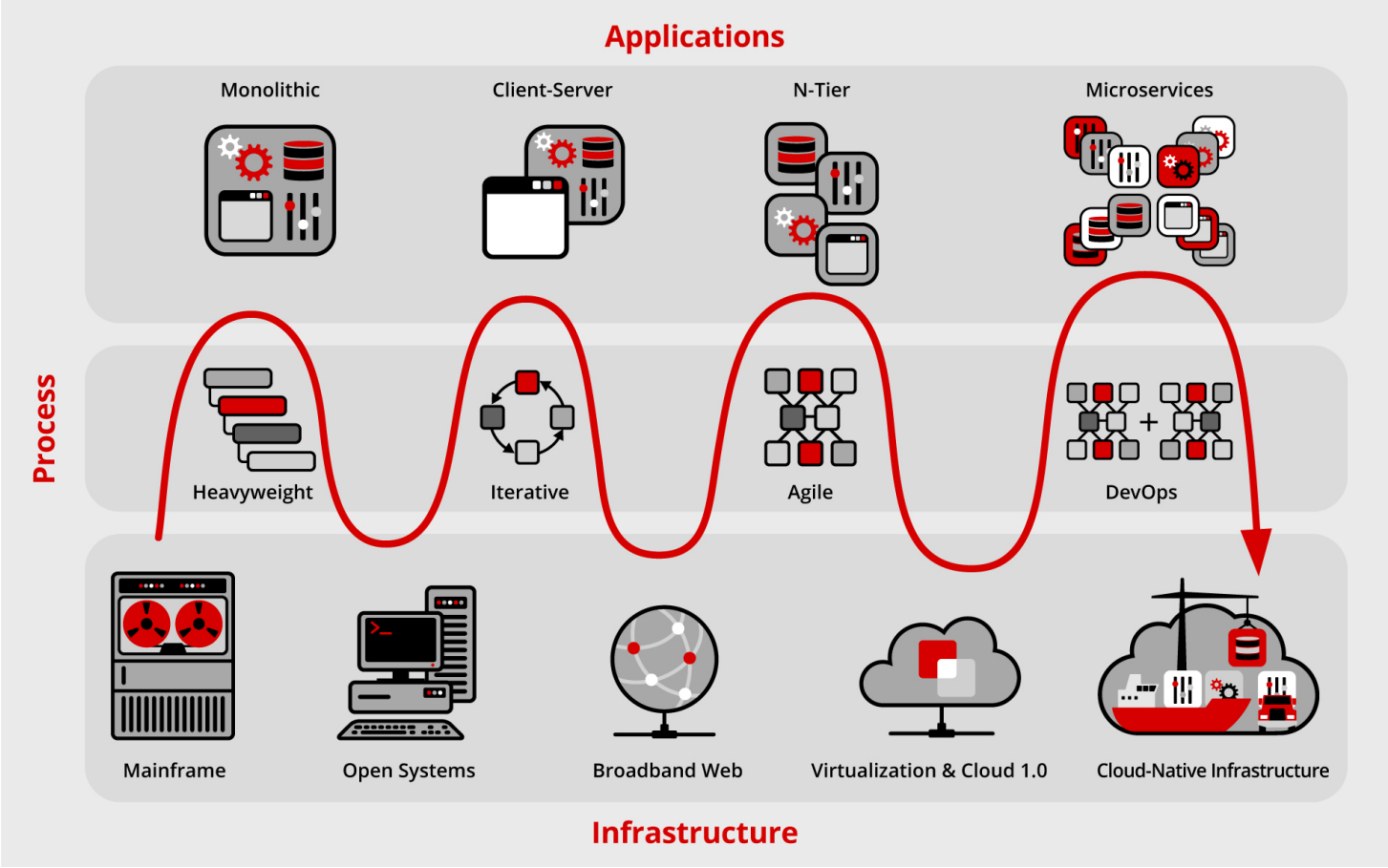
WiNG

DevOps And The Digital Trialectic

MAY 16, 2016 PETER WAGNER INFRASTRUCTURE

Earlier this year, we published a blog post (http://wing.vc/blog/microservices-containers-and-the-digital-dialectic) that highlighted the interplay over time between application architectures and technology infrastructure. In it, we explained how these two areas have evolved in a ping-pong relationship, with changes in IT infrastructure driving subsequent changes in application architectures, which then inspire further infrastructure innovations. We dubbed this the “Digital Dialectic” and our post showed how this relationship has played out from the early days of mainframe computing and monolithic applications, through to the rise of microservices and what we called “cloud-native infrastructure".

