The Good, the Practical, and the Expedient

By Steve Berczuk - March 12, 2019

Software development involves managing risks, including technical and market risks. Some technical choices may be "better" but would delay a project, leading to market risks; a system that is well built but delivered late may mean it doesn't get used.

But skipping good practice may lead to downstream problems that can slow you down later when you most need to move quickly. The definitions of "better" and "good practice" are fuzzy at best, and often arbitrary, so the "right" choice is not always clear.

When faced with something that's not working while under schedule pressure, you will often decide to make a choice that will help move things along. You may be tempted to frame this as a practical choice made through inspecting and adapting, but some choices are less about being practical than about getting things done quickly. However, that incurs risk. To manage this risk you need to be aware of the differences between *practical* and *expedient*.

A practical choice exemplifies "inspect and adapt." It considers the environment you are working in and is made with the idea of reducing friction while minimizing costs. For example, if you are using a web services production framework that has out-of-the-box support for certain tools and configuration libraries, you might choose to use those tools and libraries even though you are more familiar with other ones. Or you might choose to dispense with using one tool and migrate the project to another due to the skill set of people on your team and because the migration cost is small and one time. A practical or pragmatic choice might diverge from a standard, but it incurs minimal technical debt.

An expedient choice is one that also will lead to a faster delivery, but it does incur some technical debt. Expedient choices still can be the right choices because delivery means value and feedback, but if you don't realize and plan for any resulting technical debt, you will be in for an unpleasant surprise. An example of an expedient choice is to share a database between services in a microservices architecture. You are coupling the services in a way that, as the system grows, will lead to a need to coordinate deployments, which is antithetical to why one would choose a microservices architecture.

Making these types of choices is inevitable. The key is to be aware of downstream costs. And sometimes it's not clear what the costs of a decision will be, so being somewhat skeptical before assuming that a deviation from a standard practice is a practical choice may save you some problems later.

A decision can be pragmatic, expedient, or sometimes both at the same time. And either kind of decision makes sense when the "ideal" isn't viable. But you need to be careful not to confuse the two, lest you lull yourself into a false sense of security about how much risk and technical debt you are incurring. The perfect may be the enemy of the good, but you need to consider whether the choices you make are good ones in your context.