Independent market research and competitive analysis of next-generation business and technology solutions for service providers and vendors

HEAVY READING
WHITE PAPER

Simplifying Continuous Deployment with A/B Test:
A Phased Approach to VNF Upgrades

A Heavy Reading white paper produced for Huawei

AUTHOR: JAMES CRAWSHAW, SENIOR ANALYST, HEAVY READING

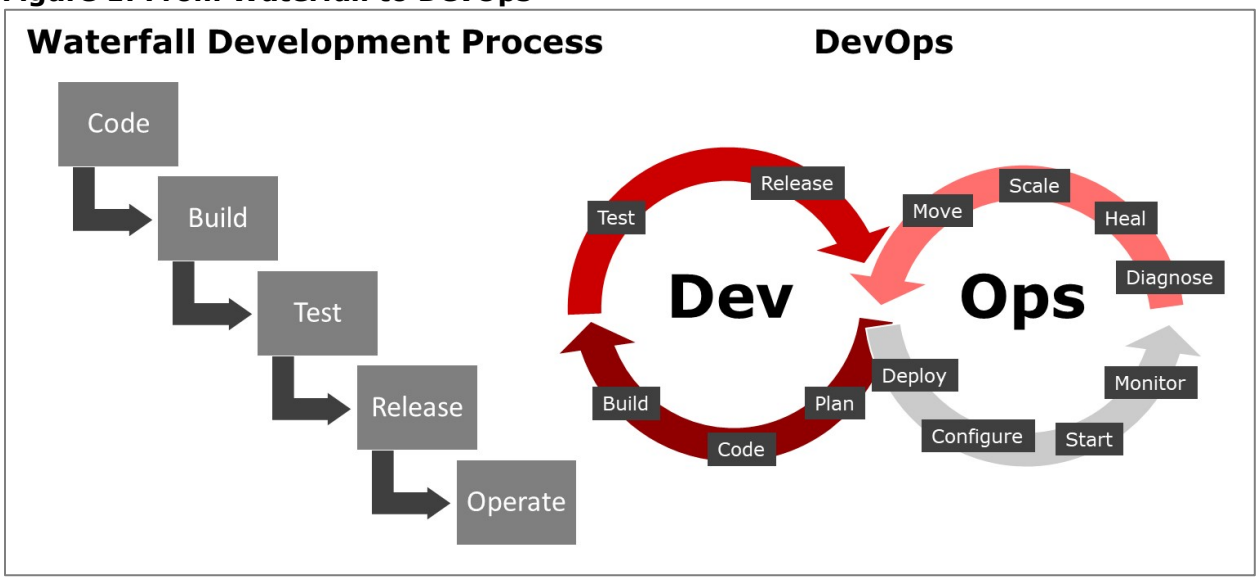
DEVOPS AND CI/CD IN THE TELECOM INDUSTRY

As telecom networks become increasingly software-centric (or software-defined), their operation and management are becoming more and more similar to general IT infrastructure. As such, practices such as agile software development, DevOps, continuous integration (CI), and continuous delivery/deployment (CD) are gradually being adopted in the network operations domain.

From Waterfall to DevOps

Figure 1 contrasts the traditional waterfall development process with the more modern DevOps approach. With the waterfall process, there is a handoff between individuals or teams as organizations move from the code and build stage to testing, release management, and operations. This can lead to long project times, as work is done in series instead of in parallel. Additionally, individuals working on each stage in the process will seek to maximize their own link in the chain rather than those of the project overall. The idea behind DevOps is that developers, testers, and those responsible for operating the software all work in much closer cooperation. By developing software in short, incremental sprints, organizations can incorporate more frequent feedback into the development process, which makes it easier to fix any bugs that are identified.

Figure 1: From Waterfall to DevOps



Source: Heavy Reading

DevOps for CSPs

Webscale internet companies have been at the forefront of the move to DevOps and CI/CD, enabling them to make massive and frequent changes to their software and hardware systems:

- Amazon can make 10,000 software deployments per hour and suffer a resultant outage rate of just 0.01% (10^{-4}).
- Google makes around 17,000 code submissions per day, leading to half of the source code being changed each month and requiring more than 100 million test cases to be run every day.
- Facebook makes 10,000 code submissions each month, equating to over 10 million lines of code.

However, the telecom industry operates under different constraints to the web giants. The most obvious difference is that, outside of legacy operations support/business support systems (OSS/BSS), telecom operators do not generally maintain large software code bases. Typically, they source software solutions from third-party vendors. The operator may provide feedback to its suppliers about software enhancements as part of a DevOps process. However, from an operational perspective, the communications service provider (CSP) will mainly be focused on deploying new software versions as they are released.

