451 Research® | Advisory

Application Deployment Agility, Declarative Models and Constraint Solutions

BY PETER CHRISTY - RESEARCH DIRECTOR, NETWORKS, 451 RESEARCH

How Cisco and SAP have partnered to leverage formal application models and constraint-optimization software to automate and speed the construction and deployment of some of the most complex business applications in use today.

Research Commissioned by

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

OCTOBER 2015

ABOUT 451 RESEARCH

451 Research is a preeminent information technology research and advisory company. With a core focus on technology innovation and market disruption, we provide essential insight for leaders of the digital economy. More than 100 analysts and consultants deliver that insight via syndicated research, advisory services and live events to over 1,000 client organizations in North America, Europe and around the world. Founded in 2000 and headquartered in New York, 451 Research is a division of The 451 Group.

© 2015 451 Research, LLC and/or its Affiliates. All Rights Reserved. Reproduction and distribution of this publication, in whole or in part, in any form without prior written permission is forbidden. The terms of use regarding distribution, both internally and externally, shall be governed by the terms laid out in your Service Agreement with 451 Research and/or its Affiliates. The information contained herein has been obtained from sources believed to be reliable. 451 Research disclaims all warranties as to the accuracy, completeness or adequacy of such information. Although 451 Research may discuss legal issues related to the information technology business, 451 Research does not provide legal advice or services and their research should not be construed or used as such. 451 Research shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice.

New York
20 West 37th Street, 6th Floor
New York, NY 10018
Phone: 212.505.3030
Fax: 212.505.2630

London
Paxton House (5th floor), 30 Artillery Lane
London, E1 7LS, UK
Phone: +44 (0) 207 426 0219
Fax: +44 (0) 207 426 4698

San Francisco
140 Geary Street, 9th Floor
San Francisco, CA 94108
Phone: 415.989.1555
Fax: 415.989.1558

Boston
1 Liberty Square, 5th Floor
Boston, MA 02109
Phone: 617.275.8818
Fax: 617.261.0688

TABLE OF CONTENTS

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

INTRODUCTION

We are unquestionably at a critical inflection point where IT strategies are increasingly driven by the need for business agility, accelerated by the improved system agility (in terms of both development and evolution speed) made possible by cloud services, the Internet of Things, big data and SaaS platforms. In the broadest sense, this is the rapid evolution toward what is being called the 'digital enterprise' – where more and more operational and contextual data is gathered, made available through broadly accessible applications, and analyzed to unearth problems and opportunities. A digital enterprise has to be agile so it can respond quickly to what is learned.

Becoming a digital enterprise, and specifically improving business agility, is increasingly seen as an imperative rather than an option, and adaptation toward these objectives is considered a survival issue, imposing pressure and urgency on IT systems. Uber is often cited as an example of how relatively simple new technology – mobile devices and cloud computing – can completely change a big and valuable industry remarkably quickly. These challenges are compounded by the growing complexity of the applications that automate the business; by the growing need for application availability as more of the business is automated for a growing global audience of employees, business partners, customers and the general public; and by the ever-growing importance of security, privacy and regulatory compliance.

This report explores two of the important interrelated IT advances that have been developed to address these demands:

1. The automation of complex application system deployment (as a specific example, the deployment of SAP Business Warehouse on SAP HANA). Similar technology (from Vnomic) is used by VCE as part of the system manufacturing process, and VCE and SAP offer quickly built implementations of SAP on VCE. The SAP/ACI use described here is best documented in published information for these examples.
2. A 'software defined' or programmable system infrastructure, characterized by the ability to define infrastructure policies at the application level, and have the infrastructure automatically implement them in detail – for example, Cisco Application Centric Infrastructure (ACI) and its L4-7 ecosystem. Cisco ACI is differentiated among SDN solutions by the ability to use a high-level abstraction, so the joint example is particularly useful to describe this kind of deployment automation.

