It's No Longer Enough to Simply Be Agile

Johnny D. Morgan, PhD
General Dynamics Information Technology; johnny.morgan@gdit.com

ABSTRACT

A tremendous amount of literature has been published about the merits of agile development practices. But in today's environment, agile development practices are quickly being supplemented with major technology breakthroughs that enhance software quality, improve enterprise performance and provide business resiliency. This paper describes three major breakthroughs; services-based architectures, cloud computing, and DevOps practices. A brief overview of each technology is discussed and how the three technologies working together provide enterprise value. The paper concludes with a discussion on the skills and talents required to implement these technologies.

Key Words: agile, cloud, cultural shifts, development, DevOps, elastic computing, information technology, IT skills, operations, organizational structures, pipelines, DevSecOps, software development, software services, testing

This paper is based on empirical observations, current literature, and engineering and project management experiences.

INTRODUCTION

Since the Manifesto for Agile Software Development was published in 2001, a tremendous amount of literature has been published that documents many agile software development frameworks such as Scrum, Kanban, and Extreme Programming. These software development frameworks have similar characteristics. All potential product features are placed into a feature backlog and prioritized for development, with the highest value features being developed first. Agile teams execute time-boxed work periods, typically called sprints, to develop these features. These sprints typically range from two to four weeks. Each agile team is composed of a small group of multi-disciplined developers that are focused on the continual delivery of valuable software. Within each team there is a Product Owner who is the voice of the customer, prioritizes the feature backlog, and accepts the delivery of each feature. There is also a person that facilitates team meetings and eliminates blocking issues that are inhibiting team progress. Within the Scrum methodology, this person is called the Scrum Master. There is a regular cadence of meetings within each sprint. The sprint commences with a Sprint Kickoff Meeting that determines what features the team will develop within the sprint. There are Daily Standup Meetings where the team reviews progress, identifies any blocking issues, and assigns work to be perform next. A Sprint Completion Meeting is held at the end of the sprint to review with customers and users outside of the agile team the actual delivery of the features that were developed during the sprint. Within the agile team, a Sprint Retrospective Meeting is also held where the team can identify and address potential improvements to team performance.

More recently, frameworks have been developed to scale agile development practices from a single team to multiple agile teams working in parallel to deliver larger systems. The most popular framework is the Scaled Agile Framework (SAFe) which adds additional team resources and process elements to synchronize the alignment, collaboration, development, and integration mechanisms of multiple agile teams to deliver large, more complex systems (Leffingwell and others 2017).

In today's environment, agile development practices are quickly being supplemented with major technology breakthroughs that enhance software quality, improve enterprise performance, and provide business resiliency. This paper describes three major breakthroughs; services-based architectures, cloud computing, and DevOps practices.

SERVICES-BASED ARCHITECTURES

A services-based architecture, or microservices architecture, "is an architectural approach to developing a single application as a suite of small services, each running it's own process and communicating over a lightweight mechanism, often an HTTP (HyperText) resource API (Application Programming Interface)" (Fowler 2014).

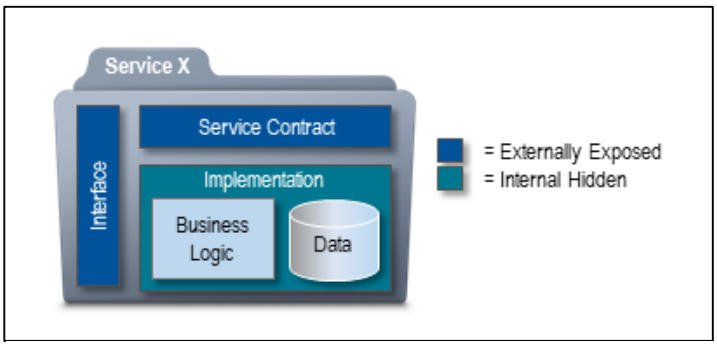

Figure 1: Service Components (Enterprise SOA: Service-Oriented Architecture Best Practices – 2005)

As shown in Figure 1, each service is:

> "a software component of distinct functional meaning that typically encapsulates a high-level business concept. It consists of several parts…The service contract provides an informal specification of the purpose, functionality, constraints, and usage of the service… The functionality of the service is exposed by the service interface to clients that are connected to the service using the network…The service implementation physically provides the required business logic and appropriate data. It is the technical realization that fulfills the service contract. The service implementation consists of one or more artifacts such as programs, configuration data, and databases…The business logic that is encapsulated by the service is part of the implementation. It is made available through the service interface." (Krafzig, Bank, and Slama 2005)

Furthermore, a service must have the following attributes:

- The service is *self-contained* and performs a distinct business or technical function.
- The service is *loosely coupled*, meaning that it has an explicit contract (interface) independent of the technology of the invoking service consumer.
- The service is *transparent*, meaning that the specific location of the service is immaterial to the service consumer, with binding taking place at deployment or runtime.

