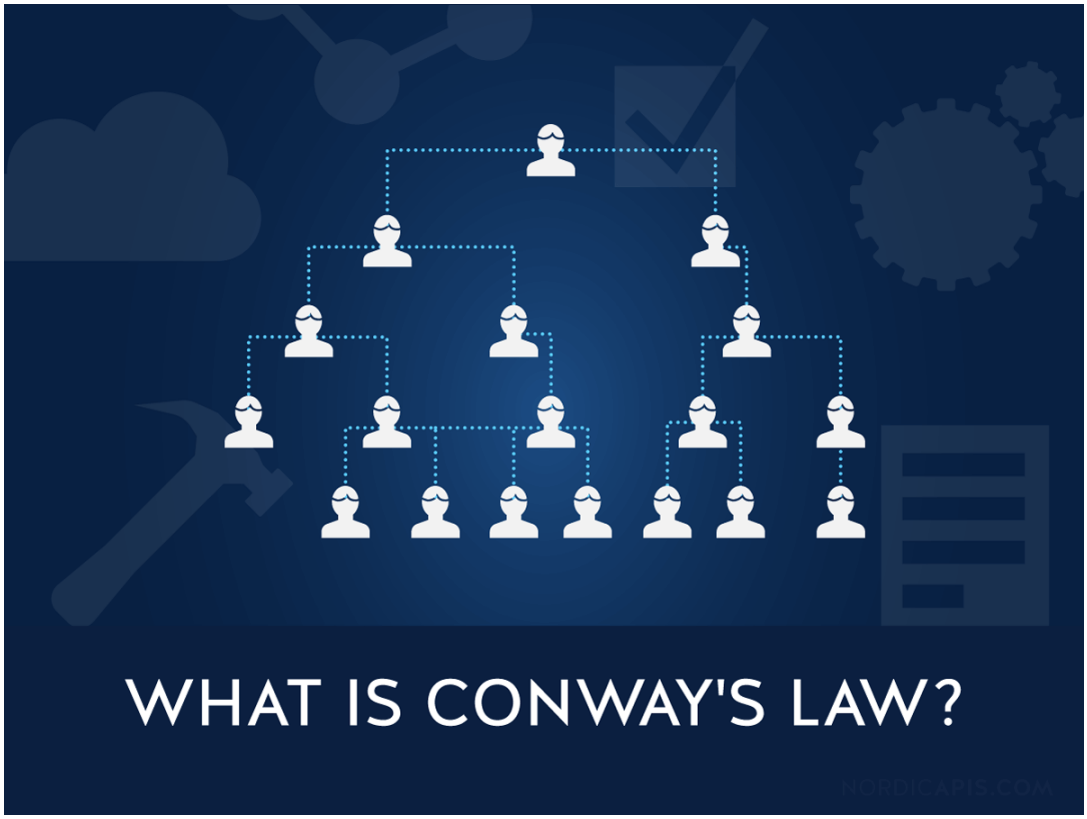
nordicapis.com

Conway's Law: What Does It Mean for Your API Strategy? | Nordic APIs |

Thomas Bush

7-9 minutes



Back in October, I attended the 2019 Nordic APIs Platform Summit in Stockholm, Sweden. Among a whole host of interesting topics, one idea that kept creeping up was that of Conway's Law, like in Zdenek Nemec's talk on choosing an API style. The straightforward theory says that we build systems according to our internal communication structures. But how does that apply to API design, is it a good thing (spoiler alert: probably not!), and what can we do about it? I'll answer those questions and more in this article — first

things first, let's get some definitions out of the way!

What Is Conway's Law?

Conway's Law is a phenomenon whereby organizations build systems that closely reflect their internal structures. It was originally formulated by programmer Melvin Conway more than 50 years ago, who put it this way:
"organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations."

The principle appears to hold true in a variety of fields — technical and otherwise — but is most commonly applied to the world of software development. And within the realm of software, there's one particular movement whose practitioners seem to have taken a real fancy to Conway's Law: APIs.

Conway's Law and APIs

The idea is simple: Conway's Law states that we tend to build systems that reflect our internal communication structures, so any APIs we build will also reflect our internal communication structures. Not only will this determine how many different APIs we end up with, but also the scope and nature of each individual API.

