Developers, Learn To Say No

Adrian Kosmaczewski

2016-05-12

All change starts with a “No.”

This morning I started my day with a dozen crashes all over the various devices that our technological life has sprinkled around us. First an app crashed on my iPad, then a web page stopped responding in Chrome, then Safari also had its hiccups, finally an iPhone app stopped responding. A couple of reboots followed.

And I have not opened Xcode yet.

Suffice to say, I had enough, and I hyperventilated my frustration on Twitter, begging for the care and support of my dear followers. Not a good start for my day.

> we’ve reached the point in mankind history where every. single. piece. of. software. is. broken.
>
> not even a webpage without bugs, crashes…
>
> — akosma (@akosma) May 12, 2016

“Software is eating the world,” they say. “Now every company is a software company,” they say. And I wish I could build a time machine and seek refuge somewhere in the middle of the Pleistocene era. Alas, we are not there yet, and anyway, given the current state of software, I would end up in the middle of the Trinity test instead.

The problem is not technology, however. It does not matter which programming language you use. It does not matter which operating system your code runs on. The problem lies in us, software developers all over the planet, because **we say “Yes” all too often.**

I have a cure for that.

We have to start saying “No,” louder and more often than ever.

A Call To Action

We should start right now.

See that manager approaching your cubicle? He will ask you in a few minutes for a “rough estimation” of a new feature.

Say “No.”

See that meeting room? Project managers and product owners are asking you whether a feature will soon be ready or not.

Say “No.”

There is a manager, right now, somewhere, trimming next year’s budget of the QA team, because whatever.

Say “No.”

The Reasons We Say “Yes”

We live in a world where, apparently, everything has to be positive. Ping pong tables. Free lunches. Money. Segways. Tesla. “Aim for the stars, you might end up on the Moon.” “Yes we can!” “Just do it!”

We are expected to have a “can-do” attitude, and to always bring positive ideas to the table, to contribute to a “positive team spirit,” that “we can,” and that, therefore, “we must.”

Now, please, read the following article carefully, and come back as soon as you have finished: “We Have Become Exhausted Slaves in a Culture of Positivity.” I think the title says it all.

The common wisdom states that young developers, freshly graduated and early in their careers are more prone to say “Yes,” but this is a misguided view, as I have been myself found guilty of saying “Yes” too quickly recently, and I have heard it very often from other senior developers, too. Saying “Yes” can happen, and it is sometimes unfortunate. It can have a terrible effect in project schedules, in project costs, and in the final quality of products—or rather, their lack thereof.

The Fear Of Saying “No”

The net result of this situation is that we fear saying no. We fear rejection. We fear being left out. I know about that: I have been rejected, I have been left out because of my tendency to be rather pessimistic in my estimations and my outlook.

Having experience, having “been there, done that,” being the “senior developer” of a team should and can include a safe amount of conservative views. What is not okay is stubbornness and failing to learn from evidence; being senior does not mean being always right. But being conservative is a totally acceptable characteristic. I would even go as far and say, it is a badly needed one.

Of course, saying no will set you aside. You are going to be left out from important meetings. You might even be fired because of your “attitude” or your “lack of integration with the team spirit” or whatever the buzzword of the day in Human Resources circles.

And you know what? You should leave. Go away. As I mentioned in my previous article, the software industry has an incredible demand for software developers, and an incredibly short supply thereof. So leave. You do not need to stay in a place where your views are systematically rejected “because positivity.” **Life is short, and you must feel confident to speak out your mind regarding project issues.**

The Virtuosity Of Saying “No”

There is, however, a happy ending, one that I have witnessed in the past, albeit not very often: when you say “No,” occasionally, somebody will look at you in the eyes, and with a puzzled look will ask you **“Why?”**

Aha! I love that moment. Hear the angels singing “Hallelujah” as they come down from the skies? You have finally struck the chord. Something important is about to happen. Breathe and smile. This is a great moment in your career.

At that moment, look at the person in the eyes, and I hope you will realize that the question was fair, frank and genuine. If that is the case, breathe deeply once again, and provide all the rationale you can about your point of view.

Explain that the feature will take longer than expected because the QA budget has been reduced. Or because the continuous integration server has to be upgraded. Or because you need more VMs in Amazon or Azure. Or because there are version compatibility problems that could appear. Or because you need more developers, *et cætera.*

You will know if the person in front of you wants to learn, or if it is just a hypocrite question. You will know. If the question does not fall in the latter case, it is your **obligation** to explain your point of view. Give all the information you have. Be generous. Draw a couple of diagrams. Teach. Be the senior developer that you are meant to be. This person then should explain to you the business background, the reasons behind the situation, the rationale for some decisions, and all those details that you might not be aware of.

See what is happening now? A conversation, a real dialogue between business and technology. And now you know why it happened: **because you said “No.”**

So, go on, say it. Loud.

Do not be afraid of saying “No.”

And then teach, learn, rinse and repeat.