- The service is *interoperable*, meaning that service interaction can be supported over a wide variety of platforms due to usage of compatible, industry standard communication protocols.
- The service is *composable*, meaning that it can be aggregated as part of a service at a higher level of granularity. (Bieberstein and others 2008)

Figure 2 provides an example of a video subscription site that is a composition of seven services and allows users to log onto the site, receive recommendations on videos to view, and allows the user to search, read reviews, select, and watch videos. This is a very simplistic example when compared to the Netflix video streaming service which provides over 114 million hours of streaming video each day to more than 117 million streaming memberships in over 190 countries and is implemented using 500 software services. (Anonymous2018)

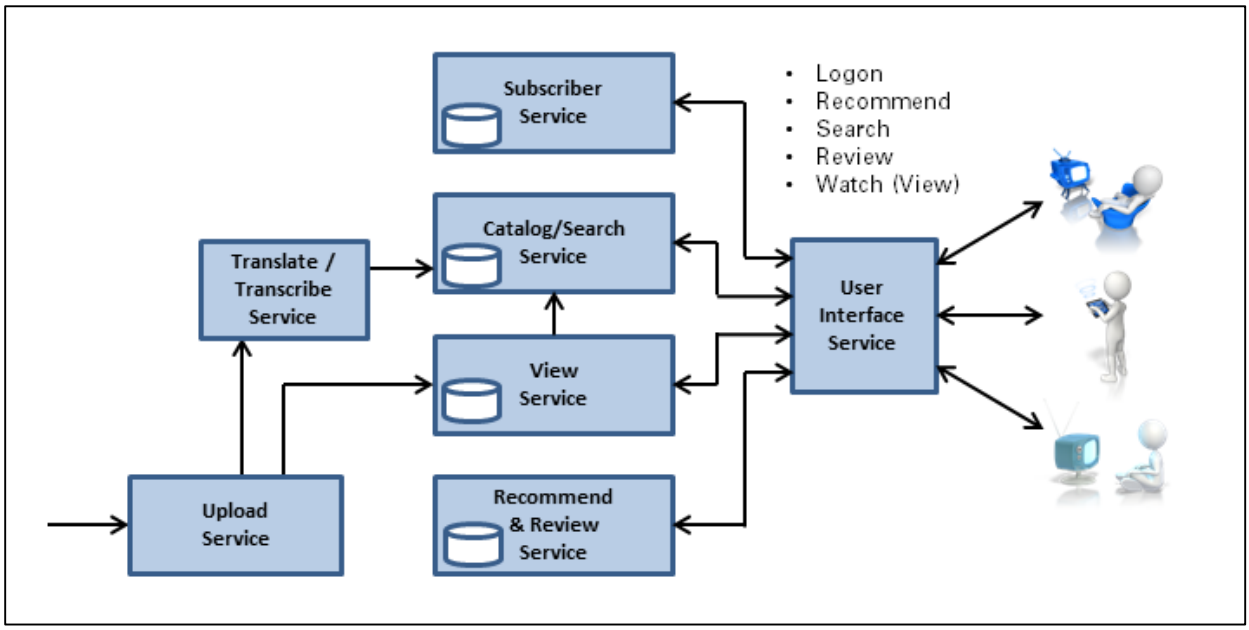

Figure 2: Video Subscription Site - Service Composition Example

A services-based architecture provides many benefits. It allows for faster and small deployments as each service can be independently developed and delivered without affecting other services. Each service can follow a single responsibility principle for the service, thus each service can implement the right framework, and use the appropriate technical skills, tools, and development language required for that specific service. It promotes the rapid insertion of new technologies and functionality. Each service can scale independently to meet the performance demands of that service, thus scalability is not required at the full application level, and it promotes greater resiliency, availability, and fault isolation as the loss of a single service may not result in a total loss of the entire application or site. (Tonse 2014)

CLOUD COMPUTING

The National Institute of Standards and Technology (NIST) defines cloud computing as having five essential characteristics:

