Microservices Enterprise Level-Set

Abstract

This report is a level-set for enterprises considering microservices or alternative approaches to traditional application development and deployment. There have been a multitude of efforts over the years to build better software and services faster and less expensively. The microservices trend has gained increased visibility of late as organizations such as Google, Amazon, Netflix and others have rapidly deployed services at massive scale leveraging agile development teams to build small, loosely coupled services.

Enterprises are evaluating and embracing an overarching movement from traditional monolithic applications to smaller, autonomous services that work together under the umbrella of a microservices-based architecture. This allows for small, independent teams to develop business-focused services without waiting for all other services to be completed. The goal is for this approach to result in efficient development, more consistent and frequent updates and greater flexibility.

At TechVision, we see microservices as enabling three specific, highly sought-after capabilities within the enterprise space: Faster Ideation-to-Realization, Lower TCO via Service Reuse, and Lower Operational Risk. With these three outcomes exists a re-balancing of assumptions at an engineering, financial and organizational level that, when effective, can lead to more integrative and even ‘DevOps’-like practices within the organization. We will continue to evaluate these dimensions as we dive deeper into the ‘what’ and ‘how’ of this architectural pattern throughout this report.

Authors:
Gary Rowe  Patrick McClory
Principal Consulting Analyst/CEO  Principal Consulting Analyst
gary@techvisionresearch.com  patrick@techvisionresearch.com

Contributions From:
Gary Zimmerman  Chris Haddad
Principal Consulting Analyst/CMO  Principal Consulting Analyst
garyz@techvisionresearch.com  chris@techvisionresearch.com
# Table of Contents

Abstract ......................................................................................................................... 1  
Table of Contents ........................................................................................................... 2  
Executive Summary ...................................................................................................... 4  
Introduction .................................................................................................................. 5  
Business Drivers for Microservice Adoption............................................................... 6  
  Catalyst to Strategic Business Drivers ................................................................. 6  
  IT Modernization .................................................................................................... 7  
  Digital Transformation ............................................................................................ 7  
  Growth and Expansion ............................................................................................ 8  
  Intersection of Cloud and Microservices ............................................................. 8  
What are Microservices? .............................................................................................. 9  
  Microservices Goals and Digital Transformation ............................................... 13  
  Microservices, Monoliths and Other Architectures ............................................. 14  
  Microservices and Security .................................................................................... 19  
  Lateral Approach to Governance ....................................................................... 20  
  DevOps and Microservices .................................................................................... 21  
  Microservices and Containers .............................................................................. 22  
  Microservices vs. Other Architectures ................................................................. 22  
  Enterprise Microservices Requirements .............................................................. 24  
Microservices Architecture Details ......................................................................... 25  
  A Microservices Mindset ...................................................................................... 26  
  Modeling Microservices Around Business Domains ........................................... 28  
  Application Development based on Small Services .......................................... 29  
  Independent, Continuous Delivery ..................................................................... 30  
  Minimizing Centralized Service Management ................................................. 31  
Solution Blueprint: A Framework for Adopting Microservices ................................ 31  
  Interface Awareness and Contract First Approach ............................................ 32  
  Culture of Automation ......................................................................................... 32
Continuous Monitoring Program ................................................................. 32
Automated Testing ....................................................................................... 32
Microservices Framework ........................................................................... 33
Microservices Best Practices ...................................................................... 33
Evaluating Microservices ............................................................................ 35
Summary and Recommendations ................................................................. 37
About TechVision ......................................................................................... 39
Appendix A - Key Technical Capabilities and Open Source Options ........... 40
Executive Summary

This report focuses on providing IT executives, enterprise architects, developers and innovators with a “level set” as to what a microservice architecture is, its applicability in large enterprises, the current and expected future state of microservices and guidelines for deciding if, how and when your organization might embark upon this architectural pattern.

Microservices are components that can be thought of as independently replaceable and upgradeable building blocks. These components are independently built and managed and are assembled to create more complex applications as required by the business. That means a change to one component doesn't require the entire system to be redeployed, as is the case with monolithic applications. Microservices decompose applications according a concept of single responsibility in which each microservice is activated via an API and executes discrete business capabilities (e.g. product catalogue, customer profile, reservation, inventory, order, billing, fulfillment) within a specific and well-defined context.

The goal of the microservice pattern is to deliver small, domain-bounded, and independent services. If built correctly, microservice components fit hand-in-glove with the practices of agile programming, continuous integration, and especially continuous (automated) delivery.

