TOGAF® Series Guide

Microservices Architecture (MSA)

Prepared by The Open Group MSA Work Group

TOGAF® Series Guide

Microservices Architecture (MSA)

ISBN: 1-947754-89-8

Document Number: G21I

Published by The Open Group, April 2022.

Comments relating to the material contained in this document may be submitted to:

The Open Group, Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, United Kingdom

or by electronic mail to:

ogspecs@opengroup.org

Contents

Preface

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards. With more than 870 member organizations, we have a diverse membership that spans all sectors of the technology community – customers, systems and solutions suppliers, tool vendors, integrators and consultants, as well as academics and researchers.

The mission of The Open Group is to drive the creation of Boundaryless Information Flow™ achieved by:

- Working with customers to capture, understand, and address current and emerging requirements, establish policies, and share best practices
- Working with suppliers, consortia, and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open source technologies
- Offering a comprehensive set of services to enhance the operational efficiency of consortia
- Developing and operating the industry's premier certification service and encouraging procurement of certified products

Further information on The Open Group is available at www.opengroup.org.

The Open Group publishes a wide range of technical documentation, most of which is focused on development of Standards and Guides, but which also includes white papers, technical studies, certification and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/library.

The TOGAF® Standard, a Standard of The Open Group

The TOGAF Standard is a proven enterprise methodology and framework used by the world's leading organizations to improve business efficiency.

This Document

This document is the TOGAF® Series Guide to Microservices Architecture (MSA). It has been developed and approved by The Open Group.

This document has been structured into 14 chapters as follows:

- Chapter 1 (Introduction) gives an overview of this document
- Chapter 2 (Microservices Architecture Defined) provides comprehensive definition of MSA

- Chapter 3 (Enterprise Architecture in a Distributed World) presents the scope of the Enterprise Architecture as encompassing all of the enterprise's business activities and capabilities, information, and technology that make up the entire infrastructure and governance of the enterprise
- Chapter 4 (Using the TOGAF Framework for MSA) covers using the TOGAF Standard to support development of an MSA
- Chapter 5 (Preliminary Phase) describes the Preliminary Phase of Enterprise Architecture development, covering maturity and readiness assessment and governance and support strategy
- Chapter 6 (Phase A: Architecture Vision) defines the Architecture Vision, including stakeholders, concerns, and business requirements
- Chapter 7 (Phase B: Business Architecture) focuses on the Business Architecture, including decomposition of business functions
- Chapter 8 (Phase C: Information Systems Architectures (Data and Application)) illustrates the Information Systems Architectures
- Chapter 9 (Phase D: Technology Architecture) covers aspects of the Technology Architecture of the TOGAF Standard and their relations to MSA
- Chapter 10 (Phase E: Opportunities & Solutions) describes Opportunities & Solutions resulting from the MSA
- Chapter 11 Phase F: Migration Planning) presents aspects of the Migration Planning of MSA components, including creation of work packages and value and cost reporting of the migration
- Chapter 12 (Phase G: Implementation Governance) describes the process of Implementation Governance and how it differs for an MSA
- Chapter 13 (Phase H: Architecture Change Management) describes the issues associated with Architecture Change Management
- Chapter 14 (Summary) provides a Summary of this document

More information is available, along with a number of tools, guides, and other resources, at www.opengroup.org/architecture.

About the TOGAF® Series Guides

The TOGAF® Series Guides contain guidance on how to use the TOGAF Standard and how to adapt it to fulfill specific needs.

The TOGAF® Series Guides are expected to be the most rapidly developing part of the TOGAF Standard and are positioned as the guidance part of the standard. While the TOGAF Fundamental Content is expected to be long-lived and stable, guidance on the use of the TOGAF Standard can be industry, architectural style, purpose, and problem-specific. For example, the stakeholders, concerns, views, and supporting models required to support the transformation of an extended enterprise may be significantly different than those used to support the transition of an in-house IT environment to the cloud; both will use the Architecture Development Method

(ADM), start with an Architecture Vision, and develop a Target Architecture on the way to an Implementation and Migration Plan. The TOGAF Fundamental Content remains the essential scaffolding across industry, domain, and style.

