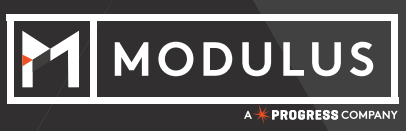
SOFTWARE-DEFINED ARCHITECTURE

TAKING WEB SCALE TO THE NEXT LEVEL IN THE CLOUD

A WHITEPAPER BY



SUMMARY

Software-Defined Architecture (SDA) is a new style of software architecture that advances already powerful "Web Scale" applications. Web Scale refers to the capabilities that large cloud services firms such as Amazon, Netflix, and Facebook have to scale up software to huge audiences while remaining agile enough to adapt rapidly. Coming in the wake of software-defined networking and software-defined storage, SDA enables a new, more flexible Web Scale by introducing a layer of virtualization between software and its consumers. The result is the power to connect a vast array of consumers to complex backend systems while simultaneously masking that complexity. SDA makes it possible to change the underling software easily, without affecting the consumer. However, getting SDA to deliver the next generation of Web scale is as much about technology as it is about the people and processes who manage it. This paper looks at what it takes to be successful with SDA and the roles that software platforms play in ensuring optimal outcomes in Web Scale computing.

INTRODUCTION

Technology, businesses, and consumers are engaged in an endless, synergistic dance. Technology influences the way people do business, but it also adapts to changing user expectations. For instance, the Web changed the way people shopped for cars. Now, the consumer expectation that car dealers will provide information online has led to the creation of wholly new car-buying apps. We are now seeing a similar evolution with the advent of Software-Defined Architecture (SDA).

SDA is the new incarnation of "Web Scale" computing. Both Web Scale and SDA refer to technologies that enable businesses to function as digital entities, engaging deeply with potentially very large groups of customers and third parties with a high degree of flexibility. SDA takes Web Scale further, establishing a virtualization layer between software applications and users – both human and machine. The boundary conceals the internal details of the underlying software. SDA is a boon to agility, making it possible to change or replace the implementation quickly and without affecting the consumer. This paper explores how SDA works and how it takes Web Scale enterprises to new heights of agility and scalability.

