Developers and Politics

Adrian Kosmaczewski

2016-08-16

Some thoughts around the tensions between technical and business teams. *TL;DR: I do not offer a solution in this text for political problems in software organisations; I just want this article to help developers not to feel alone in this struggle.*

Most software developers I know have a hard time understanding political power. We tend to assume that technical knowledge is power—which is not false; it is, however, a limited kind of power. I understand the word "Power," in this context, as the capacity to make things happen.

The basis of all evolution in social systems is, whether we like it or not, the constant shift and balance of power. History has been written on imbalances of power. Wars have been fought because of them. Millions have died and migrated because of them. And you, my dear reader, willingly or not, working remotely or not, in open source or in closed source projects, you are bound to obey dynamics that are more often than not driven by politics.

And your employer, your local user group, your preferred developer conference, all of these organisations and many others are also driven by politics, by people in charge telling other people what to do, usually but not always after getting their support in some way. Maybe you get a salary; maybe you get some stock options; maybe you just believe in your leader; all of these actions validate and give power to the ones that take the ultimate decisions, whatever they may be.

For software developers, the conflict appears when the decisions at stake are not entirely logic, rational, and seem driven by ego and political power quotas on the roulette table. We struggle in this kind of situations. We just do not know how to react. This is a natural part of the ongoing mismatch between *suits* and *t-shirts*, between business people and technical teams. This is part of our nature.

To be fair, this situation is not new, at all. The plots of movies such as "The Right Stuff" or "War Games" drew inspiration from the impedance mismatch between technical and business teams. But in your case, dear reader, your nodding while reading this text has nothing to do with an Oscar-winning performance. You would like to understand why your project managers are getting bonuses while you are not. You would like to know why some Scrum implementations become NSA surveillance programs. You would like to know how to be heard, for your ideas are good but nobody seems to pay attention to you.

You would like to know why it always seems to be better to change jobs than to change your current job.

Frustrating, huh? The truth is, I have no clue. I am a software developer myself, and I have long suffered this mismatch. I am not going to offer solutions or a checklist for people to follow and to try to surf on top of conflict situations.

In a previous article of mine provides a very simple tool that can be used in this respect. I actually mention, at the end of that text, that the aim of saying "No" is, ultimately, the establishment of a dialog. The importance of this fact should not be underestimated; it is not just saying "no" for the sake of it, but to generate a dialog. Should this dialog not happen, then yes, I suggest learning more about politics, and in the worst of cases, just leaving for other horizons.

I can totally understand and relate to the struggle generated by teams that cannot communicate within themselves, let alone with their surrounding communities. I do think however, that the lack of communication is not the fault of either party separately, but rather, of both.

In other words, if the dialog I mentioned above does not happen, if a communication channel does not work, the problem lies at both ends at the same time. We are also responsible for our broken teams, just like all other team members are.