Goals associated with microservices are accelerating pace and throughput; all valuable in supporting enterprise digital transformation initiatives. Most modern development organizations are looking for these types of capabilities. Pace starts with faster and more frequent iterations and faster deployment is a key design goal of microservices. Throughput is improved by providing a means of separating tasks and independently designing, scaling and deploying services. This is achieved via automation and increased reuse of easy-to-access components and services.

By separating concerns onto discrete platforms and not sharing database instances or web application hosts across services, every team can choose different runtime languages and frameworks for its own microservice. Also, every team is free to evolve its data schemas, application frameworks, and business logic without impacting other teams. This empowers each team to be a “self-contained unit” and fully accountable for the results.

While TechVision sees substantial benefits and momentum behind microservices for the enterprise, it’s not for everyone or everything today. Most large enterprises have not yet moved to microservices and have monolithic and other applications that will not (and in many cases should not) go away for the foreseeable future. While we are very “bullish” about the future of microservices in the enterprise, tread carefully and incrementally when moving your enterprise in this direction.

Once the decision has been made to move to a microservices pattern, the following best practices are described the help organizations succeed with their early deployments:

- Manage microservices and APIs as “business products” from conception to retirement
We conclude the report with a set of practical steps organizations can take to build out a microservices program.

**Introduction**

This report is part of a series of reports from TechVision Research covering trends in modern software development and operations management practices. As it becomes easier for technologists and even business users to gain access to a greater variety and depth of capabilities ‘with a laptop and a credit card,’ we see a number of technical patterns and practices exerting their influence in this process in a historically disproportionate manner.

**Microservices** architectures, from a software development perspective, are transforming the creation of enterprise applications. This evolving design pattern leverages familiar tools and capabilities from object-oriented programming techniques to high-scale automation to re-balance the set of concerns a given set of applications must consider at a feature, function and operational level. Unique to this pattern is an approach to deploy small, stateless components and services in as granular a manner as possible so as to improve service level agreement (SLA) achievement as well as to reduce the peripheral risk of future updates by scoping the impact of those updates to these small boundaries of concern. In practice, this has the effect of reinforcing an automation-first approach out of necessity and, therefore, has the potential to reduce overall cost of management, improve system availability and, most importantly, can significantly reduce the practical time required to go from ‘business idea’ to deployed software.

This report provides a balanced perspective for enterprise decision-makers on the good and the bad of a microservices architecture approach. This includes the current state of microservices and where we see it moving over the next few years.

Understanding that the current landscape in the application development and systems operations roles continues to evolve, we have, in certain areas, identified how other orthogonal capabilities, patterns and practices can serve as catalysts, accelerators or even as a distraction to the topic at hand.
Business Drivers for Microservice Adoption

Before we dive into the technical underpinnings of the microservices approach for enterprise software development, we’ll first identify the pain points and the business-level ‘give and take’ driving businesses to evaluate new patterns and practices. Realistically, most global companies, especially established enterprises, have a comfort level with their existing teams and processes for accomplishing their technical missions and any change (even for the better) represents a risk to the overall business systems established.

At TechVision, we see microservices as enabling three specific, highly sought-after capabilities within the enterprise space: **Faster Ideation-to-Realization, Lower TCO via Service Reuse**, and **Lower Operational Risk**. With these three outcomes exists a re-balancing of assumptions at an engineering, financial and organizational level that, when effective, can lead to more integrative and even ‘DevOps’-like practices within the organization. We will continue to evaluate these dimensions as we dive deeper into the ‘what’ and ‘how’ of this architectural pattern.

Catalyst to Strategic Business Drivers

Organizations need to be adapt and change at an increasingly rapid pace. This filters down to how we develop software and supporting services. The widely publicized concept of the API economy is an attempt to capture the need for rapid change leveraging standardized services that can be rapidly served up as needed.

Choosing to leverage a microservices architectural pattern can help to reinforce other key strategic technical efforts within. We have found that organizations continue to reinforce three specific, strategic efforts with a focus on the opportunity to accelerate business goals. These three strategic goals are IT modernization, a foundation to support growth and expansion and, ultimately digital transformation.
As articulated in the context of the API Economy concept, a microservices architecture offers a means of delivering on the value that impacts people, process, and technology to accelerate time to market. To achieve this, there needs to be a shift in team organization, processes and practices like DevOps, as well as re-aligning architecture to fit the problems to be solved rather than just layering on new iterations of technology layers.