WEB SCALE COMPUTING: THE ORIGINAL BREAKTHROUGH

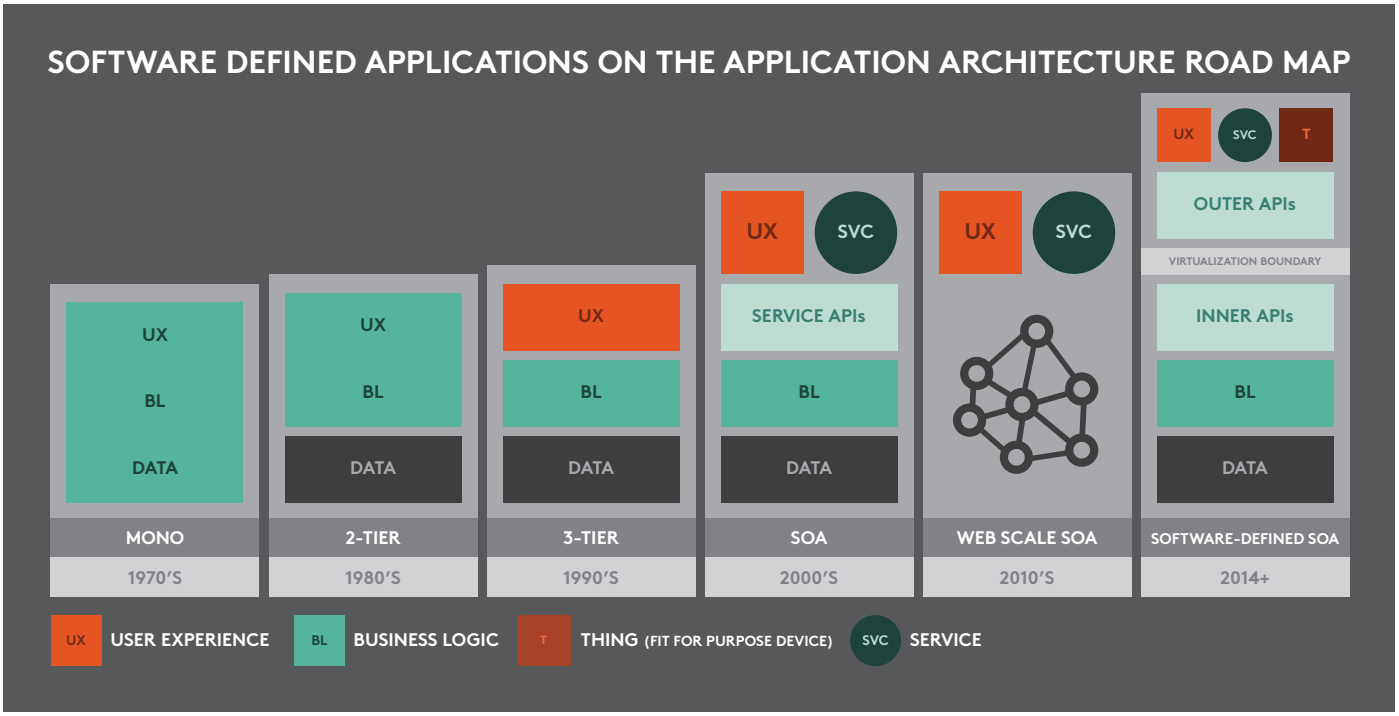


Figure 1 - Web scale as an evolution of Service-Oriented Architecture (SOA) - (Source: Gartner)

Web Scale is a term that describes where a lot of enterprise IT is headed. Web Scale means handling system growth that's non-linear, with explosive changes in demand and load. Web Scale systems can handle rapid growth efficiently. They tend not to have architectural bottlenecks that require re-tooling as requirements inevitably shift. Gartner characterizes Web Scale as a natural evolution of software architecture, as depicted in Figure 1. As the 3-tier architecture of the 1990 gave way to Service-Oriented Architecture (SOA), architectures started to divide the user experience increasingly from the business logic through web services. Web Scale further disconnects User Experience (UX) and services from underlying software and data.

Examples of Web Scale enterprise include Facebook, NetFlix, Uber, and Google. But, Web Scale is not just about size. Web Scale is about organizations and processes. It's a state of mind. Business of all sizes can create Web

Scale applications. It's about meeting consumer expectations. Consumers today simply expect the brands they patronize to be always on, always mobile, always up to date. The technologies that gave companies like Google a way to scale to massive compute environments are now available to all enterprises. There is an assumption that personalized data will be available on demand, with all the access to sensitive back systems that assumption entails.

TAKING WEB SCALE TO THE NEXT LEVEL WITH SDA

Web Scale is not easy. One cannot push a button and make it happen. It takes a technological and organizational rethinking to get there. Its success rides on many factor, including use of "industrial scale" infrastructure. Typically based on the cloud, industrial scale architecture means being able to add massive amounts of compute, network, and storage capacity on demand. Web scale needs to be built on a web-oriented architecture. Use of open standards, HTTP, and TCP-IP are essentially required for achieving Web scale. The management of Web scale applications must be programmable, with a high level of administrative automation. The development process should be based on agile methodologies within a collaborative, learning organization.

SDA introduces a structure that simplifies the implementation of Web Scale and enables it to achieve new heights of agility and scalability. The emergence of SDA is a new step in a broad IT trend. As systems migrate to the cloud, architects are increasingly using software to make connections that used to be specific to physical equipment or distinct software and data sources. Examples include Software-defined Networking (SDN), Software-defined Storage (SDS), and the Software-Defined Datacenter (SDDC). SDN virtualizes connections between systems and networks, replacing physical connections with software. SDS does the same thing for storage capacity.

FLEXIBILITY THROUGH LOOSE COUPLING

SDA takes Web Scale to the next level of agility and scalability by abstracting the internal from the external. The architecture becomes a great deal more flexible by loosely coupling of architectural elements to create a nearly total division between inner systems and external users. In this scenario, the user can be human, software, or a "thing" like a sensor or a vehicle. As Figure 2 shows, two layers of Application Programming Interfaces (APIs) form the "virtualization boundary" between the inside and outside worlds.