- On-Demand Self-Service: A consumer can unilaterally provision computing capabilities without requiring human interaction.
- Rapid-Elasticity: Can be quickly provisioned and released, automatically… commensurate with demand.
- Broad Network Access: Accessible through standard mechanisms to promote use by heterogeneous thin or thick client platforms.
- Resource Pooling: Provider's resources are pooled to service multiple customers using a multi-tenant model.
- Measured Service: Automatically control and optimize resources by leveraging a metering capability. (Mell and Grance 2011)

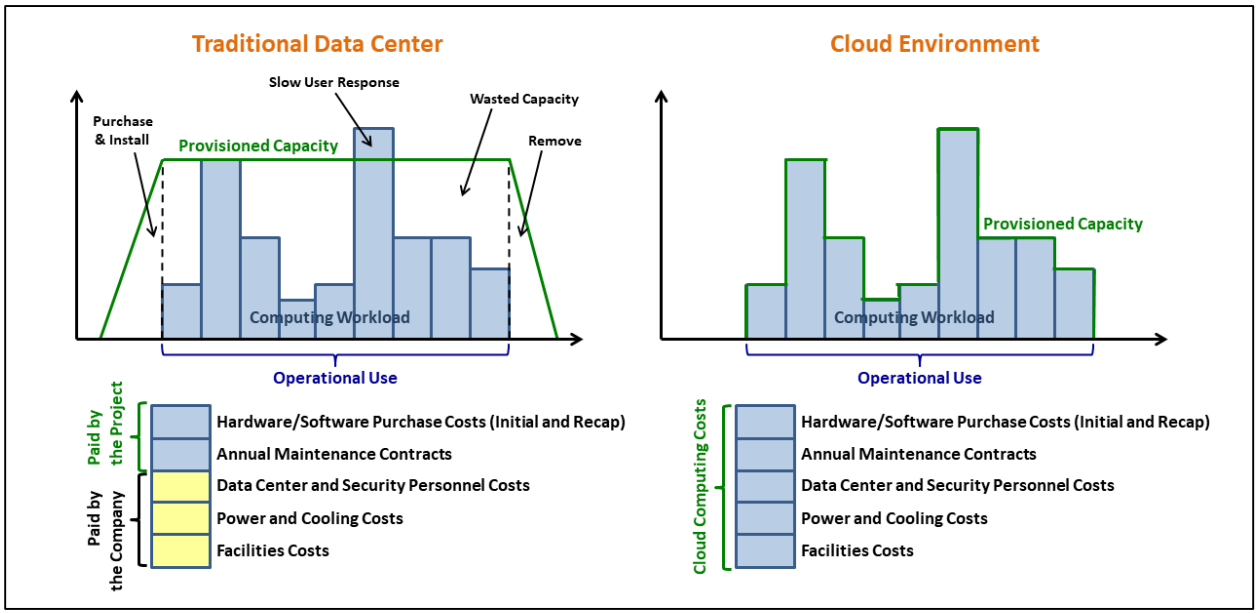

Figure 3: Comparing Traditional Data Center Computing and Cloud Computing

Figure 3 provides a graphical comparison of traditional data center computing to cloud computing. When installing a system in the data center, the project manager must predict the peak demand the system is expected to encounter and then purchase and install enough computing capacity to respond to this demand. The project manager must account in his schedule the time required to define, purchase, ship, receive, install, and configure these computing resources. In most cases, the computing workload demand will be less than the provisioned capacity, thus computing resources are wasted, but there is the possibility that computing workload demand could exceed provisioned capability thus some users may experience slow user response or may not even be able to access the system.

With cloud computing, the rapid elasticity of the cloud allows the system to add or reduce computing capacity in response to user demand. Furthermore, the system can be quickly defined and configured thus allowing for rapid provisioning and deprovisioning of resources without schedule delays. Cloud computing services incorporate technologies that promote the proactive distribution of computing workload across assets located in multiple locations therefore providing high availability and disaster recovery mechanisms with little additional costs.

As cloud computing is a measured service, the project manager must be very cost conscious when using the cloud to ensure that computing capability is shutdown when not in use. Cloud computing costs can also appear higher to the project manager but that is generally due to how some commercial companies and Government agencies allocate costs. In the traditional data center model, the project manager may only be required to include the capital expenditure costs of hardware and annual software licensing and maintenance costs in their budget. This is because the data center operator and personnel security labor, power, cooling, and facility costs are separately paid by other areas of the company, whereas all of these costs are included in the cloud computing costs.