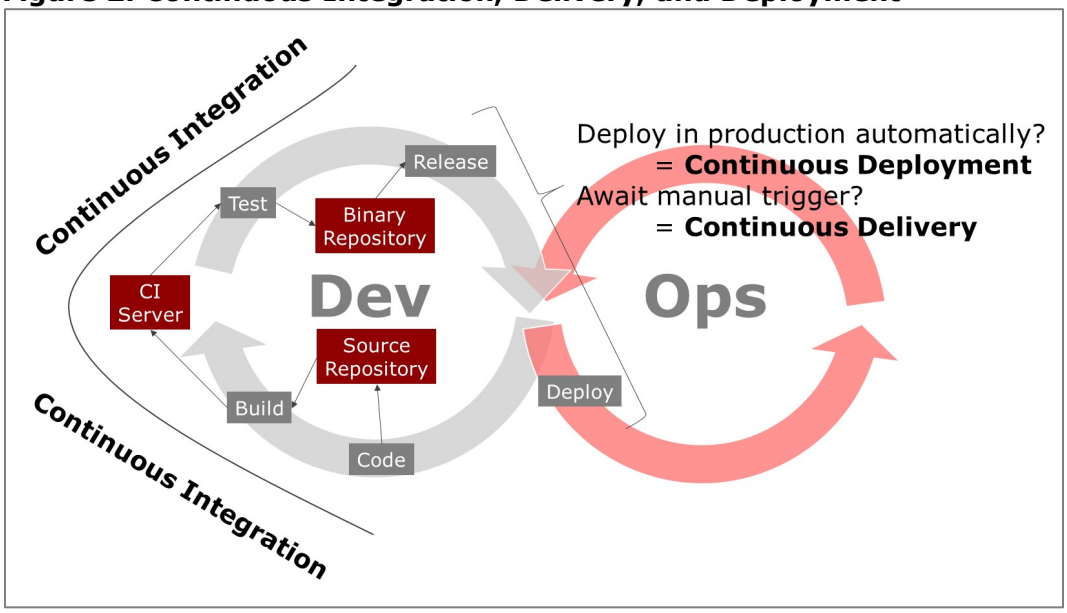
The other major difference between web giants and telecom operators is the degree of risk that they are willing to tolerate regarding outages. Telecoms is a highly regulated industry and is responsible for emergency services call center communications, first responder networks, lawful intercept, and location reporting. Telecom operators have an ingrained culture of risk avoidance, and their perception historically has been that more software updates equate to more risk. However, the example of Amazon above (four-9s reliability) demonstrates that, in practice, small incremental software updates that can be easily rolled back can be safer than large, infrequent updates. Small code changes allow bugs to be identified and fixed more quickly.

Enabling DevOps with CI/CD Tools

CI/CD and DevOps are related concepts, as **Figure 2** suggests. The main difference is that CI/CD concerns automation tools that can facilitate the high frequency of software builds, tests, and deployments. There are many different CI/CD tools, some of which are open source. Categories include source code repositories (e.g., GitHub, GitLab), CI software (e.g., Jenkins), code quality checkers (e.g., Checkmarx), and binary repository managers (e.g., Archiva, Sonatype Nexus). In contrast, DevOps is more related to a mindset or organizational approach; it is essentially about breaking down the barriers between engineering (or software development) and operations.

CI/CD tools are essential for automation and operational efficiency. From a CSP's perspective, the automation of software update deployments is becoming an increasingly important aspect to consider in ongoing operations efficiency, as discussed below.

Figure 2: Continuous Integration, Delivery, and Deployment



Source: Heavy Reading

The aim of CI is to quickly identify defects in the code base and correct them as soon as possible. Iterative software changes can be independently tested and then added to the main code base. If a change in the main code base creates issues, the change can be more easily identified if the updates are more frequent (and therefore the changes to the code base are smaller). Because CI detects issues early, problems become easier to resolve.