Let's look at an example. Suppose your organization deals in both tea and coffee, and you're looking to build some customer-facing inventory APIs as part of your digital transformation strategy. If the structure of your organization is heavily divided between "tea people" and "coffee people" — perhaps even with separate tech teams — you're likely to end up with two APIs (one for each product) coded very differently.

If your organization instead uses a single-branch structure, with the same product, marketing, and tech teams for each beverage,

you're a lot more likely to end up with just one API that does it all — and in a homogenous fashion.

Is Conway's Law a Bad Thing?

Many of us talk about Conway's Law as if it were a bad thing. That's because it often is, especially for the average enterprise. According to anthropologist Robin Dunbar, we as humans struggle to maintain more than 150 (this has been dubbed *Dunbar's Number* stable relationships at a time. With established enterprises boasting well over 150 employees, sometimes in tech staff alone, communication between teams is indeed fragmented. As a result, their APIs are loosely coupled and often inconsistent.

There's no need for me to explain why inconsistency is a bad thing — just look back at the earlier example. In the fragmented tea and coffee organization, having two inconsistent inventory APIs would force customers (most of whom purchase both products anyway) to write twice as much code. It's not like customers can just reuse the same code for each API since they behave differently.

Well, what if you can manage to keep API design consistent across your organization? Unfortunately, you still end up with a bunch of loosely coupled APIs. I know, I know — that sounds like a good thing. After all, we're actively trying to get away from gargantuan monolithic applications, instead of replacing them with sleek, *independent* microservices.

The problem is that APIs should be loosely coupled by choice, not by necessity. There is a time and a place for loose coupling. Especially on the backend, loose coupling can help to improve security and flexibility, and, importantly, speed up development cycles.

On the frontend, however, your APIs should be designed against consumers' use cases, which usually involves bundling together

separate parts of your infrastructure. Even microservices proponents like James Lewis and Martin Fowler accept that services should be organized around business capability, offering a "**broad-stack** implementation of software for that business area" (emphasis mine).

How to Combat Conway's Law

Occasionally, Conway's Law will work in your favor. But what are you to do if your organization's internal communication structures are less than ideal, or poorly reflect how customers interact with your business, as described above? It turns out there at least a few options.

The Inverse Conway Maneuver

One of the more radical approaches to tackling Conway's Law is the so-called **Inverse Conway Maneuver**. Instead of trying to fight Conway's Law, this technique actually makes use of it! Just start with the vision of what APIs you'll need to best serve your customers and — working from the outside in — structure your organization around those products.

In fact, this is exactly what tech companies like Netflix and Amazon do. By structuring themselves around small, autonomous teams (see Amazon's *Two-Pizza Teams*), they're able to more closely adopt a modular microservices architecture for any systems and products.

Of course, pushing for big organizational changes in an enterprise environment is easier said than done. However, if you can at least structure parts of your organization (especially the tech teams) around your ideal APIs, you'll certainly benefit from Conway's Law.

Improved Communication

If you refer back to Conway's original statement, you'll see that he believes systems end up reflecting organizations' *communication* structures, and not necessarily their organizational structures (although these two are usually one and the same). So, you may be able to circumvent the effects of Conway's Law simply by **improving communication** in your organization.

Let's refer back to the example once again. Even if you have separate tech teams for coffee and tea, you can still encourage (or force!) them to communicate on the creation of your APIs. If you can maintain this communication, you're likely to end up with two APIs that behave exactly the same, if not one API that does it all! And all that for not a lot of effort…

Standards

Suppose your organization is gigantic — and I mean *gigantic* — and it's simply not feasible to have all product teams communicating with one another on a consistent basis. Well, another way to manage Conway's Law is to implement organization-wide standards for APIs. This is something larger organizations are already doing, and the benefits are clear. Although this may not help with ensuring APIs are coupled in a way meaningful to the consumer, it will definitely eliminate inconsistencies in API behavior.

Final Thoughts

Conway's Law is a simple principle with big repercussions: the systems (or APIs) we design tend to mirror our internal communication structures. In practice, this means that large enterprises with loosely coupled, fragmented structures will end up building APIs of a similar nature. There are a few ways to get around Conway's Law, with the most radical being the Inverse Conway Maneuver. More approachable alternatives include

promoting cross-branch communication and implementing standards for API design.