451 Research believes that over time this kind of technology will play an increasingly important role in the construction and operation of large computer systems (i.e., clouds) and networks. Interest in this technology is widespread among system architects, and various uses exist. The SAP/Cisco application is one of the most sophisticated that we are aware of, and one of the few that has been publically disclosed so that it can be described. Our aim is that the examples here make the practical reality of this technology much clearer.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

PROBLEM STATEMENT

Business automation is enabled by a set of applications, many of which are intrinsically complex, reflecting the interdependencies, necessary policy-driven constraints and KPIs of the business processes. Deploying these applications is in turn a complex process, during which the application is:

1. **Sized** (capacity constructed to match the anticipated loads),
2. '**Built**' (constructed from component modules and libraries) along with the provisioning of the various operating system and middleware software needed to support the application operation,
3. And finally **'mated'** to the specific system (server, storage and networking) hardware needed to run the application and integrate it with other IT assets.

Building application systems that perform well and are reliable is critically important, so application deployment is a thoughtful and careful process, historically characterized by the collaboration of a set of consultants and experts, each of whom has intimate knowledge of only part of the process – application details, storage details, network details, server details, operating system details, etc. The knowledge of these experts is about the dependencies, constraints and KPIs that define the objective specifically.

The building process starts with the 'outermost applications' that define the functions to be used. Application dependencies determine all the modules and libraries those elements need, which leads to further dependencies that the modules and libraries need, and so forth. Applied iteratively, this leads to all the hardware and software elements required to constitute the entire application system. The constraints and KPIs are additional facts about the desired application system that must be true (e.g., building on a specific family of hardware, or security rules) in addition to providing all the dependent resources that are needed. The use of formal models – a key topic here – serves the dual purpose of formalizing the deployment process and also generating a formal definition that can be used as input by automated tooling or for programmable infrastructure.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

OBJECTIVE

Our primary objective here is to handle application deployment more quickly and life-cycle automation more easily, without diminishing the quality of the process, in order to construct and maintain a correctly sized, highly secure and reliable application. In fact, these solutions can improve the quality of the process and help solve adjunct problems such as compliance documentation.

Solutions Discussed

This report uses as an example the deployment of SAP Business Warehouse on HANA (SAP's in-memory database system), specifically as deployed on Cisco ACI and UCS infrastructure – a solution approach with very tangible benefits – and examines the two distinct solution pieces and their complementary role in this example.

SAP AUTOMATED APPLICATION DEPLOYMENT

SAP provides some of the most sophisticated business applications to many of the largest enterprises in the world, and for many years has studied how application deployment can be automated so that the process can be accelerated and improved. In parallel, for almost a decade Cisco has been developing leading-edge programmable infrastructure – first in the form of its Unified Computer System (UCS) server offering, and more recently in the Application Centric Infrastructure (ACI) programmable infrastructure offering, both developed by Cisco's proven 'spin in' team of entrepreneurs.

In its pursuit of automation, SAP has explored the development of formal descriptions of SAP application systems that capture the dependencies, constraints and KPIs. This is called a 'declarative' model – a description of the end result without regard to how it is built – in contrast to an 'imperative' model, such as a script that specifies the means to achieve the result. A declarative model must then be processed by an orchestration tool to create the detailed assembly instructions that build the system, operating analogously to an optimizing complier that creates specific machine code from a higher-level specification. There are two keys to this: first, capturing complex-enough semantics to describe a system like SAP accurately; second, being able to leverage those semantics to drive application and infrastructure lifecycle operations. The process needs to orchestrate the specification and assembly of all aspects of the application – compute, storage and networking – down to the underpinning hardware.

