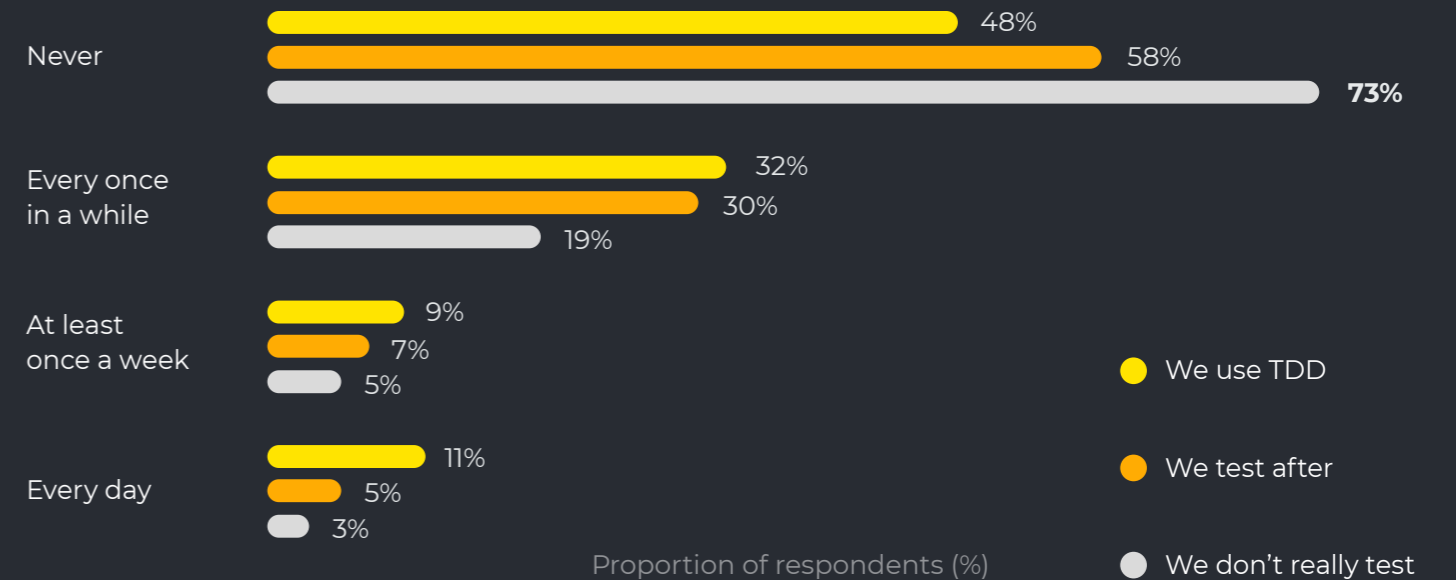
Senior Director of Engineering at Malwarebytes, [Darren Chinen](#), sums it up perfectly: "...sometimes you can't afford to hire the best, but you can afford to develop them into world-class engineers. By giving each individual professional security [by encouraging them to work on their skills together], they became more confident and contributed more."

Developers who don't pair program often also don't test code (73%). Among those who pair program at least once a week, there are more cases of test driven rather than test after development. This seems intuitive since these modes of work were introduced at a similar time, and often turn out to be complementary to one another.

PAIR PROGRAMMING VS. APPROACH TO TESTING

What is your team's approach to testing?