DEVOPS PRACTICES

IT organizations are responsible for two major goals which must be pursued simultaneously:

- Respond to the rapidly changing competitive landscape
- Provide stable, reliable, and secure service to the customer

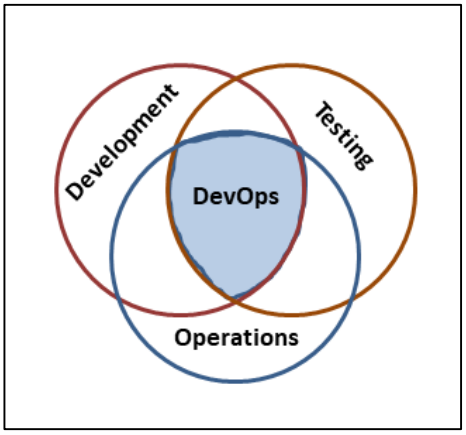

Figure 4: DevOps is the Successful Integration of Three Disciplines

With DevOps, small teams of developers independently implement their features, validate the correctness in a production-like environment and have their code deployed into production quickly, safely, and securely. As shown in Figure 4, DevOps brings together development, testing, and operations all on the same team, a team capable of playing at levels. (Kim and others 2016). More recently, cybersecurity features are also being incorporated to become DevSecOps.

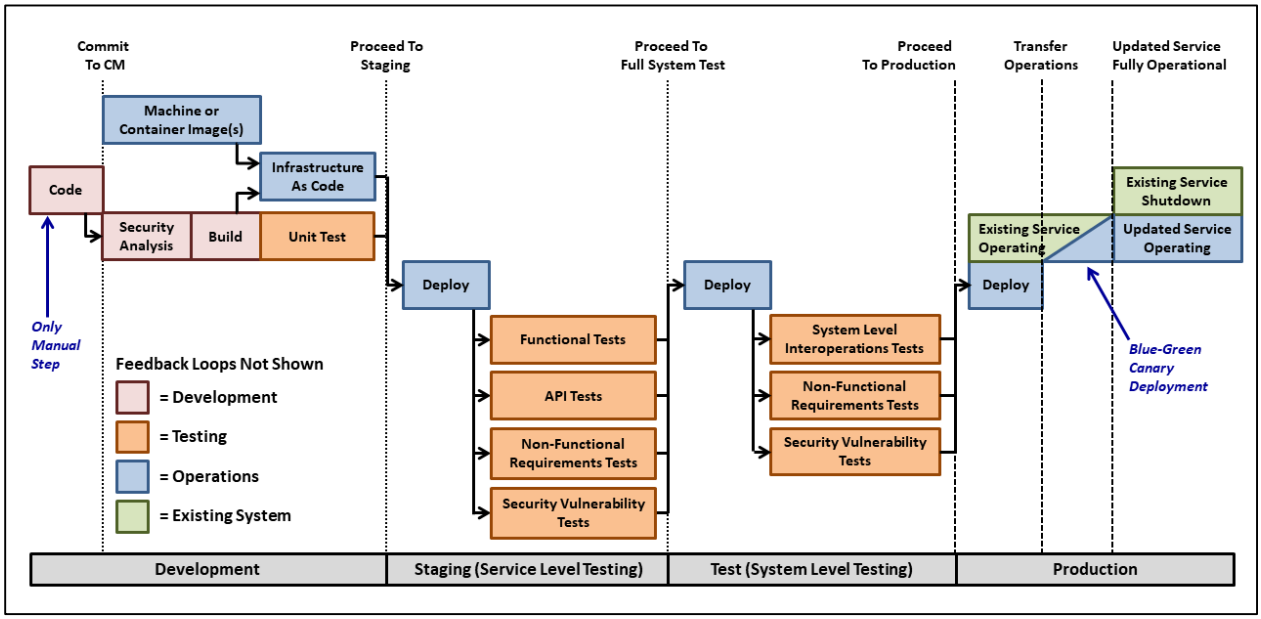

Figure 5: A DevOps Pipeline is the Automated Execution of Software Initially Constructed by the Three Disciplines

Figure 5 is a notional DevOps pipeline which is executed each time a software developer commits and merges software code updates into Configuration Management (CM). This figure highlights the interactions of the three disciplines; Development, Testing and Operations. Although not shown in this diagram, if any step in the pipeline fails, the pipeline stops, and the erroneous condition is reported. Furthermore, each activity in the pipeline is executed through software. Hardware, Operating System and other Commercial Off The Shelf (COTS) components are provisioned by cloud computing resources using configuration files and scripts using a technique called infrastructure as code. The operations personnel create this code that is then repeatably deployed in the staging, test, and production environments. Testing personnel are responsible for building and executing automated software test scripts and procedures that are progressively executed at the module, service, and system level to verify functional, non-functional, performance, and security requirements.