**IT Modernization**

Many IT departments face the issue that their organization has grown over time, building a complex dependency of operational, organizational and technological legacies. Yet, many elements of the organization are highly dependent on these legacies for day-to-day business operations.

The traditional core IT infrastructure is designed for the stability and resilience required to manage transaction and support systems and is hard to change. The long-standing IT priority has focused on high-quality data management and built-in security to keep core business services reliable for systems leveraging the consistency and reliability of a static data center. While it is hard to move from existing systems and processes, these legacy elements are regarded as barriers blocking the road to digital transformation; deemed ‘too risky’, or assumed to be unable to make the journey. This creates what many have termed a bi-modal view of IT capability; where legacy environments are segregated and managed differently than newer capabilities. But this also isolates (and possibly ignores) the information and knowledge that has been built up over the years and adds a lot of administrative overhead.

Progressive organizations are using API patterns and microservices architectures to connect fresh new applications with legacy data sources or introducing new middleware layers (API gateways) to broker connections into older systems. This can deliver tangible transformation while retaining valuable legacy components. At the same time, it allows the organization to decouple efforts to modernize the underlying core IT capabilities from new development efforts.

**Digital Transformation**

By exposing business capabilities through service APIs, you can get a better understanding of the full customer experience, gain insight into future opportunities, and take advantage of the rich data that is generated. But leveraging APIs are only part of the foundation towards digital transformation.

As organizations seek to better leverage digital capabilities to further all aspects of their business, using patterns such as the microservices architecture are a tangible way to drive business outcomes. Similar to the ‘togetherness’ approach that a DevOps approach embodies, leveraging microservices within a digital transformation program can both help to focus technical teams on tangible business outcomes more directly, but also serve as a means of achieving greater buy-in at a business when the technical approach is more directly mapped to solving historical pain points around scope creep and delivery delays.
Growth and Expansion

A business that’s not growing is dying through lost profits or complacent employees. The last big microservices business driver we’ll examine is Growth and Expansion. With a world population of over 7 billion people, there is practically no limit the size of the market to which you can sell your products. There might be customers with similar needs and the buying power within your borders or in the international market that you simply not have reached. Digital transformation has brought several opportunities to expand beyond a particular geographic region or segment. Taking advantage of these opportunities will require the ability to turn up new channels and to pump out new products quickly—to be able to innovate fast, evolve, and to reuse what you have built rapidly when needed. Microservices are tailor-made for this kind of environment, especially when combined with the flexibility of the cloud.

Intersection of Cloud and Microservices

The movement to microservices starts with the Cloud. The Cloud is one of the key catalysts to enabling the rapid and automated deployment iterations that makes the microservices pattern valuable. In asking the question of why we haven’t designed software like this in the past, the concept of Cloud is really where the answer starts; with the change in a fundamental assumption about computing resources. In the Cloud-era acquiring a server (or an instance) no longer requires weeks of procurement or even days of provisioning on the part of an IT team. IT leaders and developer that want/need resources can now acquire them nearly immediately and, more importantly, can discard them easily at dramatically lower cost. This has set the foundation for innovation, iteration, failing fast and moving even faster.

This shift in assumptions from a resource scarcity model to JIT resource availability has provided an environment supporting the microservices model we describe in this report. We’ll now further describe microservices and their potential enterprise impact.
What are Microservices?

A **microservices** architecture is a means of delivering software applications as a suite of independently deployable, small, modular services in which each service runs a unique process and communicates through a well-defined, mechanism to serve a business goal. In practice, these intentionally decoupled components are then composed into systems via the APIs and interfaces that they expose. Rather than a few, large components that are highly complex, it seeks to distribute capabilities into fault-tolerant, highly available and relatively small pieces of software, hence the name ‘microservices.’

To better frame this analysis let’s start by considering what characteristics microservices have at a high-level. The following list of characteristics of a microservice architecture was first published by James Lewis and Martin Fowler in 2014 and is considered a seminal work on the subject. This should provide a foundation for our deeper analysis throughout this report.

- **Services not libraries.** A component is a unit of software that is independently replaceable and upgradeable, and a microservices architecture is based on services which are independent components who communicate with a mechanism such as a web service request, or remote procedure call. This is different than the common method used in monolithic architecture, component libraries of code included within the codebase.

