Top 7 Things That Kill Developer Productivity

Introduction

I try to reduce the gap between the code time and active code time. So that I spend more time on building the application rather than on less useful stuff. The more gap between these denotes less productive coding. Software developers do struggle to find themselves battling productivity hurdles more often. These can cause hinder the creativity and productivity of the developer.

According to this week's average global coding time on CodeTime, Around 45% percentage of the total coding time is not utilized as active coding time. The difference is resulting in the teams and organizations wasting time and money. The reason behind the lack of productivity can be from developers as well as the way the workflow around it.

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 entire teams. After knowing the problem, you can take the necessary steps to work on it.

So, let's get started.

1. Meetings

By far one of the most unnecessary factors that lead to less productivity of not only developers but also other parties involved in this. Coding requires a focus that can be achieved while coding constantly. 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 minutes meetings stretch to an hour. Thus 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. As it refers to the amount of work that has accumulated by taking shortcuts or making suboptimal code implementation during software development. Initially, we tried to make the function work and leave optimization for later. This might work 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 works, will lead to bad code quality. It can be in terms of bad or no commenting on code, complex functions, dirty code, and others.

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

3. Code Reviews

As mentioned in the technical debts, more time spent on code reviews eventually leads to less productivity for developers. 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 with the developer's start on 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 can define as a management style where the supervisor closely observes and manage the work of the subordinate. In developer's ways, when the manager tries to control the way a developer should code. This lead to developers having less control over their code, process, decision, and creativity.

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

- **Lack of Motivation:** Micromanagement can show that the organization has less trust in the developer’s ability. Due to this, developers can easily feel demotivated.
- **Less Creativity:** Developing software is a creative task where you need to have your mind to explore creative solutions. But micromanagement will lead to having less control over the code and cause less creativity in the developer.
- **Slowed Decision-making:** Developers has to get confirmation from managers on simple decision causing a loss of time while doing this.

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 productivity of the developer as it will reduce the focus and the ability of the developer.

It will also lead to a decrease in the creativity of the developers. Problem-solving skills will also get affected due to burnout. This will lead to an increase in the number of errors made by them due to poor code quality. It will eventually be leading to a slower development cycle.

Burnout should be addressed by the organization for the well-being of the developers. It can lead to a productive and engaged development team.

6. Poor Documentation

Documentation is essential for developers as they convey critical information about code, projects, and processes. Poor documentation can cause significant delays in the development cycle. As the developer spends more time trying to understand the

codebase, project, and process. Thus leading to reduced productivity of 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. Unrealist 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 putting unrealistic deadlines will cause stress to the developers to meet the deadlines. In stress, there will be a decrease in overall productivity and code quality.

Conclusion

A developer's productivity can be hindered by many factors mentioned above. We have tried to shed light on the challenges that a software developer faces while in pursuit of effectiveness and creative solution. From ineffective meetings to setting unrealistic deadlines, we have covered many aspects that challenge the productivity of a developer.

The key takeaway you should take is that these challenges can be overcome to establish a balance between the developer's health and high productivity. You can use tools that can help you in managing the engineering teams for effective results. Hatica is such a tool that tries to help the engineering teams to increase their productivity by improving the workflow, identifying bottlenecks, and tracking progress. There are various ways you can integrate into the workflow to increase productivity here are a few of them:

- Reduced context switching of developers as they can focus on their tasks from a centralized view.
- Automate repetitive tasks so that developers don’t stuck with non-core tasks.
- Improved visibility of the task and their progress through the dashboard. Helping developers to prioritize their work more effectively.

There are other ways too that Hatica can help in enhancing the developer’s productivity. You can look more at the tool here.

I hope this article has helped you in understanding the cause of the deterioration of the developer’s productivity. Thanks for reading the article.