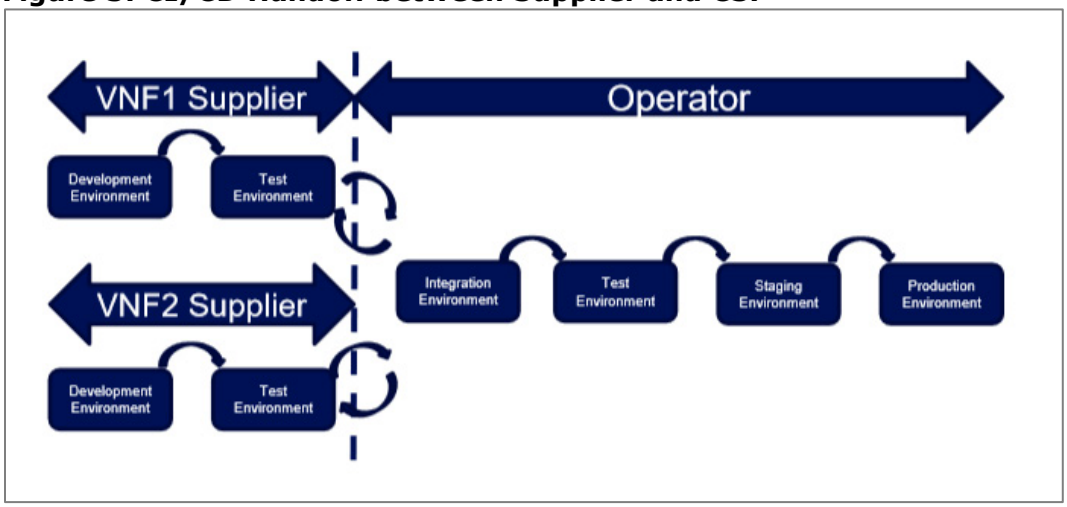
CD deals with the movement of software from the development phase into the production environment. The difference between delivery and deployment relates to the degree to which this process is automated (deployment) or manual (delivery). CD comes with its own testing and staging processes before the software goes into production. Some tests might not be easily automated or there might be other operational or business reasons why organizations would opt for continuous delivery rather than deployment (e.g., automatic live placement of every main branch change in the code base).

SOFTWARE DEPLOYMENT IN CSP ENVIRONMENTS

Within a telecom operator environment, the software development and delivery are often done by a third-party supplier while the software deployment is done by the telecom operator's own operations staff. **Figure 3** shows this handoff point across organizational

boundaries. With the move to DevOps, the operations team is faced with more frequent software updates and hence there is a need for greater automation than in the past.

Figure 3: CI/CD Handoff between Supplier and CSP



Source: Dr. Marcus Brunner, Chief Researcher, Swisscom, Network Virtualization Europe, Madrid, May 2017

When the operator makes software updates to its network, it clearly wants to minimize the impact on services. The in-service software upgrade (ISSU) is a method for updating the software on a network device without service interruption. It requires some duplication or redundancy of control plane elements such that they can download new software and be rebooted without having to affect the data plane of the device (e.g., reboot it). The Internet Engineering Task Force (IETF's) RFC 7654 notes that even when using ISSU, "most service providers will still undertake such actions in a maintenance window (even in redundant environments) to minimize any risk."

When upgrading the software of complex systems running on dedicated hardware (or adapted to run on virtual machines or containers), operators will often perform these operations during the night when network utilization is low and customer impact is likely to be negligible. Depending on the size and complexity of the update, the whole process might take 10-20 minutes.

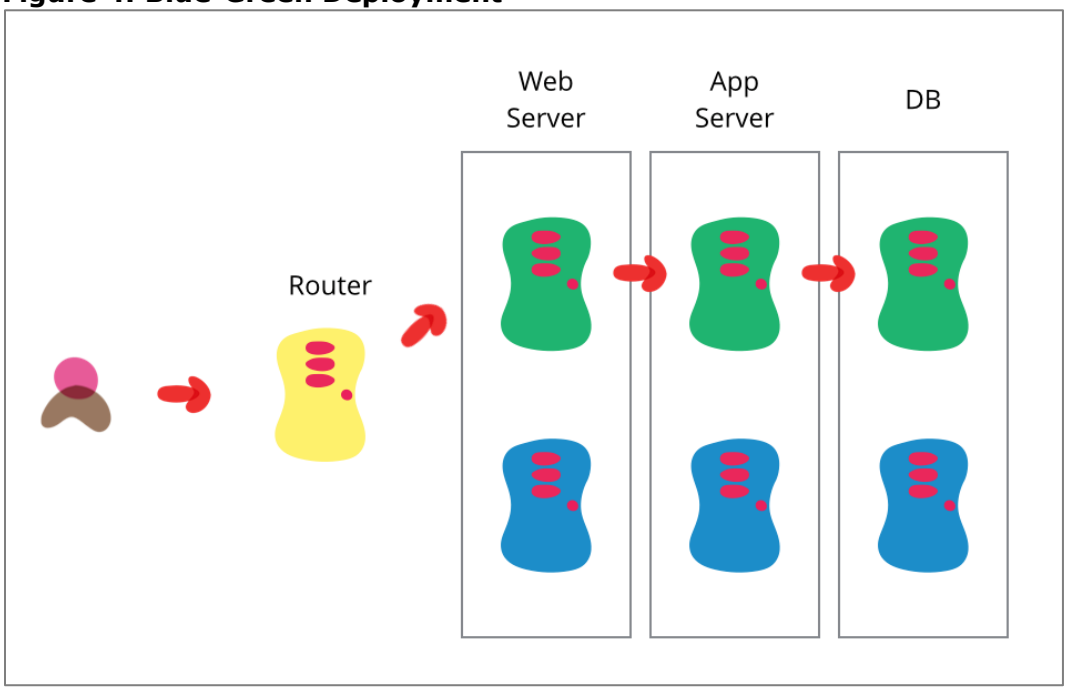
By copying the ISSU approach of partial redundancy, operators can use a second server (or collection of virtual machines and containers) to install the new software while leaving the first server online. Although this redundancy might not be an issue for small systems, for larger network functions that require significant computing capacity, it is not a cost-effective approach.