- **Organized around Business Capabilities.** When looking to split a large application into parts, organizations often focus on technology layers, leading to UI teams, server-side logic teams, and database teams. The microservice approach to dividing up responsibility is different; splitting up teams focused on services organized around business capability. Such services take a broad-stack implementation of software for that business area, including user-interface, persistent storage, and any external collaborations. Consequently, the teams are cross-functional, including the full range of skills required for the development: user-experience, database, and project management.

- **Products not Projects.** Most application development efforts that we see use a project model: where the aim is to deliver some piece of software which is then considered to be completed. On completion the software is handed over to a maintenance organization and the project team that built it is disbanded. Microservice proponents tend to avoid this model, preferring instead the notion that
a team should own a product over its full lifetime. A common inspiration for this is Amazon's notion of "you build, you run it" where a development team takes full responsibility for the software in production. This brings developers into day-to-day contact with how their software behaves in production and increases contact with their users, as they have to take on at least some of the support burden.

- **Smart endpoints and dumb pipes.** Applications built from microservices aim to be as decoupled and as cohesive as possible - they own their own domain logic and act more as filters in the classical Unix sense - receiving a request, applying logic as appropriate and producing a response. Rather than a point of central orchestration, event-driven systems tend to allow for individual services to define their reactions to complex processes rather than relying on complex protocols such as WS-Choreography or BPEL.

- **Decentralized tools.** Rather than being constrained by particular technology standards, microservice practitioners prefer selecting the technologies that best solve the problem. This leads them to produce useful tools that other developers can use to solve similar problems to the ones they are facing. These tools are usually harvested from implementations and shared with a wider group, sometimes, but not exclusively using an internal open source model. Bottomline, standards are developed through consensus, not dictated.

- **Decentralized data management.** Microservices prefer letting each service manage its own database, either different instances of the same database technology, or entirely different database systems - an approach called Polyglot Persistence.

- **Deployment automation.** Many of the products or systems being built with microservices are being built by teams with extensive experience of Continuous Delivery. Teams building software this way make extensive use of automation techniques including automated testing and deployment in order to drive visibility and consistency into their overall software development lifecycle (SDLC).

- **Designed for failure.** A consequence of using services as components, is that applications are distributed across a network and need to be designed so that they can tolerate the failure of other services. Any service call could fail due to unavailability of the supplier and the consumer has to respond to this as gracefully as possible.

- **Evolutionary design.** If you wait to launch until you completely understand and deliver everything a customer wants, the customer will have moved on. The notion is to release, probe, and learn from the market response to the product. Each iteration improves the product and evolves it to match customer preferences. A key characteristic of microservice architecture is developing components that can be tuned and revised independently from the rest of the application.
Microservices are components that can be thought of as independently replaceable and upgradeable building blocks. These components are independently built and managed and are assembled to create the more complex applications as required by the business. A key objective for microservices is that a change to one component doesn’t require the entire system to be redeployed, as is the case with monolithic applications. Early adopters of this pattern tend to find this specific outcome difficult, but a focus on automation beyond just deployment and operational processes can be the key to achieving truly decoupled component deployments.

The notion of microservices has its roots in Domain Driven Design which organizes software programs around business domain objects that have defined boundaries. For example, the relationship between a hotel chain and a guest is complex and can be modeled based on the various interactions such as:

- Guest and online reservation service
- Guest and front desk attendant
- Guest and concierge
- Guest and housekeeping
- Guest and loyalty program

Each of these interactions involves several business domains and trigger events between them. If the boundaries between the domains are not defined correctly, the programs supporting those domains can become less cohesive (for instance, loyalty program objects embedded in the other domains) and more coupled (changes in the loyalty program require changes in other domains). This creates a microservice architecture that is brittle and no longer supports the goal of independence.

However, if organized properly, a microservices architecture becomes a suite of capabilities that can be used in multiple contexts and through many different lenses. This allows for organizations who achieve this model for their purposes the ability to quickly pivot, add ‘value add’ features and have even moderate to large-scale new product launches be more of an effort in re-composing existing resources and services than wholesale and painful rewrites of business-critical software. Given a catalog of available services, even things like the reduction of duplicative work becomes a natural behavior of the overall system rather than a forced discipline requiring regular and costly manual intervention.
Solving Old Problems in a New Way

As noted earlier in explaining the intersection of this topic with the concept of the public cloud, the microservices pattern is truly a reaction to a shift in underlying assumptions of the systems they operate in and around.

