DevOps and Its Practices

Liming Zhu, Data61 | CSIRO
Len Bass, Professional Education Consortium
George Champlin-Scharff, IBM
Adopting DevOps

As we said, migrating an organization to a microservices architecture...
with the associated introduction of DevOps practices involves both technical and cultural transformations. In “DevOps: Making It Easy to Do the Right Thing,” Matt Callanan and Alexandra Spillane focus on both the technical and cultural issues associated with introducing a continuous-delivery pipeline.

DevOps is in its infancy in terms of its adoption curve. Two of the three articles in this theme issue deal with adoption issues. Some future DevOps issues are foreseeable. One clear question is, “Which practices are best for which kinds of systems in which kinds of organizations?” DevOps practices grew up in organizations providing services over the Internet with, essentially, one very complex and large system. Although Amazon, Netflix, and Google have evolved their systems since introducing them, the system elements are basically extensions of the same family. This isn’t true for the systems used in a financial institution such as a bank. The mind-set involved in evolving such systems differs from the mind-set involved in integrating two similar systems or performing many enterprise-engineering roles.

Another question is, which domains might benefit from DevOps practices? One such domain is big data systems. Many big data systems rely on rapid deployment to support their data pipeline; thus, big data systems will rely more and more on DevOps practices.

**Tool-related DevOps practices will also evolve.** Currently, specialized tools exist for each portion of a pipeline. However, the overall pipeline flow must be hand-tailored using an orchestration engine or specialized plug-ins for existing tools. Tools or tool families will emerge that are aware of the whole pipeline and that manage the orchestration of and configuration parameters for each pipeline stage. One step in that direction is ThoughtWorks’ GoCD tool (www.thoughtworks.com/go). One analogy to tool evolution is programming-language evolution. Although it’s possible to do everything in assembly language or C, domain-specific languages provide the abstractions that make specifying applications in the target domain easier.

This theme issue can only touch the surface of all the issues associated with DevOps. However, if you are migrating to microservices or have implemented them and are now dealing with postdeployment monitoring and reliability challenges, you should find this issue’s articles helpful.

**References**