Top 7 Things That Kill Developer Productivity

Introduction

A few weeks back, I had an epiphany: my code time, and active code time, had a whopping 2 hours difference when working for 4 hours. To get back on track, I actively cut down my blockers and tried to reduce the gap so I can build more, and spend less time on unwanted stuff. The greater the gap between these phases, the less productive my coding becomes.

Software developers battle productivity hurdles more often than other professionals. These barriers to developer productivity often spiral into low developer confidence, less time to code and groom, and even higher instances of burnout. It also impacts their creativity, and zeal to build.

According to this week's average global coding time on CodeTime, around 45% of the total coding time is just passive coding. The difference is resulting in the teams and organizations wasting time and money.

The lack of productivity stems from inefficient workflows, and the way devs spend their time.

In this article, we are going to look into some of the obstacles that kill the productivity of a developer. By knowing these, we can aim to shed light on the factors that can hinder development processes, delay projects, and lead to frustration for both individual developers and the entire engineering team. After knowing the problem, you can take the necessary steps to work on it.

So, let's get started.

1. Meetings

Ineffective meetings are by far one of the most unnecessary factors that lead to less productivity of not only developers but all other stakeholders. Coding requires focus. On average it takes around 30 minutes to indulge in focus mode. But due to many meetings, this focus gets broken and again the developer has to establish focus with the code.

There is also time consumption, as sometimes 10-minute meetings stretch to an hour. Thus, it reduces the time available for actual coding and problem-solving. Also, there is unnecessary attendance that is required for some meetings. If a meeting doesn't directly involve a developer's expertise, their presence might not be required.

2. Technical Debt (Fix it later)

Technical debt can be simply defined as the “Fix it later” mindset. It refers to the amount of work that has accumulated by taking shortcuts or making suboptimal code implementation during software development.

Initially, we try to make the function work and leave optimization for later. This might work in the short-term as it speeds up the project and you might meet your deadlines on time. But doing this again and again will leave a considerable amount of work to be done. Thus making it harder to maintain, extend, and improve the software

Technical debts can hamper the productivity of developers in many ways. Some of them can be:

- **The bottleneck in code review:** When technical debts increase, it leads to an increase in the time spent on the code review.
- **More bugs:** As the initial motive was speed and not optimization, this will lead to the introduction of hidden and unnoticed bugs.
- **Reduced code quality:** Working to just make it work, will lead to bad code quality, haphazard code reviews, even no code comments, complex code, and more.

All the points mentioned above will need additional time to work on it. Thus stretching the project timeline.

3. Code Reviews

Code reviews take time, and if the review process is too slow or overly bureaucratic, it can delay the integration of new code and slow down the overall development process. Sometimes developers raise PR but code reviewers don’t have time to review it. Thus increasing the time for developers to work on the next task. Even when the developer starts with the next tasks, there will be context switching while doing the code reviews. Thus hampering the concentration and productivity of the developer.

For code reviews, developers might have to attend multiple meetings causing a decrease in the productivity of developers. Often unclear or complex feedback on the code can take longer to solve the issues as it takes further conversation to understand the feedback. Code reviews are essential and a crucial part of an organization but done in the wrong way can only lead to a lack of productivity.

4. Micromanagement (Lack of Autonomy)

Micromanagement is a management style in which a supervisor closely observes and manages the work of their subordinates. In the context of developers, this can occur when a manager tries to control the way a developer codes. This can result in developers having less control over their code, processes, decisions, and creativity.

It can lead to a lack of productivity among the developers in various ways. A few of them can be:

- **Lack of Motivation:** Micromanagement can show that the organization has less trust in the developer's ability. That way, developers can easily feel demotivated.
- **Less Creativity:** Developing software is a creative task where you need to have your focus to explore creative solutions. But micromanagement will lead to having less control over the code, even hindering developer creativity.
- **Slow Decision-making:** Developers have to get confirmation from managers on simple decisions, wasting a lot of time in the process.

In all these cases, the productivity of the developer will go down.

5. Burnout

Burnout is one of the significant concerns of developers. Working on complex and challenging projects with tight deadlines, and the constant pressure to deliver high-quality code can lead to burnout. It can eventually lead to a fall in the developer's productivity as it will reduce the developer's focus and ability to concentrate on coding and building.

It will also lead to a decrease in the creativity and problem-solving skills of developers. It will eventually lead to a slower development cycle.

6. Poor Documentation

Documentation is essential for developers as it conveys critical information about code, projects, and processes. Poor documentation can cause significant delays in the development cycle as the developer will have to spend more time trying to understand the codebase, project, and process. Thus leading to reduced productivity for the developer.

While onboarding developers, providing poor documentation will lead to developers spending more time on tasks such as setting up the project, managing the environment, understanding the code, and other related stuff. In the absence of clear documentation, it becomes difficult to maintain and modify existing code. Developers may hesitate to refactor or make changes due to fear of breaking functionality. Thus, the productivity of the developers will be wasted on non-core tasks.

7. Unrealistic Deadlines

Deadlines are one of the things that demotivate the developers. When you have to work extensively during the smaller windows, you will get frustrated easily. It can cause burnout, poor code quality, negligence in code reviews, etc. This will lead to the accumulation of technical debts. Thus causing a fall in the productivity of the developer.

Deadlines are necessary to plan the development cycle, but putting pressure on developers by setting unrealistic deadlines will cause them stress. In stress, there will be a decrease in overall productivity and code quality.

Conclusion

A developer's productivity can be hindered by the above-mentioned factors which include unnecessary meetings, accumulated technical debts, stretched code reviews, more focus on micromanagement, stress causing burnout, poor documentation of code, and setting unrealistic deadlines for projects. I have tried to shed light on the challenges that a software developer faces while in pursuit of effectiveness and creative solutions.

The key takeaway here is that these challenges can be overcome to establish a balance between the developer's health and high productivity. You can use developer tools that can help you manage your productivity, deep focus, and work effectiveness.

Here are some tools that can help you in boosting productivity:

- FocusGuard: It is a Chrome extension that can help you in building focus by blocking sites.
- Code Time: It is a VS Code extension to track your coding time and active coding time.
- JavaScript Booster: This VS Code extension can provide a suggestion for code refactoring. You can find this kind of extension for other programming languages too.
- Hatica: While the above tools are confined to one task and are focused on coding, Hatica takes care of most of the work. It helps engineering teams boost developer productivity by improving the workflow, identifying bottlenecks, and tracking progress. By doing this, it can free a significant amount of developer's time. Learn more about this engineering management platform here.

I hope this article has helped you understand the cause of the deterioration of the developer's productivity. Thanks for reading the article.