Editor: **Mik Kersten**
Tasktop
mik@tasktop.com

A Cambrian Explosion of DevOps Tools

Mik Kersten

ANY DISCUSSION OF how to scale the benefits of DevOps invariably lands on tools. The planning, tracking, automation, and management tools we use define the "ground truth" of where and how work happens. One of the most interesting, and at times challenging, aspects of agile and DevOps transformations is the sheer volume of tools involved. How many are required? Must there be so many? Before we proceed further on our journey of defining value stream architecture, let's look at how this ground truth has evolved to get us where we are today.

The Catalyst for DevOps Tool Diversification

We're at an interesting time in the evolution of DevOps tools; the sheer number of available tools points to a sort of Cambrian explosion of tool specialization and diversity. Is all this diversity necessary? Will a big wave of consolidation drive the extinction of most of these tools? What are the lines of specialization driving the diversity, and do we need to consider them when architecting our software value streams? We need to address these questions and inspect the ground truth captured in today's toolchains in order to inform the discussion of how to abstract away the tools' implementation details to focus on the architecture of our value streams.

For two decades starting in the 1980s, the company providing the majority of enterprise IT shops with software development tools was Rational. For many organizations, the entire software lifecycle was tracked within the Rational toolchain. While it's tempting to poke fun at heavyweight tools while talking DevOps, Rational created a toolchain that was incredibly sophisticated and effective for its time. Along with the tools, Rational created the Rational Unified Process (RUP), a cohesive and tool-supported process framework for software engineering. RUP provided IT and software delivery organizations with end-to-end visibility, control, and predictability for large software initiatives, thereby becoming the poster child for waterfall methodology. In the 1990s, both the toolchain provided by Rational and the process and methodologies around it expanded rapidly.

Then, in the 2000s, agile happened, largely as a reaction to problems with the command-and-control style of managing software delivery that waterfall and RUP enabled. The agile movement was followed in the 2010s by the DevOps movement, and both have now disrupted the age of waterfall. The 2017 Stack Overflow survey indicated that 76.9 percent of the respondents use agile methods, whereas 26.9 percent use waterfall (n = 25,771).[1] Although many large organizations still follow the waterfall model, the benefits of faster lead times and smaller batches that come with agile and DevOps are now part of the well-documented state of the practice.[2]

Why the Explosion?

A disruption this fundamental can bring with it a change of an entire market. In this case, the DevOps tool market formed to fill the gap created by the waterfall model's displacement. You can glimpse this disruption's scope through the GrowthPoint Technology Partners DevOps Startup Landscape Map (see Figure 1), courtesy of Jake Kaldenbaugh, who carefully tracks the tools space.

Most of the many vendors on this map are vying to provide a repository or automation layer for a segment of the software value stream. What's fascinating are both the sheer number of vendors and the distinct tool categories that have emerged.

Another piece of evidence comes from a study in which my company Tasktop examined the toolchains of 300 Enterprise IT organizations. We determined that 70 percent of those