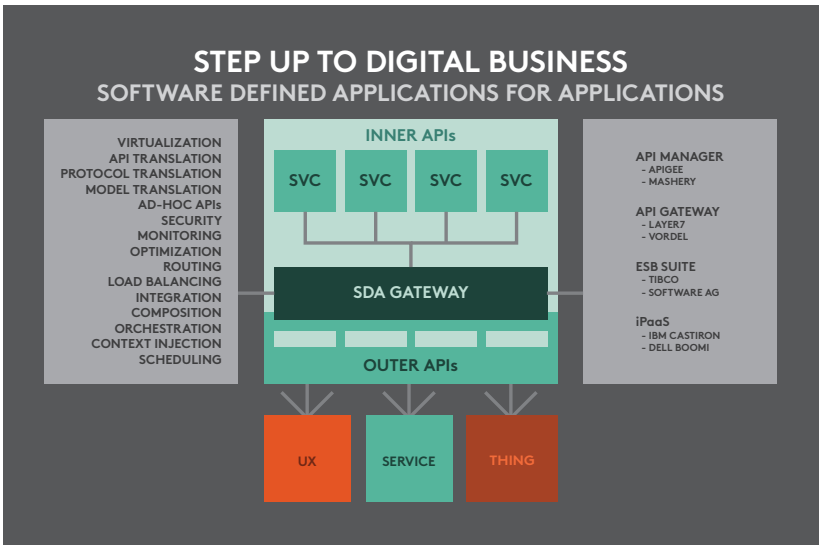


Figure 2 – SDA and the separation between inner and outer APIs, including the technological capabilities required to make it work. (Source: Gartner)

SDA starts with Services, such as SOAP web services or REST-based APIs, but they are not consumed directly. Instead, as Figure 2 shows, SDA creates a layer of virtualization between software producers and consumers. The inner APIs describe the organization of the underlying software and data using open standards. The outer APIs enable simple, safe consumption of the inner APIs. It is then possible to change or replace a back-end software component without affecting anything on the consumption side of the architecture. Being able to connect, disconnect, and reconnect applications through standards-based APIs removes much of the friction that traditionally slowed down scaling and modifications of application architectures.

THE SDA GATEWAY

The virtualization boundary between internal and external architectural elements needs to be mediated by a type of software (or a collection of applications) known as an SDA Gateway. The gateway takes care of virtualizing the Inner APIs. As Figure 2 illustrates, the gateway is tasked with many critical functions to ensure that the SDA can function. These include the translation of APIs, protocols, and models. For instance, if an external service uses JSON over REST to call on an internal API that's exposing an underlying SOAP service, the gateway needs to handle the translation back and forth between these two protocols.

The level of open connectivity enabled by SDA is potentially chaotic and insecure. Administrators on both sides needs to be aware of the status of the boundary and the performance of components that are connected through it. To mitigate the risks of unreliable or insecure APIs, the SDA Gateway performs many functions. The following is a partial list of SDA Gateway features, though few SDA Gateway products actually do all of these with equal effectiveness:

VIRTUALIZATION

The inner/outer API combination of the SDA construct creates a virtual end point for a service. Even if a service is buried deep behind a firewall, it can be surfaced as a virtual endpoint in the clouding using a REST-based API. The gateway manages this process and facilitates connectivity between the inner and outer APIs that make it possible.

TRANSLATION OF PROTOCOLS AND MODELS

As API calls go in and out of the SDA, the gateway often has to translate between the various languages, message protocols, and data models used by the SDA components. For instance, a web UX might use REST and JSON to call on a REST API to get data from a SOAP-based web service behind the firewall. The gateway translates the messages back and forth so the API call will produce the desired result. The gateway enables agility in this case because the UX developer does not have to understand how to make a SOAP call in order to get data from a SOAP resource. This is saves a lot of time. And, not all UX developers are even familiar with SOAP.

SECURITY

Exposing open APIs to the world is inherently insecure, even if it helps business grow. The SDA Gateway needs to authorize and authenticate API users, both human and machine. It has to control access and filter content. In some cases, the gateway may need to manage the OAuth protocol or create security assertions to clear secondary and tertiary API calls from a single requester.

MONITORING

The gateway monitors APIs to ensure that they are functioning as intended. This might involve defining and enforcing an agreed-upon service level, alerting administrators if a service is slow or failed, or redirecting API calls to a failover API instance.

