Editor: **Diomidis Spinellis**
Google, dspin@google.com

Bespoke Infrastructures

Diomidis Spinellis

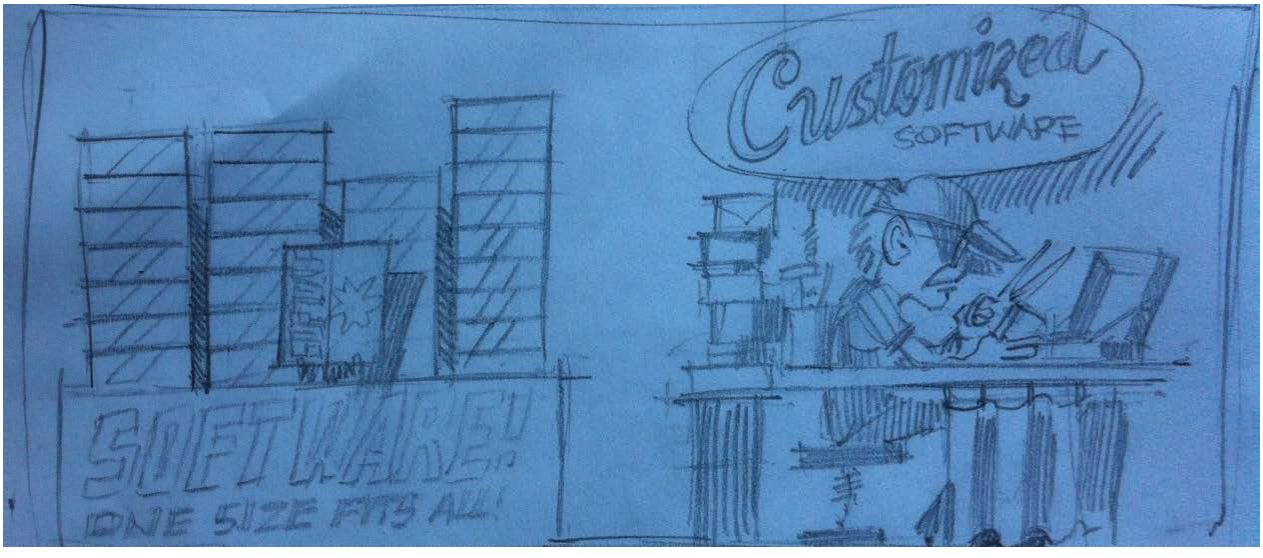
IN THE 1920S, the Ford Motor Company embarked on an ill-fated attempt to establish an industrial town in an Amazon rainforest as a way to secure a cultivated rubber supply for its cars' wheels. At the time, it already owned ore mines, forests, and a steel foundry to produce the raw materials for its cars; today, it buys from external suppliers, even its cars' electronic control units. How do these two phases of the automotive industry's history relate to the way we currently develop and adopt infrastructure in our profession?

Infrastructure developed within your organization for its own internal use can take many forms: operating systems, compilers, programming languages, version control systems, platforms for building, testing, and continuous integration, database management systems, application development frameworks, game engines, or utility libraries. Bespoke infrastructures can also extend to methods for doing work, such as the development process, code reviews, workflows, code style rules, and testing and integration practices.

The Case For

The obvious reason for creating a bespoke solution is that it can be tailored to fit your organization's unique needs. For example, you can optimize the design of a bespoke database management system or cache server to fit exactly your organization's load and query profile. Aggressively tailored solutions can run circles around offerings that try to please everyone, plus bespoke solutions can support features particular to your organization's unique needs: a programming language construct, a database column type, or a game engine interaction style.

Then there's the flexibility: as the owner of the infrastructure, you decide where it's going. If you want to add a new feature or fix a bug, you



0740-7459/14/$31.00 © 2014 IEEE

devote the required resources, and, presto, your wish is fulfilled. In contrast, if you adopt a commercial offering, you can only hope that the vendor moves in the direction you want; if you work with an open source solution, you have to coordinate with its developers (and sometimes jump through multiple hoops) to integrate your changes upstream.

Put simply, bespoke infrastructures allow your organization to innovate and keep the fruits of any findings to itself, which can provide it with tactical or even strategic advantages over the competition. As examples, consider the bespoke database and caching solutions that allow big social networking companies to drink data from a fire hose and the awesome proprietary datacenter infrastructures developed by the largest cloud service providers. Even if the benefits of a bespoke infrastructure are dubious, its mere existence can serve as a selling point or a differentiator in the market.