Trademarks

ArchiMate, DirecNet, Making Standards Work, Open O logo, Open O and Check Certification logo, Platform 3.0, The Open Group, TOGAF, UNIX, UNIXWARE, and the Open Brand X logo are registered trademarks and Boundaryless Information Flow, Build with Integrity Buy with Confidence, Commercial Aviation Reference Architecture, Dependability Through Assuredness, Digital Practitioner Body of Knowledge, DPBoK, EMMM, FACE, the FACE logo, FHIM Profile Builder, the FHIM logo, FPB, Future Airborne Capability Environment, IT4IT, the IT4IT logo, O-AA, O-DEF, O-HERA, O-PAS, Open Agile Architecture, Open FAIR, Open Footprint, Open Process Automation, Open Subsurface Data Universe, Open Trusted Technology Provider, OSDU, Sensor Integration Simplified, SOSA, and the SOSA logo are trademarks of The Open Group.

All other brands, company, and product names are used for identification purposes only and may be trademarks that are the sole property of their respective owners.

About the Authors

John T. Bell

John T. Bell is the founder and Principal Consultant of Ajontech LLC, a small consulting firm providing Enterprise and Security Architecture services for the hospitality industry. John has more than 40 years of experience in IT, 20 of them in hospitality. He has supported development of technology industry standards in IEEE, The Open Group, and in the hospitality industry through HTNG. He was an Associate Professor at Townson University for 14 years and has a long list of publications and presentations spanning his career.

Tony Carrato

Tony Carrato is a former Co-Chair of the SOA Work Group and contributor to the TOGAF® Series Guide: Using the TOGAF® Framework to Define and Govern Service-Oriented Architectures. He is currently an Invited Expert to The Open Group Security Forum and member of the Zero Trust Architecture Project and the Security Forum Steering Committee. Tony is a Distinguished Certified Architect of The Open Group, who recently retired from IBM and is now an independent IT Architecture Consultant.

Chris Harding

Chris Harding is Principal at Lacibus Ltd. For many years Chris was a Forum Director of The Open Group. He supported The Open Group work on SOA, including MSA, until he left The Open Group in 2018. He then founded Lacibus to develop, promote, and exploit a new data platform paradigm, the Virtual Data Lake. He now participates in The Open Group MSA Project as a member.

Chris started his career as a software engineer, and was then a consultant. He has a long and wide experience of the IT industry, on which he draws in his contributions toward The Open Group work, and in online blogs and articles.

Leszek Jaskierny

Leszek Jaskierny is a Master IT Architect with extensive experience in all stages of software development and delivery. Working for Compaq/HP/HPE/DXC Technology since 2002, he designed complex software solutions and delivered projects for major Financial Services Industry customers. He gained programming, solution development, and project leading experience, building Interactive Voice Response (IVR) systems, delivering data management projects and front-end applications. His current focus is on MSA, IoT, and enterprise-scale distributed transnational systems.

Peter Maloney

Peter Maloney is a Senior Engineering Fellow at Raytheon Company. His long career path has taken him from developing and applying solid-state technologies for radar system applications to

his current role as a system architect. He became interested in Enterprise Architectures and particularly SOA as a result of the ever-expanding need for providing access to increasingly complex data products to a diverse group of end users, with the resulting needs for collaboration, throughput management, and security. He is a Raytheon Certified Architect, a program accredited by The Open Group, and a three-time winner of the Raytheon Excellence in Technology Award. He holds one patent and has authored more than a dozen papers.

Ovace A. Mamnoon

Ovace A. Mamnoon is Co-Chair of the MSis, an industry recognized Chief Architect with a passion for Strategic Innovation and Enterprise Architecture. He has many years of architecture and engineering experience that encompass areas that have a strong bearing on MSA including digital enterprise, cloud-native solutions, APIs and SOA, IoT, embedded systems, SmartGrid architectures, and others. Ovace is an active participant in The Open Group and has published a number of papers as the Co-Chair of The Open Group MSA Project.

Ryan Smith

Ryan Smith is a Distinguished Engineer in IBM's Hybrid Cloud Transformation services organization, helping enterprises in all industries with many aspects of their multi-cloud journey including business and IT strategy, application and data modernization, containerization, cloud migration, technical architecture design, operating model transformation, and financial modeling. He has 21 years of experience in IT with a focus on mainframe and distributed application modernization, cloud-native development, and business innovation.

Acknowledgements

(Please note affiliations were current at the time of approval.)