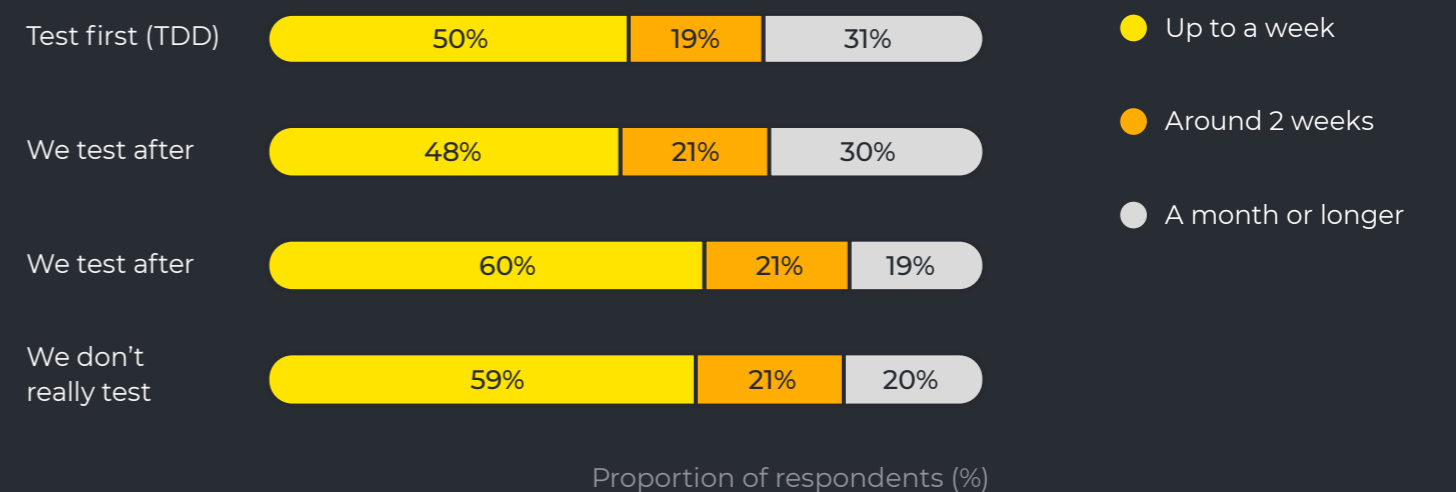
How often do you use Pair Programming?



PAIR PROGRAMMING VS. TIME TO IMPLEMENTATION

How long does it take your team to take something from an idea to ready for implementation?

How often do you use Pair Programming?



Benefits of pair programming



Delivery of ideas to the implementation phase is faster, and tends to have fewer bugs



Higher feelings of success contribute to the overall well-being of an engineering team, and can really change a team's dynamics



Anyone can pair program as it is not specific to any industry or company size



Devs that pair program are more aware of their promotion requirements, which may contribute to significantly higher job satisfaction

Challenges of pair programming



Not widely adopted, which makes some teams skeptical to implement the technique



Poorly applied pair programming can lead to double the costs since it involves two devs working on a piece of code instead of one



Programming needs to be done “out loud” so it could be difficult for introverts or those that aren't comfortable in their work setting



It could take time to figure out how many hours of pair programming per week is sustainable — we suggest starting with 2 hours per week

Best practices for structuring your team

01 Start with size

Sometimes the quantity of developers on a team is just as important as the quality of developers. The optimal team size is between 3 to 8 developers — this has also been validated by the [Scrum Guide](#). Broadly speaking, teams should be small enough to remain nimble and large enough to complete significant work within a Sprint*. For engineering managers this means that they

should have time to coach and nurture skills outside of aligning the team on strategic business goals. Organizational design is make-it-or-break-it when it comes to the future of a team. Employees churn when teams aren't structured right — on the other hand, teams that work well together can accomplish great things.

02 Prioritize online learning and development

Developers are continuous learners who prefer to stay sharp with digital learning tools — [90% admit to learning technologies outside of their formal education](#). At the same time, 30% of devs said that their employers don't support them in improving their professional skills. This discrepancy might be one of the reasons behind the significant drop in job satisfaction that can become

apparent about a year after joining a company. The most common practice for employers who do support professional development is providing employees with a growth budget. [Vice President of Product Engineering at Avature, Matias di Tada](#), tells his team what to do but ultimately "lets them figure out how to do it. You have to really trust them. Give them tools. Parenting is the same."

03 Developers thrive in flexible work environments

Sixty-one percent of developers who work remotely want to continue doing so — and 46% of those working in a traditional office today said they'd like to work remotely in the future. [Research from StackOverflow](#) shows that the highest job satisfaction ratings come from developers who work remotely full-time. There are multiple reasons behind this: greater flexibility in work schedules

enables more control over work-life balance, helps save on commute time, and limits overall distractions. [Engineering lead at OpenZepplin, Santiago Palladino](#), says one of his biggest challenges is managing a distributed team. He suggests that "the key thing is hiring remote people that have experience being remote."

*Based on Scrum methodology, a Sprint is defined as a set period of time during which specific code has to be reviewed and completed



Codility_

www.Codility.com/research