One key example of this shift in underlying assumptions is that of the implicit assumption of failure within the public cloud. Traditionally, in a well-deployed virtualized environment on VMWare's solution, an organization would leverage a toolset called vMotion to allow virtual machines to live-migrate between hosts without downtime and (generally) without anyone otherwise noticing. Especially in the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) offering, there's no access to the underlying hosts, and Amazon's CTO Werner Vogels has been quite adamant that 'everything fails all the time.' This shifts responsibility of recovery from an infrastructure-level tool that handles it at a low level to more of an operational concern that can be managed via automation, but has significant impact to deployment topology, especially for systems that might have high availability solutions but where the underlying VMWare capabilities were deemed to be sufficient to meet SLAs.

The paradigm shift in this example cannot be overstated. With the rebalancing of control and responsibility of key layers of the network and system at large, even base assumptions like static IP addresses aren't necessarily safe without additional configuration in a given cloud deployment. This is neither a good or bad thing but opens up an incredible opportunity to design systems that make fewer assumptions of the infrastructure layer of systems which, in our view, tend to make for more robust and reliable deployments. This doesn't single out microservices as the only pattern capable of achieving this in any given environment, but we have found it helpful to articulate these changes in assumptions in reinforcing that responsibilities shift in multiple directions (upward to the application layer, downward to the infrastructure, etc.), but that the needs of the system haven't changed, the tactics to achieve them have.

We discuss the impact of this set of changes in underlying assumption when we discuss the microservices design mindset further in this document.

1 thenextweb.com "Everything Fails All the Time" by Werner Vogels
Microservices Goals and Digital Transformation

Digital transformation is on the radar of most large enterprises; the opportunity to digitally engage with clients, prospects, trading partners and employees requires a flexible, adaptive and scalable set of capabilities and microservices is a candidate for providing this type of foundation. But why microservices and why might an organization consider it now?

We’ll start with the goals associated with microservices; they are pace, throughput and infrastructure optimization; all valuable in supporting enterprise digital transformation initiatives. Most modern development organizations are looking for these types of capabilities and we’ll now describe each of these goals/design elements of microservices and the overall value proposition.

**Pace** starts with faster and more frequent iterations, and faster deployment is a key design goal of microservices. This starts with simplifying development and deployment while minimizing external dependencies. It is also achieved by leveraging every opportunity to automate virtually anything that can be automated. Minimizing dependencies, providing developers with the right tools and quickly provisioning the right resources is key to driving faster development and deployment.

**Throughput** is improved via providing a means of separating tasks and independently designing, scaling and deploying services. This is achieved via automation and increased reuse of easy to access components and services. Being able to deliver code quickly is paramount to getting an edge on the market, and time to market is quickly becoming a driving force in the marketplace.

The **infrastructure optimization** goal is to provide a more resilient, available and scalable set of services required for digital transformation. Microservices can support these goals through the use of granular packaging and deployment techniques including the use of containers.

Microservices are built to scale naturally as more instances of needed components can be added without scaling the entire application. Furthermore, network boundaries provide places to add queues, circuit breakers and other means of both scaling services and mitigating major disasters. More importantly, you also gain scalability in your development teams, as you can add more teams to work independently, something which is hard to accomplish in a monolithic environment.

Key transformative components include leveraging core microservices at scale and providing a platform for event driven interactions. Microservices further supports digital transformation by leveraging an API-centric architecture and event sourcing.
About TechVision

World-class research requires world-class consulting analysts and our team is just that. Gaining value from research also means having access to research. All TechVision Research licenses are enterprise licenses; this means everyone that needs access to content can have access to content. We know major technology initiatives involve many different skill sets across an organization and limiting content to a few can compromise the effectiveness of the team and the success of the initiative. Our research leverages our team's in-depth knowledge as well as their real-world consulting experience. We combine great analyst skills with real world client experiences to provide a deep and balanced perspective.

TechVision Consulting builds off our research with specific projects to help organizations better understand, architect, select, build, and deploy infrastructure technologies. Our well-rounded experience and strong analytical skills help us separate the “hype” from the reality. This provides organizations with a deeper understanding of the full scope of vendor capabilities, product life cycles, and a basis for making more informed decisions. We also support vendors in areas such as product and strategy reviews and assessments, requirement analysis, target market assessment, technology trend analysis, go-to-market plan assessment, and gap analysis.

TechVision Updates will provide regular updates on the latest developments with respect to the issues addressed in this report.