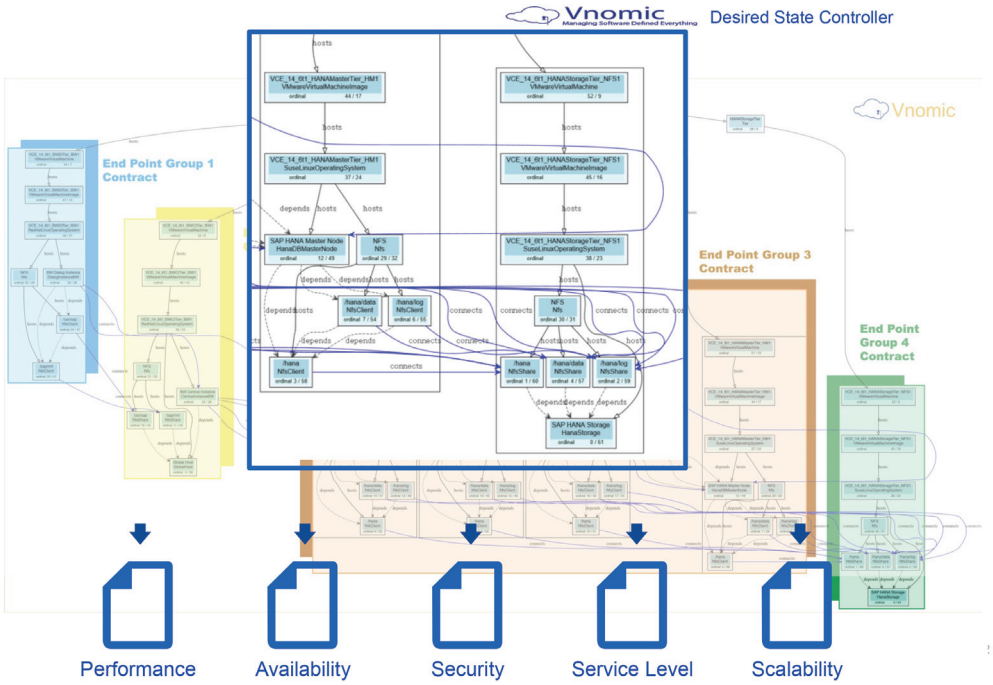
A comprehensive imperative (i.e., scripting) approach to this would quickly grow unwieldy, as the number of options at each level, and the combinatorial explosion of the product of all the options, would create an unmaintainable artifact. Scripts work well for smaller problems, but cannot scale up to this level of complexity.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

Declarative models that are able to encapsulate the complexity of real-world SAP application landscapes would naturally be used broadly, because the resulting process is immediately better (faster, cheaper and with less deployment risk) and because over time the process will improve as more deployment knowledge is captured.

SAP BUSINESS WAREHOUSE ON SAP HANA - 6 TIER LANDSCAPE

Source: Cisco



CISCO PROGRAMMABLE INFRASTRUCTURE

Cisco's work with servers, storage and networks has effectively enabled management of that infrastructure at a higher level of abstraction. The value is seen very clearly with networking, although it is not limited to networking: the network plays a fundamental role in application deployment because it defines the topology of compute and storage elements used by the application, and provides a means of partitioning the elements of the application while providing adequate interconnection capacity and security.

Networks consist of multiple interconnected devices. For the network to operate as desired, the physical topology (what is connected to what) must be right, and then each device must be correctly and consistently configured, in terms of the features selected, the protocols supported, and the specific details of each link and line card. This applies to the networking devices per se (e.g., the switches), and to the devices that provide network services to monitor, manage and secure traffic.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

In ACI, policy is centralized while configuration remains highly distributed and is automated within the infrastructure itself. This is a different approach to infrastructure automation than older SDN approaches, which centralized configuration of individual boxes. ACI's centralized controller and high-level network abstraction in essence brings business awareness to a distributed IT environment and enables the network to implement the business rules embodied in the application design.

At the application level, the specification of a network is much simpler, reflecting the interconnection dependencies inherent in the application, and providing a precise definition by which the network can enforce communication security. For example, the application team may specify the need to connect certain application components on a private network with specific rules governing their intercommunication, and additional specific rules that apply between the private network and other parts of the datacenter network, or with the broader Internet – while prohibiting all communication that is not specified by the application requirements (what is called 'zero trust networking'). The application connectivity requirements definition is a necessary beginning for any method.

Historically, the detailed device configuration was a manual process done on a device-by-device basis, and was a great source of unintended problems. The Cisco ACI infrastructure offering includes a controller that (among other functions) can be given application connectivity requirements and will then automate the configuration of the devices. But this still leaves the question of how these requirements are expressed, and how they are modified as the application changes through its lifecycle.