There is a pipeline for each service in the system. When a developer commits and merges a software update into CM, the pipeline automatically starts. The pipeline automatically completes a security analysis to ensure no new vulnerabilities have been introduced into the software. A software build is then completed, and unit tests are performed. If the code passes unit tests, the pipeline then deploys the entire service to the staging environment were a series of service level tests are conducted. If the code passes service level testing, the pipeline automatically deploys the entire service to the testing environment where it is automatically tested with other services at the system level. If the service passes system level testing, it is automatically deployed to a new configuration in the production environment. The pipeline then has the capability, called a canary or blue-green deployment, to incrementally transfer existing users and workloads from the existing, now legacy, configuration to the new configuration in the production environment. The pipeline monitors the new configuration in operation for some time period and if there are no identified problems, the pipeline shuts down the legacy configuration. When properly executed, the time from a software code commit to CM until the software is operationally executing in the production environment can be measured in minutes to hours and without human intervention. It should be highlighted that the entire DevOps team is responsible for building high quality production, testing, and configuration software that can be repeatably executed in a quick, safe, and secure manner and provides new software functionality that is stable, reliable, performant, and secure.

THESE THREE TECHNOLOGIES WORKING TOGETHER

Using the Video Subscription Site example provided in Figure 2, this section describes five scenarios that demonstrate these three technologies working together.

Scenario 1 – New Recommendation Service Algorithm is Developed: In this scenario, the recommendation algorithm is updated to provide a higher weighting score for recently viewed videos. The Recommendation Service is updated, and the associated pipeline is executed that results in a DevOps canary deployment of the new algorithm into operational use without requiring a production outage.

Scenario 2 – Idle Upload Service: In this scenario, there are no video files waiting to be uploaded into the system. As this service is implemented in the cloud using a serverless architecture, the Idle Upload Service is shutdown until a new video is received for uploading.

Scenario 3 – Relational Database Security Patch Available: In this scenario, a software patch is available for the COTS relational database. The only service that uses a relational database is the Subscriber Service. This service is updated with the new patch, and the associated pipeline is executed that results in a DevOps canary deployment of the security patch into operational use without requiring a production outage.

Scenario 4 – Higher Demand for Viewing Videos at Night: In this scenario, more users are at home at night and want to watch videos. Using cloud elasticity, five services scale appropriately to meet the variable user demand. As no videos are waiting to be uploaded and processed, the Upload and the Translate/Transcribe Services are shutdown until a new video is received for processing.

Scenario 5 – New "Find Your Favorite Actor" Feature: The organization has decided to add a feature that allows a user to search, find, and locate their favorite actor in one or more videos. The service architecture for adding the new feature is shown in Figure 6. As new videos are uploaded to the site, the video is forwarded to a new Facial/Object Recognition Service that identifies various actors in the video and marks the time location where the actor is viewable in the video. The output of the Facial/Object Recognition Service is passed to the Catalog/Search Service where this data can be searched through the User Interface Service.