The Dialectic is a powerful trend that has given rise to several generations of new companies with large market capitalizations, and we’re confident that its latest phase will produce yet more significant and valuable businesses. But another dimension that was left out of our original analysis also merits attention: the evolution of software development methods and IT operations. Shifts in these areas have contributed to—and been influenced by—changes in other dimensions of the Dialectic. The relationship between all three of these areas—the “Digital Trialectic”—is worth exploring in greater depth to fully appreciate the significance of what is happening today. (The Trialectic is depicted in graphical form here.)



The interplay between infrastructure, applications and software development/IT operations was already evident in the mainframe era, which not only gave rise to monolithic applications, but also to “heavyweight” software development approaches. A classic example was the Software Development Life Cycle, or Waterfall, method. This involved establishing very detailed requirements and schedules, with the goal being to develop software applications in a sequential fashion, from idea to final delivery. Such methods, which involved huge amounts of bureaucracy, were designed to ensure scarce computing resources were not wasted. But they made it very hard and costly to change the software when business needs shifted.

Restricted access

The typical approach to IT operations made matters worse. There was a rigid segmentation between developers and operations teams, with the latter carefully restricting access to computing capacity like a kind of digital Praetorian Guard. Developers wanting to revise code had to wait their turn with their noses pressed to the panes of the glass houses in which the mainframes—and their minders—were located.

Frustrated by this rigidity, which prevented IT from moving at the speed of business innovation, developers began adopting “Iterative” methods such as dynamic systems development and, over time, scrum, extreme programming and other approaches. Unlike their heavyweight predecessors, these methods broke down monolithic projects into smaller chunks, and introduced an iterative process that made it possible to react swiftly to customer feedback at various stages of development. The adoption of iterative methods was accelerated by the shift towards open systems, which encouraged third-party software creation.

The emergence of the web as a computing platform signaled the next phase in the Digital Trialectic. Once again, an infrastructure innovation led to a new application paradigm in the form of N-tier application design. On the development front, iterative methods had continued to gain ground and in 2001 a group of software engineers published the “Manifesto for Agile Software Development”, which marked the coming of age of the new approach with its emphasis on cross-functional teams and process adaptability.

In the following years, it quickly became clear that internet technology would have a huge impact on the way software was being developed and delivered. Increasingly, packaged software gave way to a world in which development was taking place on the server-side for web applications. While this eradicated some of the problems with traditional software development, it also posed new challenges. These included dealing with the complexity of N-tier application architectures, and the need to monitor multiple QA and user acceptance testing environments. The arrival of virtualization on the scene created the foundations for the rise of the public and private clouds. It also changed the lives of IT operations teams for the better by helping to create dynamic and programmable infrastructure.

Resolving tensions

Yet problems remained. As software became more central to companies’ competitive advantage, the pressure from business teams to deliver code fast increased. This led to tension between developers, who saw operations often slowing down releases and frequently resisting changes being pushed into production because of quality concerns, and ops teams, which frequently accused developers of trying to cut corners in order to get code “over the wall”, leaving the operations side to work long hours clearing up the mess.

Frustration with this state of affairs eventually led to efforts to break down the silos between development and operations, giving rise to the DevOps movement that emerged in the latter part of the last decade. While still leveraging the advances of agile development, this created a more collaborative environment between development and operations teams. Part of the change has been cultural, with adherents taking steps such as co-locating teams from both sides. But it has also led to process-driven change. Examples of this include the adoption of continuous delivery pipelines, which involve testing everything that goes into an application before deployment to production, and the development of tools such as Chef and Puppet that automate previously time-consuming processes in application deployment.

This shift in application development and operations processes has occurred alongside the rise of a new application architecture in the form of microservices, marking another phase in the evolution of the Digital Trialectic. The idea of developing a single application as a suite of small services organized more closely around business capabilities is perfectly suited to the desire to get code into production faster and with fewer glitches, and has been championed by web-native firms such as Amazon and Netflix. By making services independently deployable, with each one running its own process and communicating via lightweight APIs, the new approach means systems failures from continuous delivery of code are less likely to occur.

Ready for the next twist

DevOps and microservices have been powerful motivators of interest in a new kind of infrastructure, which we call "cloud native". Containers are one of the more notable aspects of this new approach and have been championed by developers keen to ensure that the code they write on their laptops runs smoothly through the phases from test to production. The fact that interaction with containers happens via APIs also makes them ideally suited to microservices. But there are other aspects of this new infrastructure that could have a significant impact in the years ahead. Once cloud-native infrastructure has become firmly embedded in businesses, we look forward to seeing how it impacts the next twist in the Digital Trialectic.

© 1990 — 2018 Wing, Menlo Park, CA