Why Every Software Team Should Have a Developer-Experience Owner

The engineer caring for the developer experience in your team might be the unsung hero who deserves more praise

Daniel Thiry Follow
Jun 10 · 8 min read ★

Photo by Esteban Lopez on Unsplash

In recent years, more and more companies have realized

sentences such as “every company is a software company” and “software is eating the world” are more than theoretical statements. Rather, they express a new reality in which software departments aren’t only cost centers providing support functions for “actual business.” Software is now very often a key component of success.

Since development capacity is a scarce resource and good IT talent is hard to find, developer productivity has become very important in almost any scenario. It’s no longer enough to have software engineers doing their job of producing software — they also need to do it efficiently. To achieve this, the developer experience (in terms of workflows, tools, communication, etc.) must be optimized, which isn’t a simple task.

. . .

What’s Developer Experience (DX)?

Developer experience (DX) is the experience a developer has while performing a specific task or doing a job.

While this definition might sound simplistic, its real-world meaning is multifaceted and complex.

DX for tools

To perform their work, developers always need specific tools, such as IDEs, debuggers, monitoring tools, versioning software, and many others.

With increasing competition for pretty much all of these tools, they're already often optimized to meet the needs of their users, and the providers of these tools are usually aware of the main factors that drive a good DX for tools.

Albert Cavalcante wrote a post about DX for dev tools, also showing the relation of DX and user experience (UX). He lists the following main factors for a good DX of a dev tool:

- **Function:** The tool does what it's supposed to do
- **Stability:** The tool is reliable and works when needed. If bugs occur, they are fixed as fast as possible.
- **Ease of use:** The tool can be used intuitively — you can find all necessary information, and there are smart defaults. (This includes more than just the user interface but things like documentation, templates, shortcuts, community support, etc.)
- **Clarity:** The tool displays all necessary information in an understandable way and provides full visibility of the consequences of any action

Beyond what's mentioned in the article, there are certainly other things to consider when you provide a tool with a great DX, such as:

- **Compatibility:** The tool works with a wide range of technologies, thus not limiting the user's decision options
- **Integrability:** The tool works with other tools in a

meaningful way and provides necessary interfaces for this

While all these factors just consider the perspective of a user of a single tool, software development takes place in a more complex setting with a variety of tools — and usually in a team. For this, it makes sense to also take a look at DX in software teams.

Our experience

We at DevSpace, as a provider of dev tools, have realized there's actually even more than one experience with our solution. One the one hand, there's usually someone setting up DevSpace/Loft— the admin — who needs in-depth access and overview as well as a technical base close to Kubernetes.

On the other hand, there are the developers who just use these tools and should get a simple access to a Kubernetes environment. For them, simplicity and ease of use is more important than technical details. Therefore, even as a tool provider, it makes sense to have the DX of a development team in mind and not to just focus on the use case of a single person.

DX in teams

DX in a team is the experience a developer has during the actual work of developing software in a team setting and with several tools.

To have a really good DX in a team, again, many factors play a role:

- **Communication:** The communication should be clear and standardized. This includes the tools used for communication (e.g., personal, chat, email, phone, etc.) as well as the audience (i.e., communicate in the right way with the right person).
- **Workflow standardization:** Everybody on the team should use the same workflow for the same task to avoid additional problems and conflicts. For example, this could include a standardized process of testing and debugging before deploying.
- **Choice of toolset:** Everybody on the team should use the same tools for the same task, and they should be configured in the same way. (Identical configuration mainly includes shared tools such as CI/CD tools or staging environments. Of course, it doesn't mean everybody needs to have exactly the same settings/user preferences, such as a dark mode in an IDE.)
- **Documentation:** Not only existing software but also workflows and processes should be properly documented and the documentation should be easy to understand and use
- **Onboarding:** If new developers join a team, it should be clear what they have to do. They need to know which tools they should install and use, which tasks to take on, and who to turn to with their questions.

This list could probably be extended by many other factors. However, it should already be clear that ensuring all of this and keeping it up to date isn't always easy. This is why I want to

propose a new role in developer teams: the *developer-experience owner, DXO*.

. . .

The Developer-Experience Owner (DXO)

What's a developer-experience owner (DXO)?

A DXO is a member of an engineering team who's designated to care for the DX of the team by selecting and configuring the tools, defining workflows, onboarding new team members, and being a central person of contact for all questions regarding DX.

Typical tasks of a DXO

As multifaceted as the DX itself are the tasks a DXO has to perform. Typically, the DXO should be responsible for the **selection of the tools** used in a team. The DXO should constantly evaluate if the currently used tools are optimal for the tasks at hand and determine if they should be replaced or differently configured.