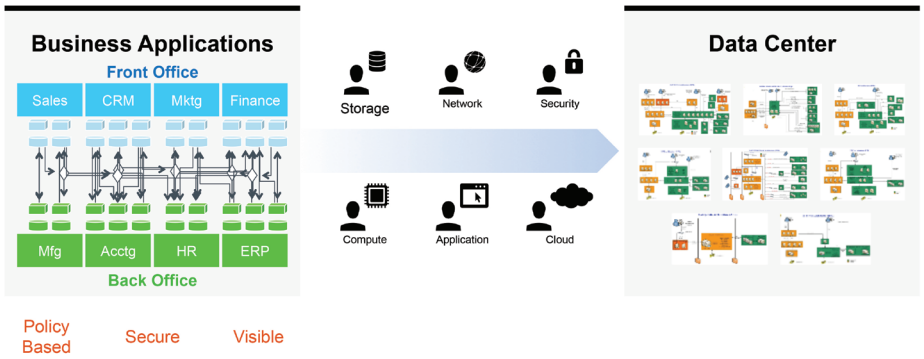
THE SAP AND CISCO APPROACHES COMBINED

The SAP approach is to formalize the application system structure definition down to the underpinning infrastructure, and to use tools that take that definition and create detailed instructions needed to deploy the application. The Cisco programmable infrastructure provides an infrastructure abstraction layer much closer to the application, thereby greatly simplifying the task of mapping the application needs to the specific network and server configuration details required to meet those needs. The two approaches are very synergistic. A deployment process based on declarative models extracts the logical communication rules implied by the application structure's dependencies and constraints (e.g., security and isolation policies) at an appropriate abstraction level so that it can direct ACI to construct the necessary virtual and physical network topologies.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

AUTOMATED APPLICATION GOVERNANCE MODEL

Source: Cisco



Application Delivery on Programmable Infrastructure with Automated Governance

TECHNOLOGY

The technology involved here is abstraction, the most fundamental software technology. Abstraction, in a broad sense, is about making complexity more manageable by concealing the lowest-level details. Disk storage is much easier to use with file abstraction because myriad details are elided (how to deal with a disk read error, for example), while the abstraction of a logical network connection masks the complexity of really getting the data across the network.

ACI provides a high-level abstraction of infrastructure that is much closer to how an application designer thinks of the network and conceals myriad details underneath. For example, ACI defines 'endpoints' and 'endpoint groups' that map to application interfaces and sets of interfaces with common policies, which are easier to deal with than ports and IP ranges. ACI's abstractions reduce the semantic distance and, thus, the translation effort from the application domain to the network and infrastructure domains, reducing the risk of mistranslation.

DECLARATIVE MODELING

Declarative modeling of SAP's application software (done by SAP experts working in conjunction with customers) creates a high-level abstraction of the assembly process. Here the value of abstraction is just as great as with ACI, but perhaps subtler. Compared with the manual deployment process that it replaces, use of a declarative model produces the same result: a set of construction instructions that, when followed accurately, deploy the application.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

If the result is the same, why is the declarative model better than just creating the complete set of instructions manually? The reason is that relatively simple changes in the deployment specification (the dependencies and constraints) can cause large changes in the full instruction sequence. More to the point, a new constraint may break those instructions in ways that aren't clear until you go through the whole process and discover that constraint is no longer satisfied, and then have to go back through the process looking for and correcting the issues.

By contrast, the changes in the declarative model are clear and explicit, and we leave it to the orchestration tool to work out the details (just as we leave it to an optimizing complier to create the specific machine code). Using a formal definition model also explicitly captures the complete definition of the deployment (exactly what drove the decisions made), and because the process is automated, it is repeatable – the same definition will result in the same deployment instructions each time, which is not at all true of a manual process.

Creating and optimizing declarative models is both art and science. Creating a good implementation solution from dependencies and constraints is a nontrivial optimization problem, but the greater challenge is the art and craft of creating the model itself. There is nothing obvious or simple about abstraction – it is invention. There are an innumerable ways to build an application deployment abstraction, and why one is better or worse only becomes apparent as you try to use it.

At some point in the future, the 'right way' to build a deployment model will be 'obvious,' and will be taught as such to computer science students. But getting to that clarity is hard and persistent work that depends on believing in the fundamental idea that formal modeling and automated implementation of the detailed instructions can eventually be much better than the established manual process. And it requires vision that drives the development, and then patience and persistence while the process is understood and refined, necessarily with some fits, starts and even restarts.