The Open Group gratefully acknowledges members of The Open Group MSA Work Group for their contribution in the development of this document, and in particular the primary authors:

- John T. Bell
- Tony Carrato
- Chris Harding
- Leszek Jaskierny
- Peter Maloney
- Ovace A. Mamnoon
- Ryan Smith

The Open Group gratefully acknowledges the following reviewers who participated in the Company Review of this document:

- Gopala Krishna Behara, Wipro Technologies
- Sonia Gonzalez, The Open Group
- Judith Jones, ATE Enterprises

Referenced Documents

The following resources are referenced in this TOGAF® Series Guide:

- Benefits of DevOps Methodology for Microservices Solutions (W196), White Paper, published by The Open Group, June 2019; refer to: www.opengroup.org/library/w196
- ISO/IEC 18384: 2016 (Parts 1 to 3): Information Technology – Reference Architecture for Service-Oriented Architecture (SOA RA)
- Microservices, IBM Cloud Education, March 2021; refer to: https://www.ibm.com/cloud/learn/microservices
- Microservices Architecture (W169), White Paper, published by The Open Group, July 2016; refer to: www.opengroup.org/library/w169
- SOA Source Book (G102), The Open Group Guide, published by The Open Group, August 2011; refer to: www.opengroup.org/library/g102
- The Open Agile Architecture™ Standard, also known as the O-AA™ Standard (C208), published by The Open Group, September 2020; refer to: www.opengroup.org/library/c208
- The Open Group TOGAF® Standard website; refer to: http://www.opengroup.org/togaf/
- The TOGAF® Standard, 10th Edition, a standard of The Open Group (C220), published by The Open Group, April 2022; refer to: www.opengroup.org/library/c220
- TOGAF® Series Guide: Using the TOGAF® Framework to Define and Govern Service-Oriented Architectures (G174), published by The Open Group, September 2017; refer to: www.opengroup.org/library/g174

1 Introduction

The purpose of this document is to contribute to The Open Group mission of Boundaryless Information Flow™ by providing guidance on how the architect can use the TOGAF® Standard (see Referenced Documents) to develop, manage, and govern Microservices Architecture (MSA) or any architecture where MSA is part of the scope.

This should promote the shared understanding of the MSA creation process according to the principles of distributed architecture, and significantly strengthen the alignment between the cultures of business and information technology. The further adoption of MSA as an architectural style in a cloud environment would result in the use of the method, metamodel, references, and other TOGAF facilities.

While many of MSA characteristics are similar to Service-Oriented Architecture (SOA), this document is dedicated to MSA only, and readers looking for SOA should reference the existing TOGAF® Series Guide: Using the TOGAF® Framework to Define and Govern Service-Oriented Architectures (see Referenced Documents).

1.1 Overview

As the market environment becomes more dynamic, companies are moving towards management of their complexity and business agility issues. Networks of current software components and their interfaces build highly complex landscapes where change management becomes more difficult and it is harder to anticipate and comprehend the consequences of change. The modern market environment also brings demands to enhance system availability and scalability, two characteristics that are drivers in the adoption of MSA.

MSA is an architecture style that defines and creates systems through the use of small independent and self-contained services that align closely with business activities. An individual microservice is a service that is implemented with a single purpose, that is self-contained, and that is independent of other instances and services. Microservices are the primary Architecture Building Blocks (ABBs) of an MSA. As part of an Enterprise Architecture, MSA brings a layer of service capabilities to extend the business capabilities of the enterprise and, where the enterprise is a small self-contained service, MSA may be perceived to be the Enterprise Architecture. Refer to The Open Group White Paper: Microservices Architecture (see Referenced Documents).

The concept of MSA is designed to simplify business operations and to promote the interoperability of the various parts of that business, in order to increase business agility. It is possible to easily define an organization's functional capabilities by structuring capacity as meaningful, granular services as opposed to opaque, siloed business units, without duplicating similar capabilities across the organization. Precisely defined granularity, aligned with business domain and clearly defined interfaces, limits the impact of changes and allows understanding of mutual dependencies.

From a software development perspective, MSA focuses on breaking applications into a set of independent microservices, promoting system flexibility and agility that is necessary in today's complex and fast-moving business environment.

2 Microservices Architecture Defined

Microservices Architecture (MSA) is an architecture style, in which software systems or applications are composed of one or more independent and self-contained services. It is not a product, framework, or platform. It is a strategy for building large distributed systems. Microservices are loosely-coupled and deployed independently of one another.

The SOA Source Book (see Referenced Documents) defines a service as an instantiation of a logical representation of a repeatable business activity that has a specified outcome. We define a microservice as a self-contained service that implements an Atomic Business Function (ABF) and is independent of other services. MSA is a style of architecture that defines and creates systems through the use of small, independent, and self-contained services that closely align with business activities.

Microservices are the primary ABBs of an MSA. An MSA has the following three key characteristics:

- Service-independence: each microservice is independent of other microservices; each microservice is developed, deployed, and changed independently
- Single responsibility: each microservice is mapped to an atomic (single) business activity for which it is responsible
- Self-containment: a microservice is a self-contained, independent deployable unit encompassing all external Information Technology (IT) resources (e.g., data sources, business rules) necessary to support the unique business activity

For more detail on characteristics, features, and governing principles of microservices and MSA, refer to The Open Group White Paper: Microservices Architecture (see Referenced Documents).