 0740-7459/18/$33.00 © 2018 IEEE

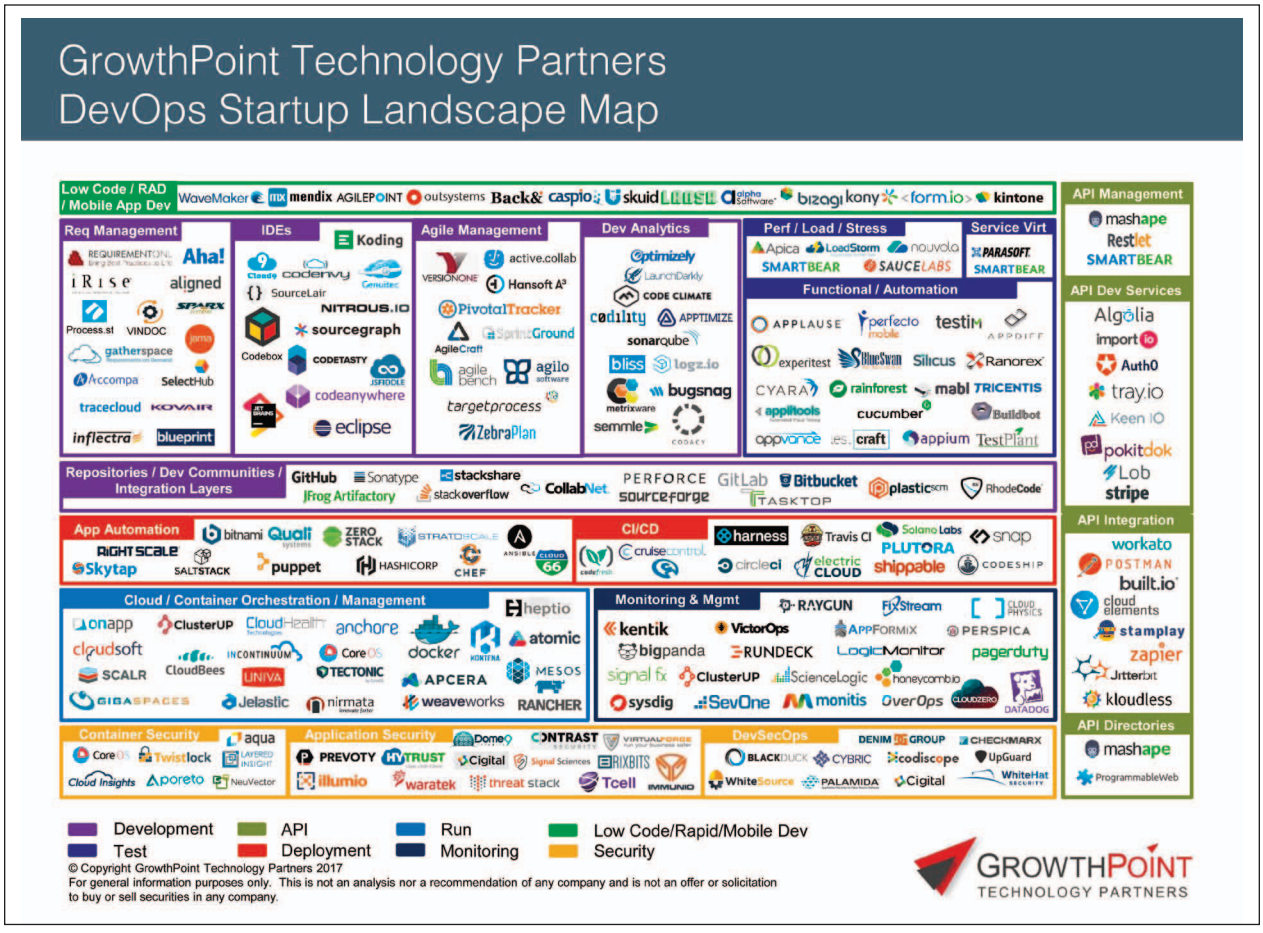

FIGURE 1. A glimpse into the exploding number of DevOps tools, which are filling the gap created by the waterfall model's displacement. (Source: GrowthPoint Technology Partners; used with permission.)

organizations already integrated three or more tools and that 40 percent integrated four or more tools.

In addition, the adoption of open source tools, such as Git, has rapidly increased over the same time frame. The 2017 Stack Overflow survey indicated that Git has achieved 69.2 percent adoption (n = 30,730), with Rational ClearCase at 0.4 percent by comparison.[1] This very rapid adoption of Git and its disruption of heavier-weight tools indicate an important trend. Agile, DevOps, and open source all have something in common: they're driven from the bottom up, with each focusing on empowering the practitioner. Like other disruptions, they represent a breakup of the top-down control model and a "democratization" of the tool chain. What's clear from Figure 1 is that this bottom-up democratization goes against the one-size-fits-all mentality of the tools that preceded it. The sheer number of tool categories indicates a specialization of tools that didn't exist before. That specialization is driven by the needs of the different types of work involved in software delivery.

As software development has scaled, practitioners have sought tools specialized for their roles. For example, a tool that tracks customer tickets and focuses on service-level agreements (SLAs) differs considerably from one that tracks issues in an agile backlog or one that's targeted at business analysts modeling customer use cases and workflows. Under the hood, the tools might appear nearly identical in terms of their data models and collaboration facilities and workflow engines. That's why in the past, organizations could use a single tool

for the different tasks. However, as the work has scaled, so has the number of practitioners.

As a result, practitioners have demanded user experiences that provide systems of engagement tailored to their role. This has pressured vendors to specialize their offerings, with the resulting Cambrian explosion of the toolchain. Consider the various categories to be different evolutionarily stable strategies for vendors, with diversity within and across categories driven by a resource-rich market of organizations building bigger and bigger software.

Dealing with Diversity

So, are these tools actually headed for a mass extinction? The analysis of the 300 organizations' toolchains revealed two types of tool diversity.