IMPLEMENTING THE STRATEGY

SAP and Cisco have jointly innovated this declarative modeling approach with Vnomic, a Silicon Valley technology company in the SAP Startup Focus Program and a Cisco Partner. Vnomic has been working with declarative models for years and has deep subject-matter expertise on applying the 'desired state' concept to declarative modeling. Vnomic was founded in 2009, and has developed the Vnomic Declarative Application Delivery and Governance Platform, which processes dependencies, constraints and state to deploy complex application landscapes like SAP's.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

The founder and CTO of Vnomic is a coeditor of OASIS TOSCA (Topology and Orchestration Specification for Cloud Applications), which is a vendor-neutral standard for declarative modeling of applications running in clouds and converged infrastructure. TOSCA-like semantics are being added to OpenStack through the HOT template specification, being developed as part of the OpenStack Heat project. In networking, YANG declarative models are another important topic, especially within OpenDaylight.

PROGRAMMABLE INFRASTRUCTURE

With ACI, Cisco has formalized and automated the network deployment process that has been learned based on decades of experience implementing customers' network systems and translating customers' network requirements into implementations. The application-level network abstraction was relatively well known – it's what people draw on whiteboards to explain a network implementation. What is differentiated about ACI compared with other SDN approaches is that Cisco chose to automate network deployment top-down from the application level from a set of declarative application connectivity abstractions, whereas most SDN competitors chose to automate network deployment much closer to the physical level, thereby providing (useful) automation but still requiring network expertise to translate between the application abstraction and the network automation abstraction – leaving open the risk of mistranslation. The lower-level abstractions may prove useful for some of the most demanding network configurations if a network admin or specific automation software is able to create a better configuration than the ACI controller.

Another example of the benefit of a high-level network abstraction is the ability to extract from an application definition a description of the network capacity, and use that to generate a synthetic load that can be used to drive a test load on the physical network.

Ignoring many of the important details, here's how automated application deployment works:

1. An initial model of both application and infrastructure dependencies and constraints is extracted from a variety of subject-matter experts. For the example in this whitepaper, SAP provided expertise on BW and HANA, while Cisco provided expertise on ACI and UCS.
2. An orchestration tool (in this case, from Vnomic) processes the model to produce a complete set of scripts:
    a. The model is 'flattened' into an interconnected list of all the required components (a 'directed graph'), starting with the desired application functions and working down to the hardware that ultimately implements the functions.
    b. The orchestration tool discovers the current 'as is' state of the system under management, including hardware and software, and computes the differences between the as-is state and the desired state. This list of differences might be very long, for a 'greenfield' deployment, or very short, for a minor adjustment.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

   c. The orchestration tool develops a set of deployment instructions to remediate all differences between desired state and as-is state, starting with the hardware and working up, making important heuristic decisions (like capacity allocation) along the way, as well as making sure that all specified constraints are met.

3. The resulting implementation is manually inspected and tested, a well-understood process because the same thing happens in the traditional manual deployment process:

   a. Problems are found and diagnosed.

   b. The model is changed and refined to reflect the new learning.

   c. The orchestration tool may be refined as well.

4. Steps 2 and 3 are iterated until a suitable implementation is constructed.

The first time through the process is similar to the manual deployment process, and takes more or less the same time – but with some very important improvements:

- The process is well defined and repeatable.
- The reason for all choices is understood and expressed in the model or the operation of the orchestration tool.
- The detailed deployment process is fully detailed – you know exactly what happened and why.

ROLLING OUT THE INNOVATION

SAP and Cisco are engaging with mutual customers, both large enterprises and cloud service providers, to use this new deployment methodology, combining declarative modeling of applications with programmable infrastructure. The customer benefits are obvious: reduced cost, deployment risk and time-to-value, with improved security, governance and auditability.

Many SAP customers will be able to use pre-developed declarative models of popular SAP applications. SAP BW on HANA was selected as the first declarative model to roll out because BW is easy to use out-of-the-box. But customers have often modified other SAP applications, such as SAP Business Suite, to drive business system differentiation. The declarative approach is now being extended to S/4HANA, the newest version of SAP's suite.