The key benefits of using microservices are:

- Technology-independence: each microservice can be built on its own technology, which enables the system to move away from any technology at any point in time and adapt to newer models
- Re-usable: microservices are highly re-usable and can be composed with other services
- Deployment flexibility: a microservice has its own lifecycle, and can be built (or changed), tested, and deployed on its own
- Flexible scaling: scaling of an MSA-based solution is easily achieved because instantiation of additional services can be performed independently
- Decoupling: independence, the inherent characteristic of microservices, enables decoupling where clients are not aware of the implementation details, and microservices are not aware of each other's function – consequently, decoupling is at a high level in the microservices environment

3 Enterprise Architecture in a Distributed World

3.1 Overview

The TOGAF Standard describes the scope of the "Enterprise Architecture" as encompassing all of the enterprise's business activities and capabilities, information, and technology that make up the entire infrastructure and governance of the enterprise.

The architecture crosses multiple systems and multiple functional groups within the enterprise. This is one of the key factors that has driven the evolution of all domains of architecture – i.e., Business, Data, Application, and Technology – to more optimally cater to the needs of this distributed enterprise.

Distributed architectures have organizational and people implications, in addition to technology implications. Typically, a greater degree of alignment is needed between the modular nature of the solutions as well as the teams. Large, centralized models should be decomposed into leaner, federated models.

Each entity of the enterprise has different solution needs – their agility and time-to-market, and their pace of change. A modular approach to solutions enables the enterprise to build in flexibility for both short-term and long-term needs. The benefits of modular solutions, such as microservices, are more justifiably covered in other documents published by The Open Group, such as The Open Group White Paper: Microservices Architecture (see Referenced Documents).

The mandate of an Enterprise Architecture, in a distributed enterprise, is to ensure the congruency and homogeneity of capabilities and standards amongst these distributed, modular solutions (including partner solutions), while simultaneously allowing for the independence and polyglot needs of each, so that they are aligned with business strategy and provide clear business value. Areas that require particular attention include:

- Interoperability through standardized, well-defined contracts for APIs and other interfaces
- A reference architecture that has the ABBs to address interoperability (integrations, APIs, events, state, and data), visibility (telemetry, monitoring, reporting), management and operations (paramount if the deployment architecture includes a footprint in the cloud), and security
- Well established standards for the enterprise in each of the TOGAF Enterprise Architecture domains (Business, Data, Application, and Technology) to enable synergy across the organization

  These standards should be for enterprise strategic alignment, and should not hinder the needs of the individual modular units. Each unit should inherit from the enterprise standards and extend those with their own.

- Appropriate level of governance, which addresses the needs of the enterprise without sacrificing the autonomy of the individual entities of the enterprise – a federated governance model is a better fit than a centralized one

One of the challenges of distributed architectures is the ability to provide an aggregated view of multiple ecosystems that could include any combination of public cloud, traditional infrastructure, mainframe, and Software as a Service (SaaS) workloads. This requires robust enterprise-level foundations for DevOps, monitoring, and reporting, test automation, governance, and security.

The role of Enterprise Architecture in a distributed, hybrid cloud world is to design, incubate, and propagate re-usable assets such as reference architectures, patterns, DevOps pipelines, and service management solutions. These assets are able to support traditional and cloud platform governance. Governance can be centralized; however, optimized and agile enterprises have seen better success with de-centralized governance – i.e., small "g" governance; see Chapter 5 (Preliminary Phase) and Chapter 12 (Phase G: Implementation Governance) – allowing each business value stream to inspire open innovation from self-directed teams composed of business and IT members working in integrated units.

The following sections provide a summary view of the implications of distributed architectures from each of the Enterprise Architecture domains. A more comprehensive coverage of each is in their respective chapters of this document.

Note that this document covers MSA as an architecture style supported by Enterprise Architecture, and therefore does not cover Enterprise Architecture as a whole.

3.2 Implications for Business Architecture

Business Architecture has a pivotal role in the distributed world. In addition to ensuring directional alignment with enterprise strategy, it takes on greater responsibility in areas of process and people. Culture change and redesign of the Business Architecture are often major challenges for enterprise transformation. For example, traditional Business Architecture domains are often mapped to value streams that collapse and expand the existing Business Architecture design, and this has implications on organizational design and resources.