Phased Software Upgrades

An alternative approach to standard ISSU cutovers is to do a phased upgrade of the system. Some IT industry experts describe this as Blue-Green deployment, where the software is transitioned gradually from the blue variant to the green. Others describe it as a Canary Release in reference to the birds that miners used to take down the mine as an early warning of gas (i.e., a test for a problem). Huawei describes this approach as A/B Test given its similarities to the variant testing seen in website design and other fields.

Back in 2010, Martin Fowler, a renowned software development expert, published an article about Blue-Green deployment, which summarized a section from the book *Continuous Delivery*. Fowler describes the Blue-Green approach as having two identical production environments; blue is live and green is undergoing testing of a new software release. Once the operator is happy that the new software is working correctly, they switch the "router" so that all incoming requests go to the green environment while blue becomes idle (ready for new software to be downloaded and tested or ready for a rollback if unforeseen problems emerge with green). The two environments could be different physical servers or different VMs running on the same (or different) hardware.

Figure 4: Blue-Green Deployment

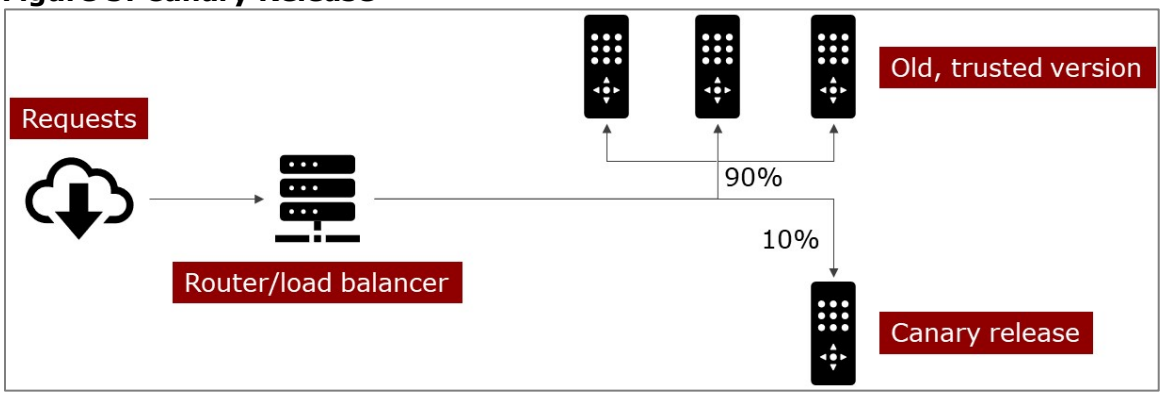


Source: MartinFowler.com

The Canary Release approach is also described in an article on Martin Fowler's blog. The approach is similar to Blue-Green, but both the old and new version of the software are run concurrently. The new version of the software is sent a small portion of incoming requests from the router while the old (and trusted) version continues to serve the majority of requests. The guinea pig users could be randomly selected or chosen according to some criteria. For example, some companies will test a new version on employees first before

releasing to customers. As organizations gain more confidence in the new version, they can start releasing it to more servers in their infrastructure and routing more users to it.

