Developer Experience

ASW Seminar work 22-23

Álvaro González Erigoyen, UO282790
Andrés Cadenas Blanco, UO282276
Iván Vega García, UO276670

Interviewee short summary:

Abi Noda (https://www.linkedin.com/in/abinoda/) is the founder of Pull Panda (a tool for improved pull requesting, bought by GitHub in 2019) and one of the most important personalities in the field of DX studies, founder of GetDX.

Topics:

1. What is Developer Experience

Although now a days it is being used quite carelessly, DX refers to two main concepts:

- The classic DX: solutions and tools that developers might use to improve their workflow.
- The new DX: the actual experiences that developers go through when doing their job.

The importance of DX is always paramount, because DX is directly tied with productivity and satisfaction of employees. Even more so in today's remote and hybrid models, where traditional workplace experiences and techniques do not apply, or have a lesser impact.

2. Benefits of a good DX

We must take into account that DX is used to predict the productivity and in general to improve the performance of employees. It is also attractive for companies to have good Dx as this will attract potential talented employees.

Leadership influence in the DX and seniors might face problems which are divided in two local and global problems. Local problems are such as complexity of the code or the ones happening at the level of their team. Global problems are focused on problems shared by other teams. For this purpose, it exists the DX specialized team focused on finding and erasing the issues.

3. Advertising DX as a company

Companies nowadays are beginning to use as an attractiveness the DX. There is no formal way of tracking this so there are some mechanism candidates can do to check how well developed the DX is in a business this are such as using the job interview as a channel of communication between the company and the candidate. Examples of questions that may be asked are Which is the reviewing process followed or how the code is deployed locally… It is also important to check the reputation among former and actual employees.

4. Measuring DX. Alternative metrics

Now, we know that developer experience is important, so, how do we quantify it?

We can talk about two main ways to measure DX, system metric and self-reported measures. For example, on the system metrics approach you could look on GitHub the amount of time needed to produce and build a solution to check if it fits your models and to compare.

On the other hand, on self-reported measures, you could ask the developers about how they felt about the feedback loop of the project. Were there some parts where due to some code review your production had to be delayed? Did you have to control different proyects at once so the time for the project was reasonable?

The thing is, system metrics, even if they are used as of today, are not representative of the real results. We are missing the root, the business context. Developer experience is about the end-to-end experience of building and developing and releasing software and working across the team or multiple teams to accomplish that goal. You are not going to get any information about that on GitHub.

And, since it seems like developer experience is tied to productivity, what is productivity? How can we measure it?

We don't know. What productivity is and how it is measured are questions that people have been trying to answer for the past three decades.

Historically, companies have been using things like lines of code written, pull request made, or speed of production. Companies need to keep track of the number of products they can create during a period of time because they need these kinds of data to evaluate how things are going. But most of the time developer experience increases the overall productivity of the company and these kinds of measures are not real effective.

5. How to improve DX

Some companies are now investing in DX teams, teams whose purpose is to review, measure and improve DX across the company. They do this by treating the developers as if they were customers: they act as a stand-alone company that asks periodically for feedback to their users.

DX is not just business of the company, of course. As a developer, it is your responsibility to improve DX, both for your own benefit and for the company's. One of these techniques is job crafting, a technique that involves asking for small modifications of your assigned workload to better fit your individual personality, skillset, and rough points.

6. Code review challenges

For a software company, the importance of code reviews is obviously high. It is one of the main tools used to ensure quality standards in products, as it diminishes the possibility of passing errors into production. Nevertheless, it carries some difficulties and can really harm DX, both from the reviewer and from the reviewee.

Starting with the person whose code is being reviewed, they can be frustrated by the time it takes to get their code reviewed, by the harshness of the review or lack thereof.

Also, the reviewers face some problems: senior engineers, which usually do most of the code reviews, they face badly documented code, unclear changes, and completely wrong implementations. These are issues that must be properly communicated to the developer, but that are, often, difficult to convey.

Many of these issues can be easily solved with good documentation, shrinking review size in favor of amount, and logging changes and review results, which ultimately improve DX by a fair amount.

7. Barriers to measure DX

There are several barriers to improving developer experience, but it really begins with a lack of visibility and awareness.

Since companies don't try to measure DX, real problems that affect developers such as how many barriers do they have to do their daily work, are not known, if they are not known, they are not prioritized, and if they are not prioritized, they will never be solved. Most of the time this will lead to people leaving the company or not working to their full potential.

The second problem is that, even if we give visibility to these problems, some developer or some team may not speak their minds in fear of undesired consequences but then you may have a developer experience team that communicates all the things and tries to measure it among other teams.

In terms of visibility on remote, in a co-located environment you have the coffee machine, and people can have a coffee and have informal conversations about how their day is going. There you will find a lot of developer experience related information, like whether there is something that irritates or worries them. This kind of conversation, is almost nonexistent on remote conversations where most of the time the things that people talk about is about spring objectives.

In terms of some advantages of remote work, elements such as work life balance, uninterrupted time, for example, in some cases have improved due to remote working. However, you do see some evidence of the contrary where people are having more difficulty with the work-life balance because there's not this boundary between the office and where they work. And overall, since DX depends on the reality of each individual, working from home most of the time means more variables into the equation of figuring out how to make thing better.

These barriers need to be understood because as we can see, developer experience is something really important that can directly affect the productivity of a company, and you will need to confront these barriers if you intend to make this improvement

Links:

Productivity from the point of view of Oracle

Developer experience from the miscrosoft playbook

An Oracle tool to help devOps (you can see the importance of deployment)