OPTIMIZING, ROUTING, AND LOAD BALANCING

An SDA Gateway is responsible for managing the flow of API calls and routing intra-service messages that occur between SDA components. This may require load balancing to maintain service levels.

INTEGRATION

The SDA Gateway functions as an application integration mechanism, though it is loosely coupled. There are no proprietary connectors or hard-coded integrations.

COMPOSITION AND ORCHESTRATION

Some SDA Gateways are able to orchestrate workflows that consist of sequential API calls.

Figure 3 shows an example of orchestration. In this case, a retailer wants to create an automatic sale on winter boots any time the temperature drops to a certain level outside a store location. To do this, a gateway can orchestrate a series of tasks that call on a weather sensing "Thing" that is connected to the gateway. The orchestration could set up an event listener that responded to a cold "event" by prompting the retailer's ecommerce server and product database web services to publish the boot sale during the cold snap and take it down after the rain stopped. To make this work, the gateway has to transform between SOAP and REST messages while doing the orchestration.

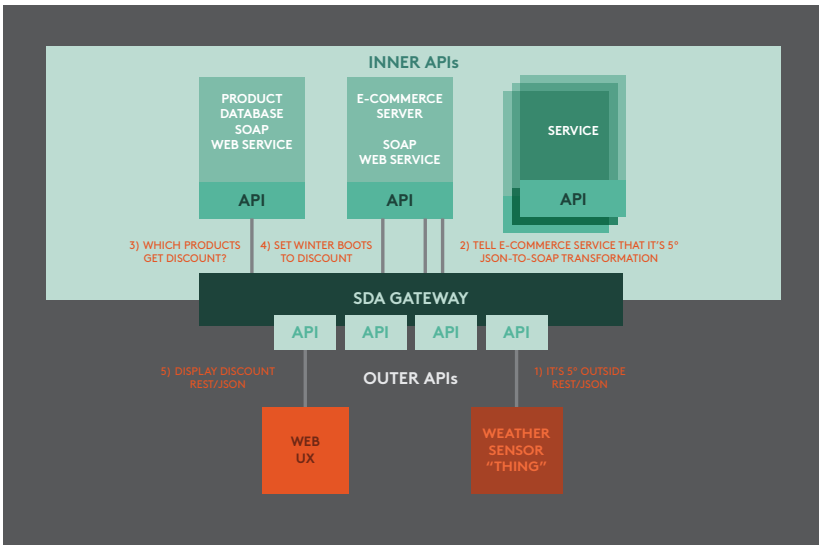


Figure 3 – Example of orchestration of SDA components using APIs and an SDA gateway.

The scenario depicted in Figure 3 shows how flexible and scalable the SDA gateway approach can be. The retailer could easily add new UXs and things to the outer API or more services on the internal APIs. Switching the orchestration is relatively simple because the entire process is done with open standards-based APIs. No application integration technology or proprietary adapters are required. Because the connections between the components are logical and software-based, adding additional capacity is also simple. The whole setup can scale quickly.

THE ROLE OF SOFTWARE PLATFORMS IN SDA

It's important to bear in mind that the "S" in SDA represents both the connections between applications as well as the applications themselves. The success of SDA depends partly on the gateway, but the actual software itself is critical for achieving the kind of agility and scalability that the architecture promises. For this reason, it's necessary to understand the how the underlying software platforms make the SDA possible.

Software development and deployment in the SDA presents an incomplete, evolving picture. However, a number or realities and best practices are emerging. The most basic question that needs to be asked and answered about a software platform in SDA is, "Does it further the agility and scalability inherent in the architecture?"

SDA AND AGILE METHODOLOGIES

Some perspective can help. SDA has arisen in parallel with a significant change in the way software is developed and deployed. In recent years, the traditional, linear "waterfall" method of software development has been supplanted in many cases by newer agile methodologies. Agile software development methodologies involve writing software code in short bursts, known as "sprints," that might last a week or two. Instead of gathering all requirements for an application, producing a lengthy, complete design document, and then spending a long period of time coding an entire application, the agile approach breaks the process up into smaller steps. One advantage of the agile process is that it can show stakeholders features they have requested in a rapid timeframe so it is possible to change the design of the application as the development process goes forward. Agile tends to be more flexible than waterfall.

