The Pragmatic Programmers

# The Stress Equation

## Reduce Burnout, Increase Happiness and Productivity

Marcus Lagré
*edited by Julia Watson*

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit https://www.pragprog.com.

## Understand Your Complexity Debt

Technical debt is a concept that gets thrown around a lot, but I am not sure everyone who uses it actually understands what it means—or what the costs actually are. If that understanding was common knowledge, there wouldn't be so much debt around. Also, the concept only catches the technical aspects of what slows us down.

This is why I've started to refer to it as *Complexity Credits*.

Complexity credits are small debts that we accumulate over time. None of them are especially eye-catching on their own, but when zoomed out we see a pattern of behavior where our complexity debt is increasing.

Acting on credit is a perfectly viable business decision. Sometimes we can't wait as we need to strike while the iron is hot. If we don't have the financial means at this very moment, we call the bank to get credit. However, we can't continuously act on credit. If we don't have a viable repayment plan for those credits, the bank will eventually say no.

The problem with technical or organizational credit, as opposed to monetary credit, is that there is no bank. We are free to take as many credits as we like. This will slowly build up cognitive load and system inertia—the interest rate of those credits—that stops us from being efficient. Eventually this will reduce the speed of the organization, and harm our ability to act under pressure.

### Be Aware of the Type of Debt

While technical debt is probably the most obvious form of complexity credit in software, there are other things that cause us to accumulate complexity—both regarding our product or software and our organizational structure.

As our flow of work grows slower, there is a tendency to throw more people at the problem. Coordinators, project managers, more developers. This further deteriorates the system, because more people always means more complexity. More brains to keep informed, more heads to keep aligned, more chefs that stir the soup—more meetings. To remedy the confusion that usually follows, we "increase clarity" by assigning ownership—essentially defining who gets yelled at when expectations are not met.

With this "clear ownership", people grow protective of their own interests and their own responsibilities. So to get things done we need more negotiations—more meetings.

If this continues for years and decades, the organization will grow so complex that no one intuitively understands it. Official channels become so slow that it's impossible to get anything done. When this happens there is a tendency for a "black market" of back channels to evolve.[5]

If this is not continuously addressed, knowing these back channels then becomes the only realistic way of getting things done within a reasonable timespan.

## Have a Mindset for Reducing Complexity

Alfred North Whitehead was a philosopher and mathematician, perhaps best remembered for writing *Principia Mathematica* together with Bertrand Russel. In his book *An Introduction to Mathematics [Nor11]* he argues that civilizational advancement is a matter of reducing complexity to preserve cognitive energy,[6] expressing it as:

> *Civilization advances by extending the number of important operations which we can perform without thinking about them. Operations of thought are like cavalry charges in a battle—they are strictly limited in number, they require fresh horses, and must only be made at decisive moments.*

In other words, we don't have endless cognitive energy. If we waste that energy on things that could be made simpler, we reduce our ability to excel on harder problems. How many companies have this type of policy when deciding on tooling, product design, tech stack, or organizational structure?

My take on this quote is:

> *An organization advances by increasing the amount of finished work, without increasing the workload.*

With that in mind, what has your organization done in the past month to make it easier for yourselves to deliver value to your customers? If the answer is *nothing*, you really should consider how you can start working with continuous improvement, otherwise you may soon find yourself in a very costly complexity spiral.

## Listen to the Alarm Bells

One good indicator of organizational cholesterol is how long it takes for new employees to start delivering valuable work. How long does it take for them

5. https://www.heinz.cmu.edu/faculty-research/profiles/krackhardt-davidm/_files/krackhardt-hbr-spring2011.pdf
6. https://quod.lib.umich.edu/u/umhistmath/AAW5995.0001.001

to understand the software architecture? How long does it take for them to understand the organizational context they will be working in?

I have worked with companies that will say it takes at least ten to twelve months for a new programmer to start delivering value. They almost say it with pride, because they think they are so "advanced" that it takes an experienced engineer almost a year to understand the system and the product.

That shouldn't be a source of pride. That's an alarm bell!

Every company should be very aware of when they are wasting their employees' cognitive energy. That energy should be preserved for the big puzzles and the real problems. Reducing cognitive load should be one of the main focus points for continuous improvement.