3.3 Implications for Data Architecture

Distributed architectures, and more specifically MSAs, introduce more copies of data than centralized solutions. In order to meet certain non-functional requirements such as response time and availability targets, data may need to be copied to multiple data stores (e.g., operational data store of a microservice, cache, edge device) so that the lack of availability and latency of upstream dependencies do not negatively impact the overall performance and availability of the microservice and ultimately the user experience of the applications dependent on the microservice. Data Architecture and governance therefore become more complex as the integration requirements, messaging, and analytic data related to these microservices evolve.

3.4 Implications for Application Architecture

Monolithic application owners have the luxury of making decisions for all components of an application, whereas distributed systems based on microservices will have dependencies and constraints imposed by the owners of the microservices consumed by the apps.

APIs will become much more prevalent and backwards-compatibility will be a growing challenge depending on the complexity of the application and the number of services it consumes. The mandate for microservices to be self-contained results in architectures with less complexity than monolithic architectures. Scalability, elasticity, and resiliency become important non-functional characteristics especially for microservices that are re-used heavily across the enterprise or by a large number of consumers (i.e., consuming applications and ultimately the total number of transactions per day).

3.5 Implications for Technology Architecture

Technology Architecture in the distributed world has a much wider scope to address, including cloud (public, private, and hybrid), containerized clusters, edge devices, etc. Though many of the architectural decisions for technology are within the autonomous teams' responsibilities, the Enterprise Architecture will need to define guardrails and re-usable automation components for key platforms and patterns. For example, microservices can run in serverless and container platforms, yet there are pros and cons for each deployment option. However, with the growing demand for cloud managed services, the ability to select optimal technologies without the need to recruit an expert to manage the full stack mitigates risk through these trusted relationships with selected cloud providers. Microservices can just as easily run on traditional systems and edge devices. The implication here is that the number of technology choices is rising so prescriptive guidance from the enterprise perspective to autonomous development squads (encouraged through the availability of re-usable code and reference architectures) is extremely important to avoid future technical debt and a portfolio that becomes even more heterogeneous over time.

4 Using the TOGAF Framework for MSA

An effective Enterprise Architecture is critical to business survival and success, and is the indispensable means to achieving competitive advantage through IT. The TOGAF Standard is a detailed method and a set of supporting tools for developing Enterprise Architectures. It codifies the good practice that has evolved in the work of IT and Enterprise Architects over many years. This document will help the architect to decide where and how to use MSA so it is less about the Enterprise Architecture; rather it focuses on Data, Application, and Technology Architecture design in the context of microservices.

The TOGAF Architecture Development Method (ADM) breaks the complex process of architecture development into a number of simpler steps, or phases, in which the architect considers different aspects of the overall problem:

- Preliminary Phase
- Phase A: Architecture Vision
- Phase B: Business Architecture
- Phase C: Information Systems Architectures (Data and Application)
- Phase D: Technology Architecture
- Phase E: Opportunities & Solutions
- Phase F: Migration Planning
- Phase G: Implementation Governance
- Phase H: Architecture Change Management
- ADM Architecture Requirements Management (continuous process for MSA)

Those familiar with the TOGAF Standard will recognize the following graphical depiction of the ADM in Figure 1. This document will only focus on the microservices layer of the architecture and therefore only a subset of the ADM is relevant, as described below Figure 1.

