Do the Work vs Get the Work Done

Nandinii Yeleswarapu

September 7, 2025

Introduction

In software development, we talk a lot about Agile. Teams say they are "Agile," they hold daily stand-ups, they use Jira boards and sprints. Too often, though, Agile is practiced as simply *getting the work done*. Closing tickets and hitting deadlines rather than *doing the work* as it was meant to be: learning, adapting, and improving together.

That gap isn't a failure, it's an opportunity. If we shift how we think about success Agile becomes less of a buzzword and more of a mindset.

Agile's True Intent

The Agile Manifesto emphasizes adaptability, collaboration, and delivering value continuously. It is not about rushing to the finish line.

- **Get the work done:** check off tasks, deliver a feature as quickly as possible, optimize for throughput.
- **Do the work:** write tests, refactor when needed, collaborate with teammates, and improve continuously.

One mindset is about chasing "done." The other is about finding value in the journey and that's where growth happens.

Why Developers Struggle

Many developers struggle with Agile because it runs against how we were trained to think. From early schooling through college, the system rewards *completion*: finish the homework, pass the exam, get the grade, move on.

For example, in most courses if you fail one exam badly, it's extremely difficult to recover enough to pass the class. But what if instead, you failed a little less on each exam thus showing steady improvement, iteration after iteration? Shouldn't that be more meaningful? In education, the answer is usually "no." Samuel Beckett once wrote: "Ever tried? Ever failed? Try again. Fail again. Fail better." In Agile, this is exactly the mindset we want, failing small, learning, and improving over time.

This disconnect helps explain why developers often find Agile mindsets challenging. But it also shows us a path forward: if education rewarded iteration and growth, we'd graduate students who already think like Agile developers.

Interviews and the Same Opportunity

This same tension shows up in the hiring process for most software development jobs. Technical interviews often measure how quickly and correctly someone can solve a random coding challenge. Many people adapt by spending hours grinding LeetCode problems, memorizing patterns, and practicing until they can recognize and regurgitate solutions on demand.

But is that really what makes a great developer? In practice, much of development is not about having the answer right away, but about being willing to *do the work*: researching, experimenting, iterating, and learning until you find a solution. Imagine if interviews measured not only problem-solving speed, but also resilience, curiosity, and adaptability. That would reward the very qualities Agile depends on.

Towards Agile-Ready Education

If we truly want developers to thrive in Agile environments, maybe the problem starts much earlier. We need to cultivate an Agile mindset in schools and universities.

I actually experienced a glimpse of this in one of my undergraduate courses. Instead of grading only on fixed cutoffs, the professor built improvement into the grading system. The course had two midterms and a final exam, but the grading criteria was flexible:

- If you did well on two out of three exams, those two would count.
- If you struggled on the midterms but showed steady improvement and did really well on the final, then only the final exam score would be used.

This approach rewarded progress and resilience rather than perfection at every checkpoint. It encouraged students not to give up after a bad exam, but to keep learning and improving — exactly the kind of mindset Agile thrives on.

Imagine if more courses were structured this way. Students would graduate already comfortable with iteration, adaptation, and learning from small failures. This kind of education would produce graduates who already think in terms of "doing the work," not just "getting it done."

To sum it up

The software industry doesn't just need faster finishers, it needs steady builders, reflective learners, and adaptable teammates. Agile at its best is not about racing to "done," but about finding value in the journey.

If we can align how we teach, how we hire, and how we build software with that philosophy, we'll empower the next generation of developers to thrive, not by just getting the work done, but by truly doing the work.