Many discussions of SDA assume that agile methodologies are in use, at least in the "outer" side of the architecture. In addition, it is also often assumed that related processes and technologies are in use together with agile. These include "Continuous Integration" (CI) and "DevOps." CI makes it possible to add newly coded features or updates to live software in production without having to reinstall the entire application.

DevOps refers to the merging of software development and IT operations teams. When an IT department uses DevOps, it is uniting two previously separate groups that were responsible for developing and deploying software. The result, if DevOps is done right, is to speed up the development and deployment of code. When CI and DevOps are performed together, the iterative pace of software change can accelerate greatly.

SDA platforms should ideally support these capabilities. There is a natural synergy. SDA gives IT managers the ability to add, modify, and remove integrations between systems and data with relative ease. If the underlying software can also be modified quickly, the net effect is greater agility. Managers of SDA would prefer not to hear, "You can integrate this application in 18 months when we're finished with the waterfall process." The integration requirements would likely have changed by then, in any event. They want to hear, "You can add our new build next week."

WHAT THE SOFTWARE PLATFORM IN SDA DOES

The very definition of a software platform is changing in this current moment. It used to be well-understood that software was developed in a dedicated development environment, tested, and then deployed on a separate production server. This is less the case today, due to the fluidity with which agile processes work. For example, with Node.js as hosted by Modulus and others, the barriers between software development and production deployment are nearly nonexistent. Code can be developed, deployed, and scaled seamlessly through a single interface on a single platform.

From today's vantage point, the software platform for SDA is responsible for the creation and deployment of some of the software that functions in the SDA. Not all. The very nature of SDA is heterogeneous. There are going to be all sorts of software assets at work in an SDA environment. But, where the IT manager can control the software platform, the following criteria should be taken into consideration:

The Software Platform is Where You Write and Deploy Code into the SDA – The software platform should be able to manage the deployment of code efficiently from development to production.

You can Host APIs on the Software Platform – APIs, that key enabling technology of the API can be hosted on the software platform. For example, some enterprises host their APIs on Modulus and other platforms like it. Placing an API on a cloud-based software platform makes it possible to scale up API instances in response to changes in growth in load.

You Manage Your Code in the SDA through the Software Platform – The software platform should give you the ability to manage multiple versions of your application in a coherent, seamless way. With the rapid pace of iteration, a good software platform keeps administrative overhead to a minimum for managing the coding workflow.

The Platform Gives you Usage Metrics – Administrators should be able to get metrics on application and API consumption and performance from the software platform.

The Platform is where the Team Collaborates on the Code in the SDA – A good software platform is designed for team collaboration. This can be for development only or for DevOps. The platform should enable communication and cooperation between different stakeholders, including line of business managers if they need to be included in the software development process.

The software platform overlaps to some extent with the gateway. Some gateways provide some of the same functions as the software platform. The reverse is also true, but they are distinct technologies. In the best case, the software platform and gateway are configured to work together. Together, they form the basis for a strong SDA.

CONCLUSION

Web Scale is the path that today's businesses need to take to become digital enterprises. SDA is the means to get there, at the new, high level of agility and scalability required for success. SDA works when there is a fusion of architecture and technology, as well as people and process. The architecture calls for a virtualization layer between software applications and human and machine users. Instead of integrating applications with each other directly, or through proprietary connectors, the SDA relies on open standards-based APIs. An SDA Gateway realizes the virtualization layer, giving enabling a high degree of flexibility to connect, disconnect, and reconnect many different types of software and IT assets, such as data sources and storage. SDA translates into agility, making it possible to change or replace the implementation quickly and without affecting the consumer.

Software platforms comprise the other technological element of a successful SDA. While the Gateway facilitates the handling of the API calls and the overall structure of the architecture, the software platform makes possible the kind of underlying software components that bring out the scale and flexibility potential of the SDA. The software platform, which is likely to be employed for the execution of an agile development methodology, gives developers and IT operations managers the ability to create, deploy software and APIs that function within the SDA. The platform is the tool they use so SDA can take Web Scale computing to its greatest potential for agility and scale.