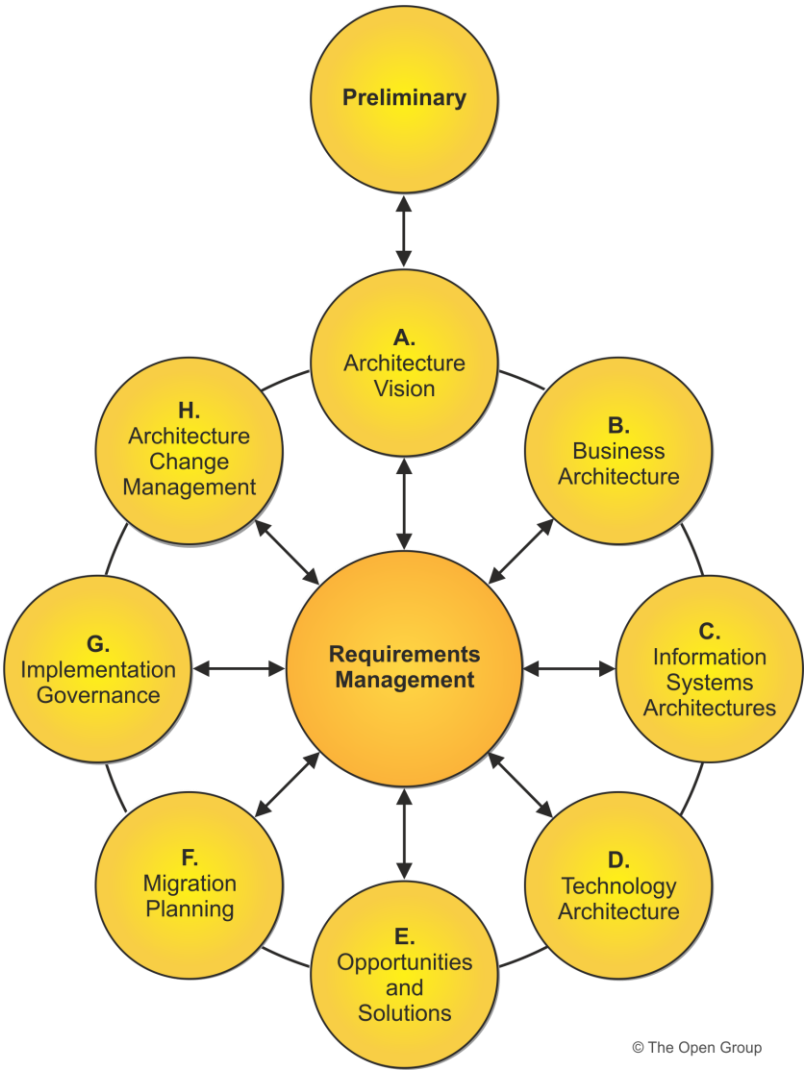

Figure 1: TOGAF Architecture Development Method (ADM)

The following chapters describe, for each phase of the TOGAF ADM, what the architect should consider, particularly when looking to apply MSA, and how this affects the outputs of the phase. In short, it explains how to use the TOGAF Standard to do MSA.

This is not a standalone description. It assumes knowledge of the TOGAF Standard, and leaves out everything that is not related to MSA. The architect can find all the information needed on The Open Group TOGAF Standard website.

5 Preliminary Phase

The TOGAF Preliminary Phase is about preparing the enterprise to determine "where, what, why, who, and how we do architecture" in the enterprise, concerning all the ABBs.

An MSA provides the service layer in an architecture, and the associated infrastructure for those services. To reap the major benefits of an MSA, it usually focuses on the technology capabilities in the enterprise and may be a vital cog in providing a set of highly reliable and scalable capabilities. This document provides information for the MSA layer; information for the development of the Enterprise Architecture is provided in other sources, such as the TOGAF® Series Guide: Using the TOGAF® Framework to Define and Govern Service-Oriented Architectures (see Referenced Documents).

Given this role for the MSA within an overall enterprise, it is worth noting that coordination and alignment with other stakeholders will always be a mandatory concern during the development of an MSA. How Enterprise Architects will engage and support an MSA implementation is an enterprise-level decision. The implementation of the MSA for a particular set of business capabilities, and the technology choices for that implementation, will be the responsibility of the Project Architect. A typical set of stakeholders is listed in Table 1.

Table 1: MSA Implementation Stakeholders

Stakeholder	Role Within the Enterprise
Enterprise Architects	Responsible for overall Enterprise Architecture. Prescribe company standards and chair Architecture Review Board.
Architecture Review Board	Responsible for reviewing and coordinating all Enterprise Architectures for compliance to company standards. Requests for waivers and deviations must be approved by this board.
Project Architect	Responsible for the architecture development of an individual project (in this case, the MSA). Must coordinate with other stakeholders to ensure project success.
Domain Architects	(Common in larger enterprises.) Responsible for the architecture of a particular bounded context within the enterprise (Business, Data, Application, and Technology).
Development/Operations/IT	Responsible for the development of the deployed systems and their daily operations. MSA is well aligned with DevOps methodologies, emphasizing automation, feedback, and matching of development to production environments. Substantial autonomy is granted at the team level, while maintaining alignment with Domain and Enterprise Architects on technology selections and roadmaps.

The TOGAF framework provides for incremental architecture development. This framework still applies to an MSA, although the emphasis in some phases will be different. The Preliminary

Phase ensures the skills, capabilities, and governance required for MSA are addressed. Each cycle through Phases A to H creates an increment to the architecture, with Requirements Management being a continuous phase to capture all MSA requirements and enable a dynamic architecture approach. It should be noted that for MSA these cycle times can be expected to be quite short, measured in durations of weeks rather than months.

The Preliminary Phase is where the architect adopts the principles of service design particular to an MSA: independence, self-containment, and single responsibility. This decision is necessary even if the enterprise is already service-oriented. This affects two other outputs of the phase: the governance and support strategy, and the content of the initial Architecture Repository.