Fundamental diversity adds value by increasing software delivery productivity. For example, teams developing Java applications might be more productive using Jira, whereas teams developing with Azure and .NET might be more productive using VSTS (Visual Studio Team Services).

Accidental diversity doesn't contribute positively to organizational goals. This category includes tools inherited through mergers and acquisitions or similarly functioned tools that were selected independently owing to a lack of centralized governance. For example, an organization could have three bug trackers: a 20-year-old legacy tool created in house, a new developer-favored issue tracker, and an open source issue tracker that resulted from an acquisition.

From a value-stream-architecture viewpoint, both types of diversity must be accounted for. Accidental diversity should motivate organizations to consolidate and rationalize. This activity is relatively straightforward; it simply implies that the value stream should contain only one tool for each required tool category. What's more problematic is when organizations can't distinguish between accidental and fundamental diversity.

While examining value streams, we've identified six varieties of fundamental diversity:

- *Stakeholder specialization*. The various stakeholders of software delivery require different tools to be effective for their particular discipline. For example, support people need tools that support SLAs or ITIL, whereas developers need tools streamlined for code review and commit.
- *Scale specialization*. Some tools are specialized according to organizational size. For example, a lightweight Kanban tool can be great for streamlining the flow of a dozen teams, but a hierarchical requirements tool is needed for tracking the requirements of safety-critical systems.
- *Platform specialization*. Vendors who provide a development platform often provide a tool-based on-ramp onto that platform. For example, Microsoft provides end-to-end DevOps and agile solutions that are optimized around Azure as the deployment platform but are less tailored to the more heterogeneous Java ecosystem.
- *Zone specialization*. In *Zone to Win*, Geoffrey Moore identified zones of varying stages of maturity for businesses to focus on.[3] A more experimental *transformation zone* product might require only the most lightweight and experimental tools, such as GitHub's simple issue-tracking features. More mature products, such as those in the *performance zone*, might require closer integration with business requirements and planning.
- *Supplier diversity*. As outsourcing and consumption of open source software increase, it becomes impractical to expect software suppliers to use the same tools as the sourcing organization. For example, open source projects tend to use open source tools, and small suppliers tend to use lightweight tracking tools instead of the enterprise tools needed for large-scale software delivery.
- *Legacy*. The cost and disruption of moving away from a legacy system, such as an older tool or in-house defect tracker, can be overly high. This is especially the case for established products in maintenance mode or in the performance zone. These tools can be another source of diversity if modernizing them is overly disruptive.

Although all organizations should aim to weed out accidental diversity, the norm today is a heterogeneous, best-of-breed toolchain. While the fast growth of startups and new vendors means some consolidation seems inevitable, enough need exists for specialization that I predict the heterogeneity will grow further before shrinking.

For example, enterprise IT organizations are starting to move away from the project-aligned value streams that have a lifecycle aligned to the project time frame and budget. Instead, organizations are employing the software-and-tool-vendor

approach of product-oriented value streams. This shift from project to product is resulting in the growth of yet another category of product management tools, somewhere between traditional requirements and agile planning.

Also, as software development becomes more complex, so will the specialization of the toolchain. This is similar to other fields (such as the medical field) in which the benefits of the division of labor have caused ever-increasing specialization in expertise. As software complexity grows, so will the number of specialized practitioners, driving further specialization of the tools.

The problem is that, when we try to analyze and improve how software is built, the morass of tools makes it difficult to see the forest for the trees. No single tool has a model of the end-to-end system. Yet value streams will continue to be defined in tools, with each delivery stage implemented in a specific tool's scheme and workflow model. But to take the next step in DevOps, we need to start thinking end-to-end. The only way to achieve that is to establish an architectural discipline for managing the layer above the toolchain.

The evolution of toolchains will continue, as will the specialization that meets the needs of each stakeholder involved in software delivery. Here, I outlined why this specialization, and the resulting tool diversity, is a fundamental aspect of the modern DevOps toolchain. Stay tuned for how we approach raising the abstraction level and modeling the layer above the toolchain.

ABOUT THE AUTHOR

MIK KERSTEN is the Founder and CEO of Tasktop. Contact him at mik@tasktop.com or follow @mik_kersten.

References

1. "Developer Survey Results: 2017," Stack Overflow, 2017; insights.stackoverflow.com/survey/2017.
2. G. Kim et al., *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, IT Revolution, 2016.
3. G.A. Moore, *Zone to Win: Organizing to Compete in an Age of Disruption*, Diversion, 2015.