Figure 5: Canary Release

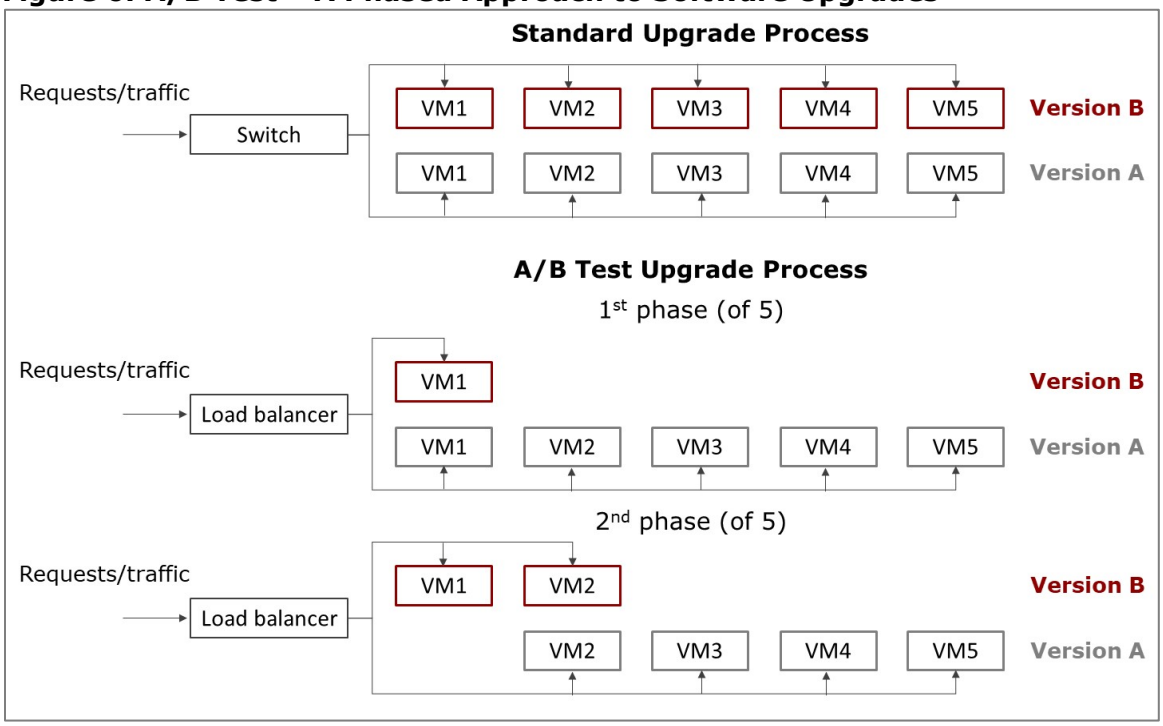


Source: Heavy Reading

A/B Test

The A/B Test approach that Huawei advocates is similar to the two-sample hypothesis testing that is commonly used to optimize website design (see this Wikipedia entry for details). A/B Test is similar to the Canary Release in that it starts with test users (guinea pigs) and then gradually switches more requests from the old version of the software (A) to the new (B). **Figure 6** compares the typical approach for virtual network function (networking software) upgrades seen today in the industry with the A/B Test approach.

Figure 6: A/B Test – A Phased Approach to Software Upgrades



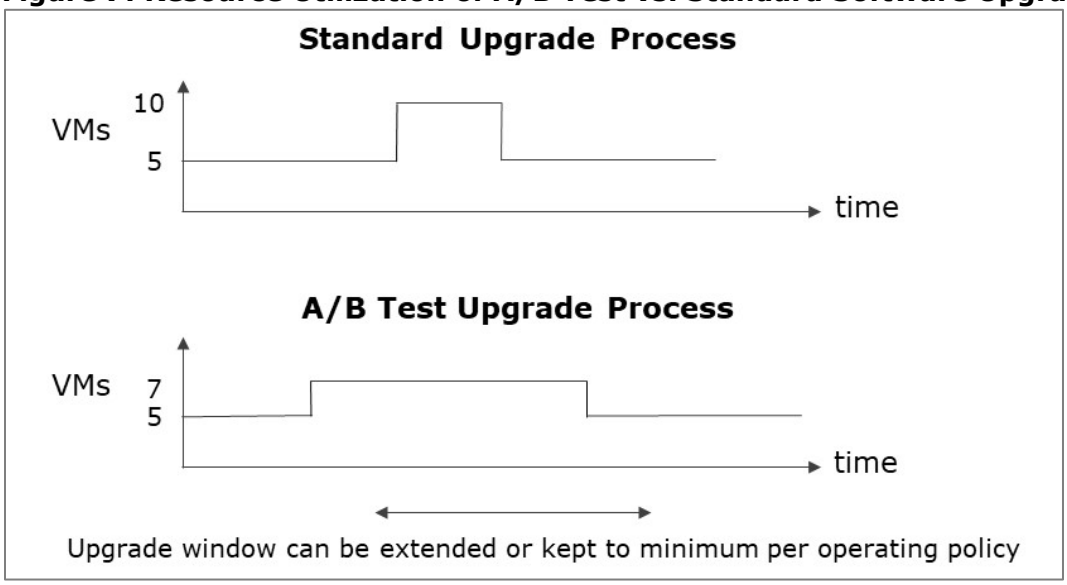
Source: Heavy Reading

The standard approach involves provisioning a replica of the resources (VMs), installing the new software on the replica resources (version B), switching all the traffic from version A VMs to version B, ensuring there are no unforeseen problems with version B, and then releasing the version A resources (VMs) back to the pool. The number of VMs used doubles during the upgrade window.