5.1 The Principles of MSA

The starting point for MSA development with the TOGAF framework is that the architecture team for the project under consideration adopts the key principles of an MSA as architecture principles. (Again, regardless of whether the Enterprise Architecture has already adopted service-orientation as a guiding architecture principle.)

Architecture principles define the underlying general rules and guidelines for the use and deployment of all IT resources and assets across the enterprise. They reflect a level of consensus among the various elements of the enterprise, and form the basis for making future architecture decisions. The Preliminary Phase defines the architecture principles that will form part of the constraints on any architecture work undertaken in this project. They are typically developed by the lead Project Architect, in conjunction with key stakeholders, and are approved per the governance which controls this project. Note that the architecture for any project must be consistent with the overall Enterprise Architecture and its adopted architecture principles, or else a waiver must be explicitly requested and approved by the Architecture Review Board. The architecture principles are included in the Tailored Architecture Framework, which is an output of the Preliminary Phase.

The key architecture principles for an MSA, based on microservices, are enumerated in The Open Group White Paper: Microservices Architecture (see Referenced Documents. These principles will not all be replicated here, but we reiterate the principle of service-independence as an example, since it is the fundamental principle which otherwise distinguishes an MSA from an SOA, of which it is a subset.

Principle	**Independent Services**
Statement	A microservice is independent of all other services.
Rationale	Independence of services enables rapid service development and deployment, and permits scalability through instantiation of parallel, independent services. This characteristic also provides resilience; a microservice is allowed to fail and its responsibilities are taken over by parallel instantiations (of the same microservice), which do not depend on other services. When a microservice fails, it does not bring down other services.
Implications	Both design and runtime independence of services are required. It is necessary for the business to determine whether providing scalability and resilience of the business function are paramount considerations. If so, MSA provides a means of achieving these characteristics.

An enterprise wishing to use the TOGAF framework for MSA should include this principle in its set of architecture principles. The many benefits in resilience, scalability, and reliability which can be achieved with an MSA will not be attained if this principle is ignored.

If the architect is introducing the TOGAF framework to an enterprise that is already committed to the use of microservices and distributed architectures, or that is part of a larger enterprise that has made a strategic decision to use MSA, then adoption of these principles is a given. If, on the other hand, the project is adapting, for example, a legacy monolithic application to an MSA, or is launching an entirely new development, there are implications for the organization and the architecture governance which must be considered during this Preliminary Phase.

5.2 Maturity and Readiness Assessment

Development and deployment of a distributed MSA assumes that the enterprise is either already or prepared to become service-oriented as a starting point. The process of conducting an SOA maturity assessment during the Preliminary Phase, using The Open Group Service Integration Maturity Model (OSIMM), is described in the SOA Source Book (see Referenced Documents) as part of the review of the organizational context for conducting Enterprise Architecture, and will not be replicated here. Other aspects more specifically tied to implementation of an MSA should also be considered.

- Team organization – relevant skills and competencies for the microservices required

  "Stovepiped" in this context refers to the independence of an MSA team in the vertical sense, meaning that the entire lifecycle from architecture development through technology selection and microservice development and deployment is owned by the team responsible for that microservice. Development and deployment of the microservice components of a distributed architecture will not be effective if the core MSA principle of service-independence is not reflected in the organization. Individual teams must have the flexibility of selecting the optimum technologies and implementations which best suit their project development, always of course subject to the overall constraints laid down in the Enterprise Architecture.

- Considerations of microservice lifecycle *versus* team ownership

  It is a common best practice in distributed systems development for a single team to own a microservice over its entire lifecycle, from development through deployment and ultimate retirement. This is an organizational decision that should be made early on.

- Agile/DevOps methodologies

  Although we are not attempting to be prescriptive with respect to methodologies, we would be remiss in not noting the rapid adoption of Agile/DevOps methodologies across the industry. These are particularly well-suited to distributed architecture implementations such as MSA and cloud-native, and should be examined seriously by the Enterprise Architecture team. These methodologies will also have an impact on the membership of the Enterprise Architecture team itself.

As with the introduction of any significant new idea, it is good to start with a small project and learn from experience before implementing on a wide scale. The architect can undertake a complete but rapid TOGAF cycle, without spending too much effort on detailed analysis, to define a pilot project. Successful implementation of that project will then lead to final adoption

of the principle and close off any maturity assessment gaps identified over time. Of course, the organizational implications of an MSA distributed architecture must be assessed as part of this learning process.