Many SAP customers will have some new requirements in their deployment process (different choices in system hardware, or different operating system and middleware details, for example), so each new deployment may bring some new learning and require multiple iterations through this process. But the assumption is that much of the learning can be reused so that in the end, this makes a dramatic difference in the length of the deployment process and contributes greatly to IT and business agility because most of the required dependencies and constraints are already detailed.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

The value doesn't stop with application deployment. It is possible to use these same mechanisms during the entire lifecycle, after an application has been deployed, either to discover (manual) post-deployment modifications that were incorrect ('configuration drift') so they can be reversed, or to drive incremental changes to the application that reflect fixes and additions to the application or underlying dependent components.

The desired-state approach introspects an operating application system and builds a detailed description of the as-is structure (the components and how they interconnect). This as-is description is compared with the desired state, which could be either the 'as originally specified' description or a description of how it should be modified today to incorporate fixes and updates. The comparison is used to generate a script to remediate the state or to move the state forward to a current definition. This is useful to ensure compliance and to demonstrate correct operations for auditing, and even as a means of judging whether the changes (often the unintended consequences of various maintenance activities) are severe enough to demand attention.

'PROJECT BENJAMIN'

In the fall of 2015, SAP and Cisco described 'Project Benjamin,' a large (108-node) UCS Cluster that has been used to validate SAP configurations that include SAP HANA, HANA Vora (an attached 'data lake' system built on Hadoop and Spark), SAP Business Warehouse, and S/4HANA.

When it was disclosed, the Project Benjamin cluster had 33TB of memory (divided between HANA and HANA Vora); a total of 2,880 cores; and 3PB of disk (3,000 TB) when fully populated. SAP configurations can be specified, and declarative models are used to automatically build and deploy the configuration, as well as to drive the ACI infrastructure, using tools developed by Vnomic.

At the time of the disclosure, Project Benjamin was being used to validate a diversity of configurations and enable detailed performance characterization (e.g., for a potential customer proof-of-concept). It seemed likely at the time that in the future the technology could be used to design, construct and test specific customer systems when ordered.

COMPETITIVE APPROACHES

The biggest competition here is from the status quo. The existing methods do work (manual application deployment, manual network configuration and administration). The new approach introduces new technology that is unlikely to be known by the experts performing the work today. In such circumstances, it always helps to have some higher-level driver. In today's world, the need for greater business agility and the move to the digital enterprise provide that motivation.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.

In the case of application deployment automation, the biggest competitors are application 'blueprints' – definitions of specific hardware and software configurations that have been certified as able to deliver some defined level of application performance. Use of these blueprints constrains the resulting systems greatly (it disallows the flexibility normally associated with configuring these business-critical assets), and is less precise in terms of the rules defining the configuration (the configuration is defined; exactly how that definition was created is usually not).

When it comes to networking in the broadest sense (constructing the infrastructure topology for all compute and storage endpoints), no other SDN solution combines a high-level definition of the application interdependencies and requirements with the detailed configuration of the physical networking, while retaining the richness of existing network functionality. Most other SDN approaches that act on the physical network do so at a much lower level of detail. Those that provide application-level abstractions either use an overlay/underlay model that largely ignores the configuration and maintenance of the underlay transport network, or use new network technology (like OpenFlow) that can't match existing network implementations in terms of functional richness.

Lower-level abstractions (compared with ACI) could prove useful for very demanding network applications if the network admin team (or application-specific low-level automation) can do a better job than the ACI Controller. ACI is differentiated in delivering a high-level abstraction that drives rich network system infrastructure.

CONCLUSION

Companies are rapidly digitizing their businesses, and they are using IoT, big data, cloud technologies and SaaS as foundational solutions. But doing so raises a host of complexities – bringing together different teams, technologies, deployment options and processes.

In short, the first step in the digitalization journey is the automation of application and infrastructure delivery and governance. With the level of complexity that is involved, the most effective approach is a desired-state, policy-driven solution for delivery and governance of both applications and infrastructure.

© 2015 451 RESEARCH, LLC AND/OR ITS AFFILIATES. ALL RIGHTS RESERVED.