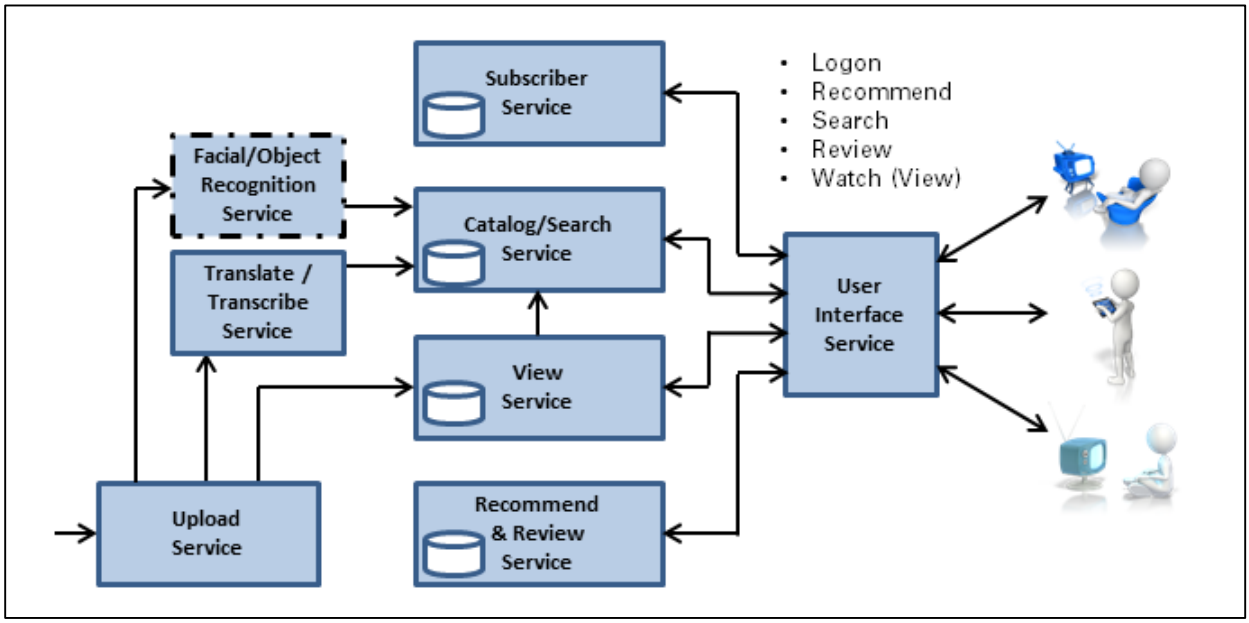

Figure 4: Adding a New Feature Via the Execution of a New Service

The new Facial/Object Recognition Service is implemented using an existing cloud service offered by many cloud service providers. It is executed using a serverless architecture and is shutdown when no videos are waiting to be uploaded and processed. Three existing services require updating. The Upload Service must pass the uploaded video to the Facial/Object Recognition Service. The Catalog/Search Service must be updated to accept the output of the Facial/Object Recognition Service and make the results available for searching. The User Interface Service must be updated to allow users to search for their favorite actor.

Coordination is required between the affected services to move the feature into production. The Catalog/Search Service needs to be deployed first to accept the new data and make it available for search. The Facial/Object Recognition Service needs to be deployed next to provide recognition results to the Catalog Search Service. The Upload Service is then updated and deployed to send new video files to the Facial/Object Recognition Service. Finally, the User Interface Service is updated and deployed allowing users to access and use this new "Find Your Favorite Actor" Feature. Each of these services has their own DevOps pipeline. Thus, the deployment of each service is independent of the other services, provided the sequence of deployment of each service is completed as previously discussed. Each service can use a canary deployment strategy that does not require any outage or down time, from a user perspective.

TALENT, TEAM, AND CULTURAL CONSIDERATIONS

The implementation of these technologies, coupled with agile development practices, provides tremendous benefits but organizations need to consider the impacts to their workforce. A recent study by Cisco, shown in Figure 7, indicates that three of the top four Information Technology (IT) skills in demand are related to cloud computing, enterprise architecture (including services-based architectures) and DevOps practices. The study highlights that many companies will need to retrain and adapt their existing workforce to implement these technologies. (Levy et al. 2019)