The Case Against

Proprietary infrastructure is only known within the organization that hosts it. Consequently, new employees face a significant hurdle before they can become productive and stop inundating their colleagues with questions. Contrast this with the case of a widely used offering that lets newcomers add value to the organization from day one by folding in their relevant knowledge, experience, and improved practices. The use of a bespoke infrastructure imposes its own vocabulary, hindering the informal communication of developers with colleagues in other organizations. Along the same lines, users of a bespoke solution won't be able to reach out to the global online community for answers and support, a convenience that we take for granted today.

Maintenance is another issue. Let's assume that, at the time you set up your bespoke infrastructure, it suited your organization better than any alternative. However, to paraphrase Robert Anton Wilson, it takes just two years for some brilliant software to turn into a nightmare without changing a single line of code. Unless aggressively maintained and developed, bespoke infrastructures can easily fall behind the state of the art. What was once a nimble trailblazer opening new directions for your organization can quickly become a dinosaur that holds progress back. I've heard developers complaining that their organization's bespoke development tools, probably once a source of pride, are in such a state of disrepair that they spend more time waiting for their environment to work than the time they invest in actually writing code.

Then come the development and support costs, which will include not only the (typically highly paid) engineering time needed to bring the infrastructure to life, but, just as importantly, management distraction during both its early days and its, inevitably capricious, ending ones. Add to this the opportunity cost of depriving other profitable projects of engineering resources, and the price can really go up.

But the problems don't stop here. Given that infrastructure is critical to operations, the owners of bespoke solutions can (often unintentionally) hold the organization ransom to secure cozy working arrangements. This drives down morale and encourages empire-building by piling new layers of bespoke stuff on top of existing ones. As you might expect, such vested interests in an organization stand in the way of looking at better alternatives, and the organization misses out on the benefits of the latest and greatest technology.

It takes just two years for some brilliant software to turn into a nightmare without changing a single line of code.

Finally, consider developer mobility. On one hand, developers who, for years, have been writing code in your organization's obscure programming language that no one else uses will find it difficult to get an offer that will lure them away. On the other, the smart people who work with your niche infrastructure will quickly realize that it negatively affects their career prospects and will start looking for alternatives. Thus you'll end up working only with those unfortunate souls who have nowhere better to go.

A Balancing Act

Maintain a healthy amount of skepticism regarding homebrew solutions: the cards are stacked against the adoption of infrastructure that's "not invented here." By definition, bringing in such infrastructure

means change, and this triggers peoples' conservative instincts. Developers who have learned to use the bespoke tool or library will have to learn the new one, and, worse, those who developed it will have to find other ways to contribute.

It's impossible to break new ground with established solutions, so the need to come up with a never-ending stream of bespoke solutions might just be the cost of doing business at the frontier. Yet, the problem may not be in creating and using these infrastructures, but in not letting them go when they've served their purpose.

You might hear arguments about the investment put into a bespoke infrastructure's development. Given that this is a sunk cost, it shouldn't influence your decision either way. Rather, you should simply consider the relative merits of the two solutions, the cost of the alternatives, and any switching costs. Sadly, misplaced loss aversion regarding a sunk cost often taints an organization's judgment.

If universally available tools don't quite fit the bill, consider customizing a general-purpose solution to your needs. Thankfully, modern technologies are often easily customizable via myriad configuration options, plugins, and modules. (Often to the point of absurdity; consider the 12,000 theme downloads available on eclipsecolorthemes.org.) Look for existing customizations before launching your own.

Another approach is to adopt an open source tool and improve it to address your organization's requirements. Then, cooperate with the tool's developers to contribute your changes back to the community. This isn't just out of altruism; feeding your changes back upstream ensures that they remain part of the tool in the future.

Finally, when called to make a choice, consider that the trend is toward a transition from bespoke infrastructures to widely used, general-purpose technologies. I've seen this transition happening in many organizations, often with pain and regret for the earlier decision to follow the bespoke solution sirens. When you design infrastructures, train your instinct to go with the flow: adopt and build on the best and greatest technologies used by your community.

DIOMIDIS SPINELLIS works at Google as a site reliability engineering software engineer. Contact him at dspin@google.com.

Post your comments online by visiting the column's blog:

www.spinellis.gr/tools

See www.computer.org/software-multimedia for multimedia content related to this article.