With the A/B Test approach, only two redundant VMs (or containers) are required during the upgrade process, not an entire replica of version A resources. If the VNF is running on five VMs, that means the additional resources required during the upgrade window is just 40%, as opposed to the 100% with the standard upgrade approach. Clearly, this fixed overhead for the A/B Test upgrade is smaller in percentage terms for VNFs that require a larger number of VMs (e.g., 20% overhead for a 10 VM application, 10% for 20 VMs) Once a VM with version B is added, the load balancer adds this to its list of VMs and starts draining the traffic from the first VM running version A. As with the Canary Release approach, the migration period can be extended to mitigate the risk of unforeseen problems occurring with version B so that a rollback can be undertaken while affecting only a subset of customers.

Figure 7 compares the resource utilization of the two approaches. The standard upgrade process requires users to double the number of VMs dedicated to the VNF during the upgrade process whereas the A/B Test only requires two additional VMs.

Figure 7: Resource Utilization of A/B Test vs. Standard Software Upgrade

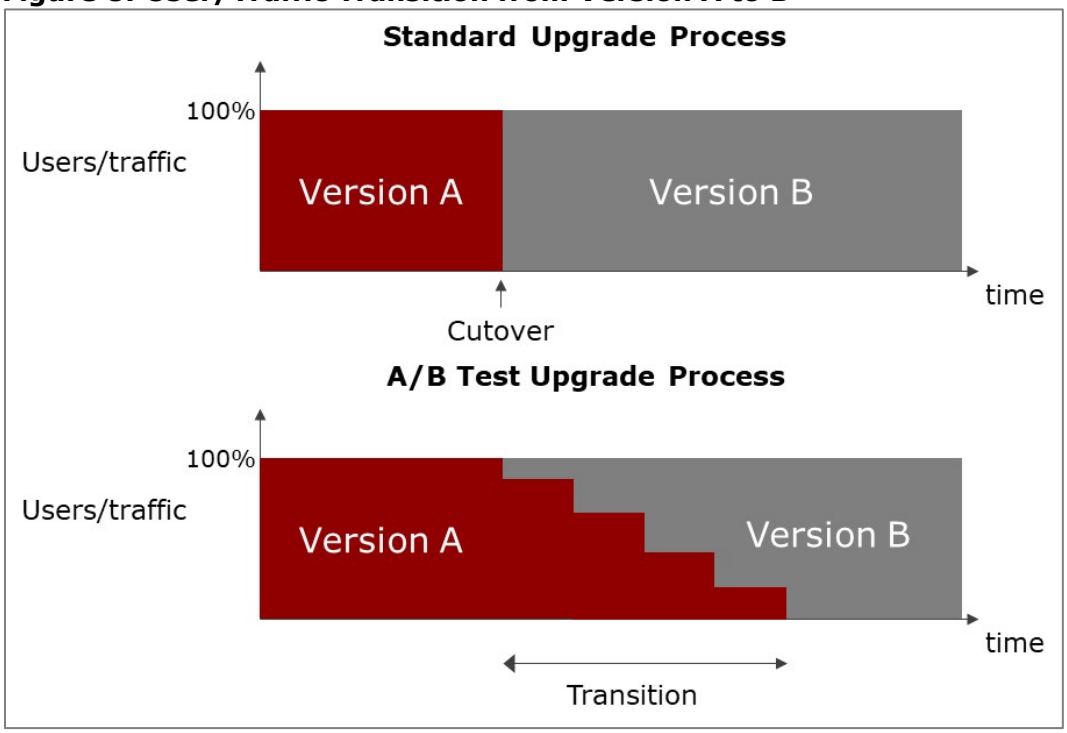


Source: Heavy Reading

Figure 8 shows how users (or traffic) is split between the two software versions during the upgrade process. The standard process has an abrupt cutover of all users from version A to B. The A/B Test process has a phased transition, with test users initially being routed to version B and then successive batches being switched over. If any problems are detected

with version B during the transition period, not all users will have been affected and a fast rollback to version A can be initiated.