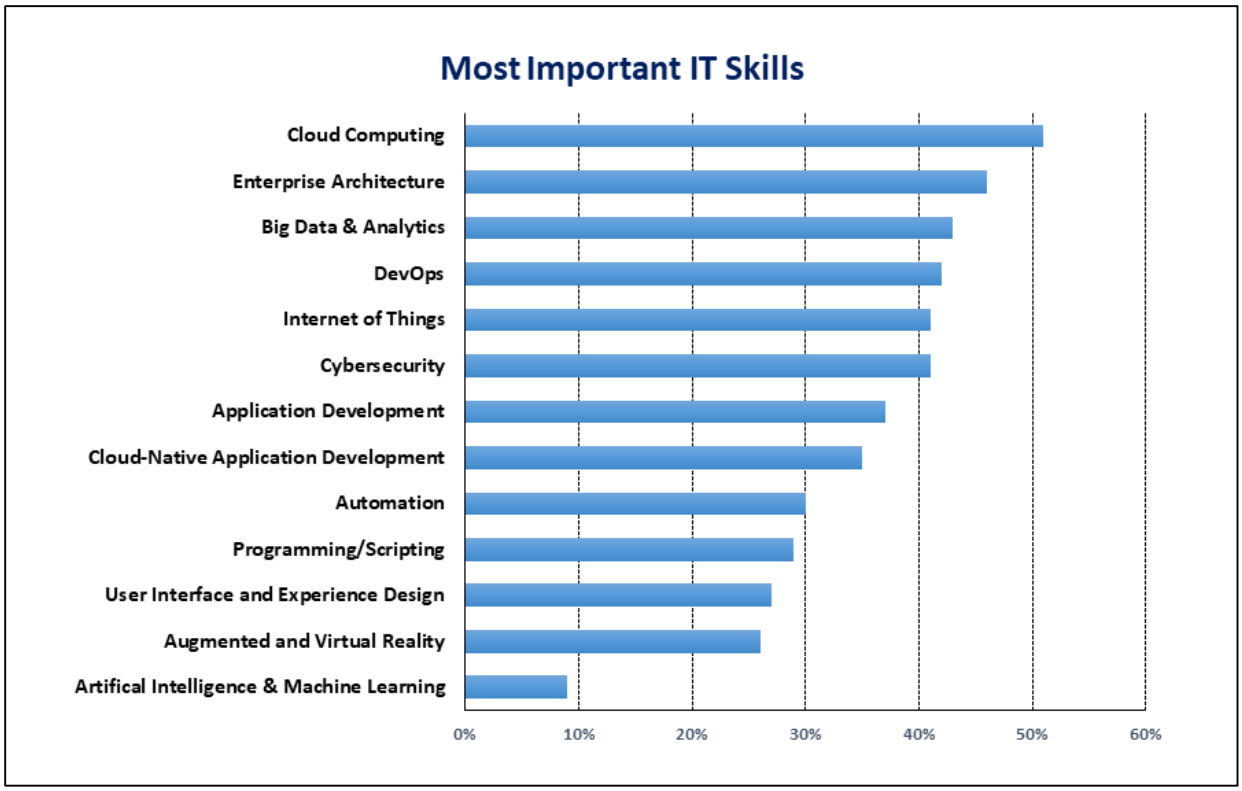

Figure 5: The Skills Required to Execute These Technologies are in the Most Demand (Next-Generation IT Talent Strategies, 2019)

As shown in Figure 8, many organizations align agile teams to the services they own. Furthermore, agile teams are not only responsible for developing services, they now have end-to-end ownership of the service that includes deploying, operating, and supporting their services in

the production environment. As Netflix says, "You built it, you run it." Netflix candidly admits this mindset improves the quality of the delivered services as no developer wants to come in late at night, over the weekend, or on vacation to fix a broken service. (Meshenburg 2016)