Especially in CI/CD, the tool landscape is evolving very fast, and the average developer doesn't want to manage the associated processes. Here, developers often need access to a staging environment, which could also be managed by the DXO, either by **managing the staging environment** itself or by **managing the access of the developers to it**.

Another aspect of the role of the DXO is **keeping the**

developers in sync — i.e., ensuring everybody knows what's going on and when and which updates are needed. This may also include **preventing the emergence of a bad form of shadow IT**, where developers use tools or processes without telling anybody because they fear it may not be allowed. In such cases, the DXO should be involved in **testing the tools and workflows** and should subsequently have the power to formally allow and promote their use.

Related to this, the DXO is responsible for the **documentation of all internal workflows**, which requires a constant process of updating and revising to keep the documentation current and relevant. This is especially important when **onboarding new developers** to the team. In this situation, new team members regularly face a lot of issues, and the DXO shouldn't only provide them all the necessary information but also be the **go-to person for any remaining questions**.

In general, the DXO serves as a **central point of contact for all issues, ideas, and complaints concerning the developer experience** in the team but also plays a key role for nontechnical managers in communicating with the dev team.

Benefits of a DXO

As stated in the beginning, developer productivity is nowadays a key competitive advantage. Besides the obvious productivity advantages, it can also lead to higher-quality outcomes if everybody knows exactly what they're doing, why they're doing it, and can focus on their specific tasks.

It can also improve developer happiness, resulting in lower employee turnover, and more motivation by giving the developers the opportunity to focus on their actual work instead of being distracted by communication problems, malfunctioning tools, and finding missing information.

Additionally, every new team member will be positively surprised if they can meaningfully contribute from Day 1 instead of having to set up their environment on their own without a trusted person to turn to for questions.

Of course, all of this can be achieved without a dedicated DXO. However, similar to the concept of *code ownership*, ownership of the DX will enforce responsibility for it. It'll also be easier to keep the team synchronized because there will be fewer people involved in decision making.

The Role of a DXO

So what makes the ideal DXO candidate? At first, as the role is so tightly connected to internal processes and communication, a DXO should almost always be someone who has been working in the team for a while. They need to know the company's goals and the existing software architecture — as well as the current toolset and the reasons why it was chosen as it is.

This usually requires a more senior position for DXOs, but I believe this isn't a necessity if a suitable candidate can be found otherwise.

Generally, a DXO needs to have an exceptional understanding

of software engineering and the related processes. If the DXO is supposed to manage a team with both front- and back-end engineers, it should be a person who understands both worlds. Since the role often also involves CI/CD, an understanding of DevOps, or even operations technologies, would certainly be helpful, too.

Another very important trait of the ideal DXO is the ability to communicate very efficiently but also sympathetically. They need to be a trusted person the team members can turn to, even with "stupid" questions. This also requires a very good reputation with their colleagues as DXOs need to have the authority to make decisions.

Finally, it's very important the candidate is willing to take on this role. as it'll be at least a major distraction from actual software development.

Our experience

The role of a DXO already exists in many teams but is not called DXO. It's usually a rather implicit role that someone in the team has taken on.

From our experience at DevSpace providing developer tooling, the implicit DXOs are often the ones trying out our solutions first. They evaluate them and often also configure them for their team's use case if they have decided to introduce them to their companies. Typically, we see these implicit DXOs have official roles like VP of engineering, senior engineer, or DevOps engineer.

. . .

Recommendations and Conclusion

Some companies already hire DX engineers, but this is mostly an externally facing role for an engineer who's responsible for SDKs used by outside developers or who works as a developer advocate for open-source software. I believe an internally facing role, caring for the experience of your own developers, is at least as important as an externally facing one.

While there might already be someone in your team working as implicit DXO, it could be beneficial to turn this into an explicit, formal role. Still, that doesn't mean it needs to become a full-time position. It should rather come with additional responsibility, decision authority, and a time budget allocated to test new tools and care for the dev experience in your team. It can also make sense to choose a small group to share the tasks in this area.

So maybe think about your team, or ask around and see if there already is an implicit DXO. If not, you should consider promoting someone to DXO. And if there is one, you should really think about appreciating what they do to make their colleagues happy and the engineering processes better by giving them an additional title, more time, and potentially a raise for the important work they do.

Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. Watch

Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. Explore

Explore your membership

Thank you for being a member of Medium. You get unlimited access to insightful stories from amazing thinkers and storytellers. Browse