Figure 8: User/Traffic Transition from Version A to B



Source: Heavy Reading

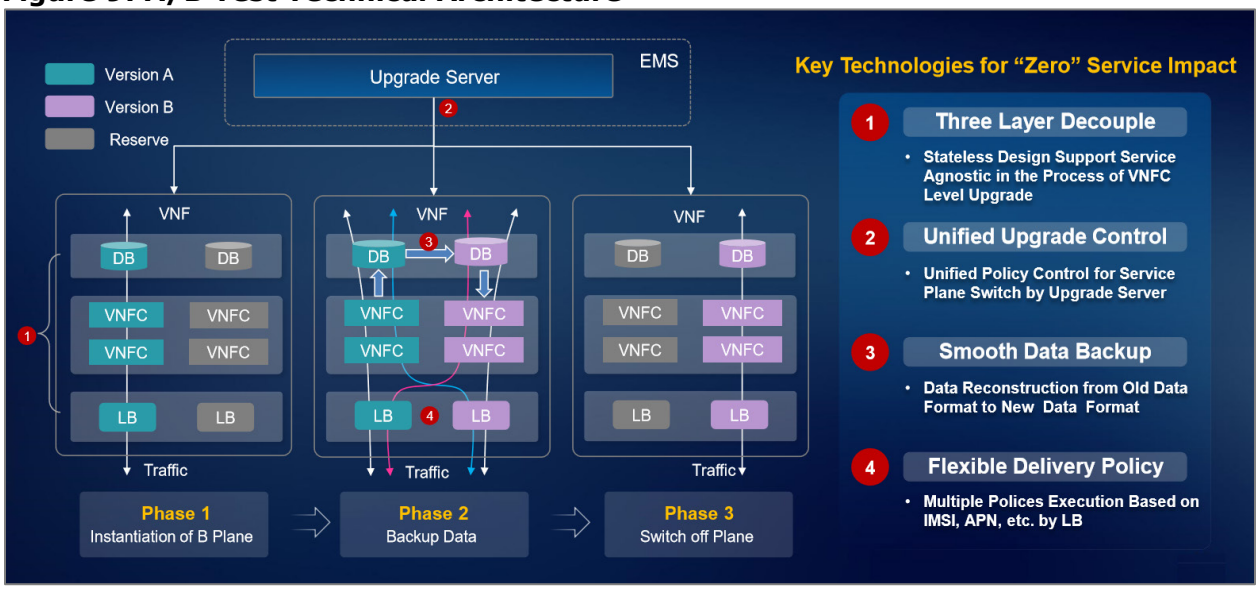
Technology Enablers for A/B Test

As **Figure 9** shows, the A/B Test approach splits the VNF into three layers:

- Session database (DB: using a stateless design for multi-version data format compatibility)
- Service processing (VNF Component: can archive multiple instances and versions)
- Service load balancer (LB: hides the internal topology from neighboring network elements)

An upgrade server manages the process, a database backup is made for each VNF, and each load balancer switches traffic from the plane running version A to version B.

Figure 9: A/B Test Technical Architecture



Source: Huawei

Operator A/B Test Case Studies

In March 2018, Huawei announced that it had demonstrated the A/B Test with Vodafone. A new Virtual Evolved Packet Core (vEPC) software version (supporting voice over LTE) was installed on a VM in the data center, in tandem with the VMs running the old version of the software. Subscriber data was synchronized batch by batch from the old version to the new version, until all VoLTE services were migrated to the new version. Performance and service quality tests were performed at each step to ensure no unforeseen problems had arisen with the new version. The whole migration was automated; the operations team simply monitored the migration in case of any problems.

In August 2018, Huawei announced that it had implemented a commercial, carrier-grade A/B Test solution with a Tier 1 Latin American operator. This operator had struggled with the traditional software upgrade process, which was complex, error-prone, and had to be performed at night to minimize the risk of disruption. With the A/B Test approach, the operator was able to upgrade VNFs in three batches during the day. The entire upgrade process was enabled with a single click of a mouse. Importantly, neighboring VNFs that were not being upgraded were not affected by the migration, unlike the traditional process.