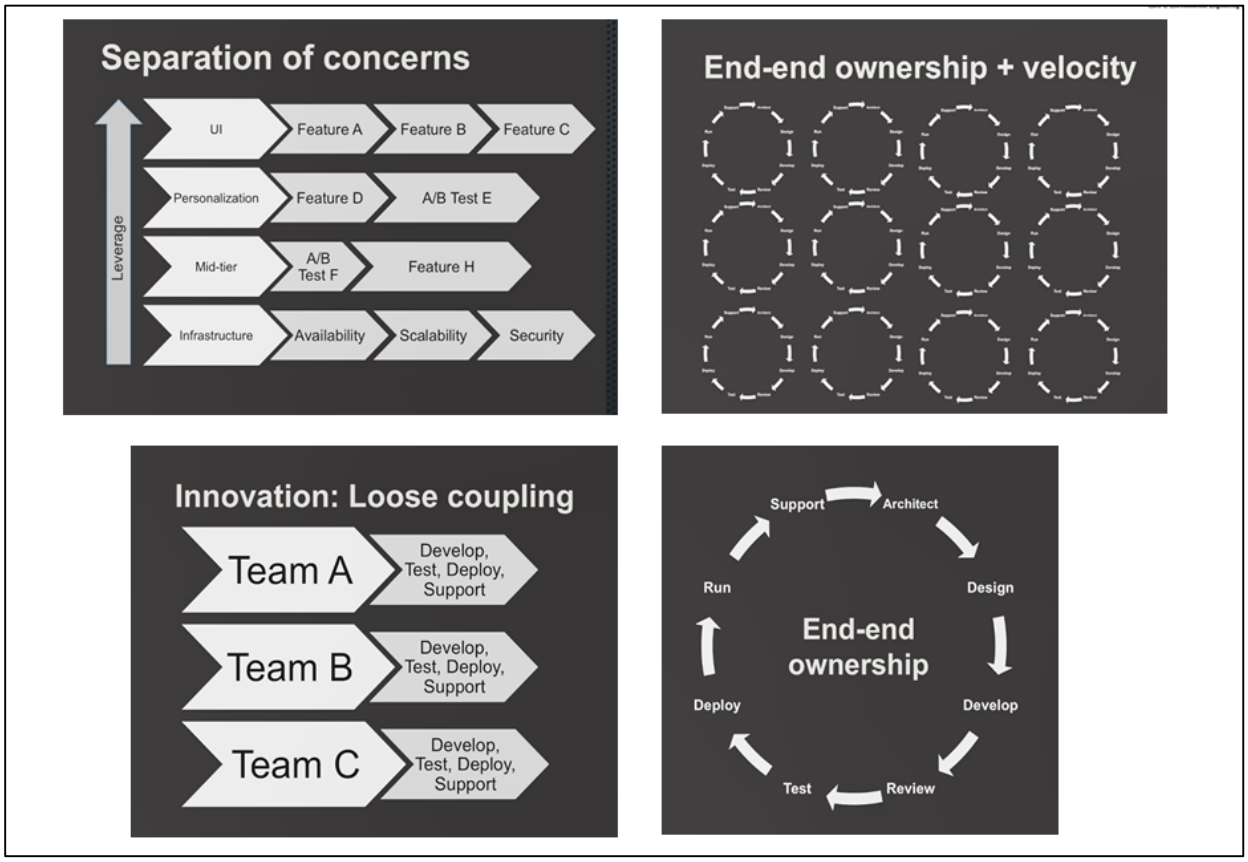

Figure 6: Different Organizational Structures, Responsibilities, and Skills are Required (Microservices at Netflix Scale: Principles, Tradeoffs and Lessons Learned, 2016)

The combined execution of these technologies is fundamentally a cultural shift and organizations need to assess their willingness to move forward by answering fundamental organizational cultural questions such as:

- Is your organization motivated to attract and retain great talent in the name of continued growth?
- Is your organization "promoting" your best developers into management or technical leadership tracks?
- Are your employees stuck in survival mode?
- Is your organization ready to invest in retraining and retooling its workforce?

BIBLIOGRAPHY

United Stated Securities and Exchange Commission Form 10-K, Netflix, Inc. 2018. Los Gatos, CA: Netflix, Inc.

Bieberstein, Norbert, Robert G. Laird, Keith Jones, and Mitra Tilak. 2008. *Executing SOA; A Practical Guide for the Service-Oriented Architect*. Upper Saddle, NJ: IBM Press/Person plc.

Fowler, Martin. "Microservices; a Definition of this New Architectural Term.", accessed March 30, 2019, https://martinfowler.com/articles/microservices.html.

Kim, Gene, Jez Humble, Patrick Debois, and John Willis. 2016. *The DevOps Handbook; how to Create World-Class Agility, Reliability, & Security in Technology Organizations*. First Edition ed. Portland, OR: IT Revolution Press, LLC.

Krafzig, Dirk, Karl Bank, and Dirk Slama. 2005. *Enterprise SOA; Service-Oriented Architecture Best Practices*. Upper Saddle River, NJ: Person Education Incorporated.

Leffingwell, Dean, Alex Yakyma, Richard Knaster, Drew Jemilo, and Inbar Oren. 2017. *SAFe Reference Guide: Scaled Agile Framework for Lean Software and Systems Engineering*. Vol. 2017. Willard, Ohio: Pearson, Education, Inc.

Levy, Eran, Delaney, Kevin, Hill, Jessica and Buckalew, Lauren. "Next-Generation IT Talent Strategies; how CIOs can Close the Skills Gap and Drive True Business Transformation." Connect Futures - Cisco, accessed March 30, 2019, http://connectedfutures.cisco.com/report/next-generation-it-talent-strategies/.

Mell, Peter and Timothy Grance. 2011. *The NIST Definition of Cloud Computing*. Gaithersburg, MD: National Institute of Standards and Technology.

Meshenburg, Rusian. "GOTO 2016 - Microservices at Netflix Scale: Principles, Tradeoffs & Lessons Learned." GOTO Conferences, accessed March 30, 2018, https://www.youtube.com/watch?v=57UK46qfBLY.

Tonse, Sudir. "AWS Re:Invent 2014 (PFC304) Effective IPC in the Cloud: The Pros and Cons of Micro Services Architectures." Amazon Web Services, accessed March 30, 2019, https://www.youtube.com/watch?v=CriDUYtfrjs.