One of the main things about humans is that we make mistakes. Nobody is flawless. People who seem to be, are most often called ‘the machines’ as they resemble the automated processes within them. But are the processes really flawless? Of course not! After all, technological advances were once created by humans. Mistakes are unavoidable. Therefore, the key to success is not so much eliminating mistakes but avoiding them and being able to fix them. In the 21st century, the time factor is also crucial. But what if we do not even know that the mistake was made? Especially in the case of automated processes. You know well what I am talking about. Admit how many times you did not know for hours or days that something failed. And the amount of time spent searching for what it was that failed? Do not even get me started…

But what if I told you that there was a way to:

- Monitor all the processes constantly
- Catch the broken code on the spot
- Fix the problems instantly, right after (or before!) they occur, and…

… do it all before the end user even realizes it.

A big round of applause to our hero – Sentry

To put it plainly, Sentry is a monitoring platform that focuses on helping the developer keep an eye on the processes within the apps and catch errors. Let’s say there is an error in the code. Using Sentry will help identify where the problem is and why the code is broken. Every company that uses Sentry has full visibility into the code. They can quickly catch the error before it becomes an issue and starts affecting the performance and uptime (the time when the application or service is working). Time also has a different meaning here. Despite the error, the app can still be in an operational state but work slower. Sentry also monitors the response time and flags up when it is not up to the agreed standard. By providing all the answers, Sentry helps to learn about how apps work, what in that work is essential, and most importantly, how to save time on resolving issues that should not happen in the first place. In other words, if there is something wrong with the system, Sentry knows about it and helps to prevent the app from malfunctioning, the loss of customers and the loss of business. And business performance is naturally in every business owner’s interest.

Why it is important to catch errors (quickly!)

The obvious and straightforward answer to that question seems to be: to assure the app's or system's functionality. This is a general idea. But imagine the situation where the error occurs. The app slows down or stops working. But apps are made to simplify people's life. The end user gets used to them and subconsciously depends on them in due course. If they stop working and there is no plan B for the end user, there is bound to be dissatisfaction. And dissatisfaction of an end user will affect your business reputation, which might ultimately affect the business performance. From a practical point of view, what happens to the business when the app stops working? The end user starts to search for a new app. There are countless available replacements on the market, and it truly is so easy to lose clients and income when the app is flawed. God forbid the errors and bugs happen frequently. Even more, than once is enough for the system users to lose trust and interest in the product. And competition does not sleep. Or maybe they do? But only because Sentry or other monitoring systems deal with possible errors for them. Have you ever wondered why a created software or app suddenly encounters a problem and stops working?

Errors in the software – why do they occur?

Except for the apparent human error, we must consider several other factors that affect the apps. One is the complexity of the code or the technology in question. It is logical when you think of it – the more complex something is, the easier for it to become flawed. Another thing to blame is that the systems rarely work separately nowadays. Many of them are a part of one larger web. You can imagine how many interfaces must connect to make everything in the system work smoothly. Systems or apps also frequently undergo the implementation of new technologies that are supposed to help them be more efficient. Unfortunately, during the implementation, errors often occur. There might also be the case that not enough checks were done to establish whether the data is correct or sufficient, and it turns out not to be.

Whatever the reason for the occurring error is, the thing about the software that all of us need to remember is that not a person on this Earth can understand all complexities of software. A single line of code is bound to have thousands of dependencies or outputs it deals with. That is why, once again, errors are going to occur. One of the most important things for the business is not the fact that the mistake happened but how quickly it could be resolved, and the system could be

brought back to the desired functionality. The good news is that, with the complexities of today's systems, not every error is crucial for their basic functionalities, or at least some can wait a bit longer to get resolved.

Errors detection and fixing in Agile

Working in Agile means that software updates are very frequent (one sprint usually lasts between 4-12 weeks). Frequent updates mean frequent testing. The monitoring system should, of course, detect all the issues. But how to solve all errors timely where there might be thousands that occur in a new update? Like with everything in life, one must prioritize. There are different types of incidents affecting different areas of system functionality. Some are more important; some can wait their turn. The severity levels are as below:

1. Non-incident support tasks – occurrences that have no operational impact on the system. The software is still operational with no negative impact on productivity. It is, for example, a change of an element on the website, adding a more detailed description of a website element, etc.
2. Minor incident – some less important/non-essential features of the software become unavailable, or a good alternative solution is available. The customer impact is minimal. For example, the style of the text was changed but did not display properly. However, the text is visible.
3. Medium incident – important features of the software are affected, but there are some alternative solutions; some areas of the software are not impacted, but some functionality is unavailable. For example, the comment section on the website does not work, but the user can email support.
4. Major incident – the number of users who are unable to use the software or reasonably continue working using it; major loss of function or performance with no alternative solution available. For example, the end user cannot perform checkout in an online shop.
5. Critical incident – no customer can use the software. The full loss of function or performance occurs, and there is no other available solution. For example, the website has been hacked or is subject to an error in DNS change and is therefore not available worldwide.

Errors within different apps can be prioritized. For example, a critical error within one app may not be classified as such compared to critical problems within other apps. Instead, its severity might be classed as high, prioritizing the resolving of

other apps' issues. Another criterion considered while assessing error severity is the probability of its occurrence. For instance, if the error is critical, but the chances of its happening are very low until it happens, its classification will be significantly lower.

A typical Agile approach to dealing with errors is... avoiding them! A good monitoring platform can identify the problem, assess its severity, and time to fix it, but most of all can do it quickly. And it is not a matter of testing the update or system after the problem occurred and the user complaint is already logged. Sentry does all the work before the problem can even occur. This is where the magic happens. The developer accepts the problem, anticipates it, and has the chance to eliminate it before the end user finds out about its existence. This not only lowers the dissatisfaction level but saves lots of time and the loss of money because of a malfunction. Agile is a broad and interesting path of system development. To learn more, see our posts What do "Agile" mean in context of software development? and How to "Agile?" – full script

Sentry does double magic

The detection of problems or errors before they happen is impressive. Sentry not only alerts about the issue but also gives a clue about what change might have caused it, what functions and versions of the app are affected, and finally, who might be the best equipped to solve the problem. But Sentry does something more. It gives awareness. The developer becomes aware that some problems can even exist. Provided reporting with the highlighted issues gives a unique insight into what happened and helps avoid such problems in the future. So, we can safely say that except for avoiding the error occurring in the nearest future, the developer might get into the habit of never letting it flag up again. Now, that is a real deal! No malfunctions mean happy customers and no income loss from a long-term perspective.

A few words from the actual Sentry's user

Sentry operates multiple popular systems, and its compatibility is systematically updated. We use Sentry in our business's day-to-day processes and products for our clients. Since its implementation, there have been significantly fewer errors logged because we know about the possible issues before they even have the chance to happen. Join our clients whose projects are monitored by Sentry. We assure you

that the products we develop are as flawless as possible. All that thanks to Sentry.