BENEFITS OF A/B TEST APPROACH

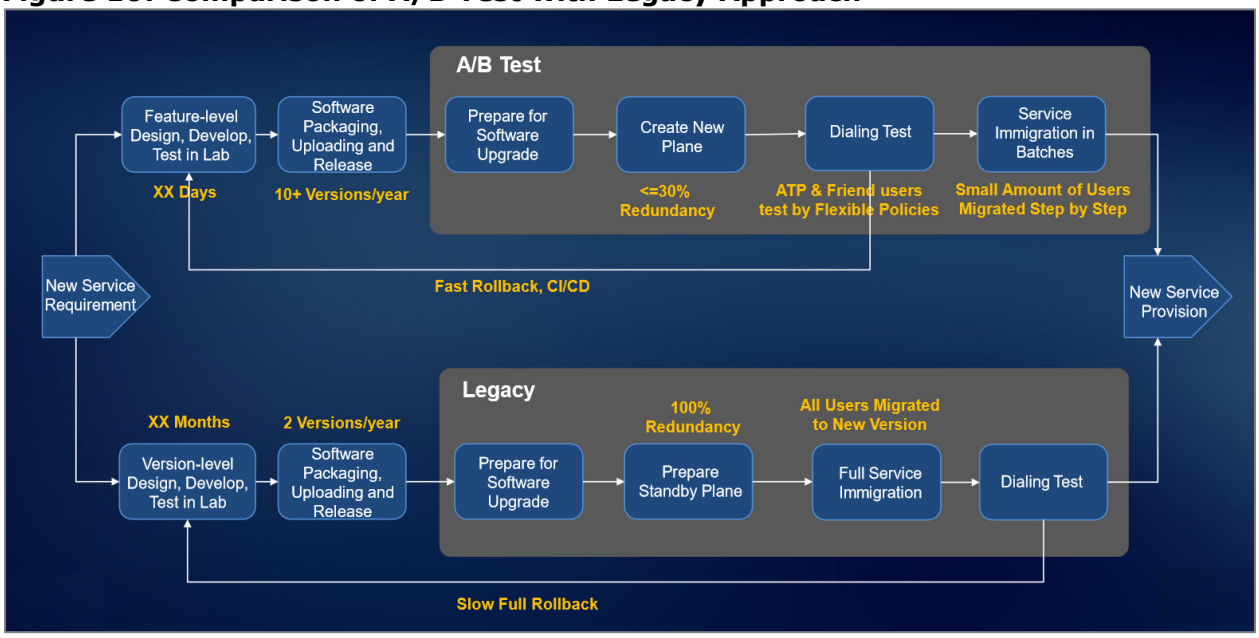
A traditional software upgrade with physical network functions is a complex and lengthy process. For each site where an upgrade is due to take place, the operator needs to obtain the software packages, verify the software in a testbed, prepare the upgrade script, upload the host software, activate the host software, migrate customers, and verify the upgrade has been successful. The entire process can take several months. Unless the system supports ISSU, the operator would also be faced with service interruptions of around 15

minutes during the upgrade window. Consequently, the operations team would be forced to carry out upgrades in the middle of the night to minimize service impact.

With the move to network function virtualization, operators are looking for a more agile approach to VNF updating that allows smaller and more frequent updates to software, in line with DevOps best practices. This agile approach to upgrades should avoid any impact on the services running on the network, and it should consume the minimum amount of network resources. By simplifying the software updating/upgrading process network, operators should be able to deploy new versions more quickly. They can also roll back to prior versions of the software with relative ease. This agility should accelerate the time to market of new network inventions and foster a greater culture of innovation.

As software upgrades move from two or less per year to 10 or more, there needs to be a significant simplification in the planning and preparation process so that this can be reduced from months to a matter of days, as indicated in **Figure 10**. Because software upgrades will be much more frequent (and potentially contemporaneous), operators will want to minimize the amount of resources (VMs and staff) during the upgrade process. As shown below, the phased approach of AB/Test can reduce the amount of resources (VMs) tied up in software upgrades. Highly automated testing can also reduce the amount of staff required.

Figure 10: Comparison of A/B Test with Legacy Approach



Source: Huawei

As **Figure 11** summarizes, the phased approach of A/B Test holds several benefits over the current industry norm for VNF upgrading. Service downtime can be eliminated. Upgrades can be done gradually during the day when more of the operations team is available to deal with any unexpected problems. Fewer redundant resources (VMs) are required during the upgrade window. The upgrade process itself is highly automated reducing the amount of staff needed to manage it.

Figure 11: Comparison of A/B Test with Current Industry Norm

Factor	Industry Norm	A/B Test
Service Impact	No interruption with ISSU but "out-of-service" upgrades can require interruptions of 10-20 minutes	No interruption
Upgrade Window	Typically done at midnight when traffic is low	Can be done any time
Resources Required	Need 100% redundancy during ISSU upgrade, as new version is deployed in tandem with existing before cutover	Only need two additional VMs to host new VNF; as new VNFs are booted, the old VNFs are retired in steps and their VM resources reallocated
Service Agility	Low – many manual steps involved in loading software, rebooting VMs, etc.	High – "one-click" upgrade process, automated testing; the service load balancing aspect enables the user to hide the internal topology of the system from neighboring VNFs, thereby avoiding the need to make any changes to neighbors to support the upgrade

Source: Heavy Reading