5.3 Governance and Support Strategy

The TOGAF Standard does not attempt to describe all aspects of implementation and operational governance; only those areas directly related to the architecture under development. It assumes that detailed governance for those areas is in place. The Preliminary Phase includes confirmation of the architecture governance and support strategy as part of the Organizational Model for Enterprise Architecture. It needs to be understood up-front that the governance requirements for a distributed architecture such as MSA are going to be very different from traditional Enterprise Architectures, even those focused on SOA. In order to function efficiently, an MSA team must follow two parallel tracks for governance; making its own governance decisions as needed, applied only to that MSA, while still residing within the guardrails established for the enterprise.

It should also be borne in mind that governance is not about developing a set of governance requirements, but rather the process and approach to apply them. Following the agile principles, governance should become an ongoing activity performed by the architects working side-by-side with implementation teams.

As noted earlier, the MSA will be by definition a subset of the overall Enterprise Architecture. In order to allow for efficient development of MSA/cloud services, the individual development teams must have the freedom to select the optimum technologies and methodologies to speed their development and deployment cycles, which will change as a function of time, if nothing else. At the same time, these teams must still adhere to the overall governance rules established by the Enterprise Architects, or explicitly ask for a waiver allowing them to take exception to one or more of these rules. Failure to adhere to the rules established by the Enterprise Architects runs the risk of institutionalizing technological chaos and instability.

It may not be appropriate to undertake the detailed development of governance rules and procedures as part of the Preliminary Phase. It could be better to confirm the architecture governance procedures, and to commission a separate project to define implementation and operational governance procedures before implementation starts.

For MSA governance, although we are not prescribing a DevOps methodology, a distributed and rapidly changing set of microservices will require the following kinds of governance tools:

- Appropriate levels of review prior to any change releases (e.g., “two sets of eyes” methodology)
- Least privilege concepts applied to protect production pipelines
- Robust change monitoring
- High levels of automation – both of monitoring functions and testing

For further information on DevOps and MSA, we recommend reviewing The Open Group White Paper: Benefits of DevOps Methodology for Microservices Solutions (see Referenced Documents). A high-level view of how MSA governance is related to the Enterprise Architecture and IT governance is given in Figure 2.

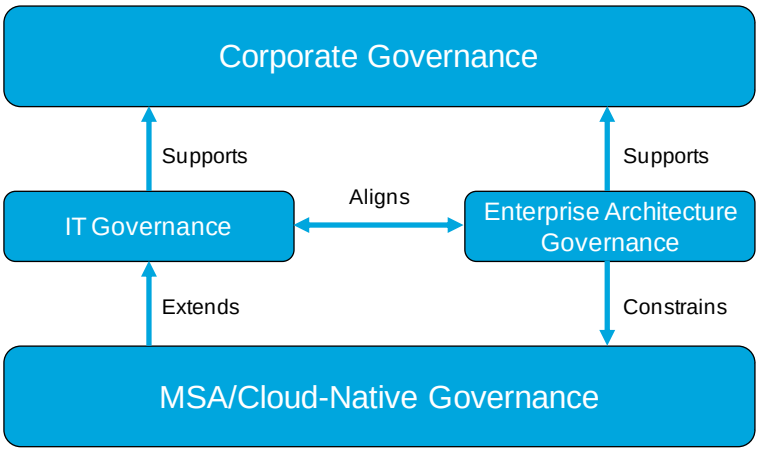

Figure 2: SOA Governance Supports IT and Enterprise Architecture Governance

The MSA Governance Reference Model is largely identical to the SOA Governance Reference Model, other than being restricted to MSA processes. It is depicted in Figure 3.

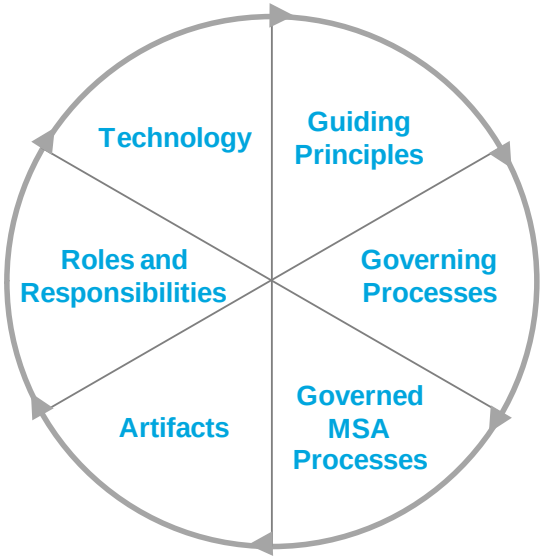

Figure 3: MSA Governance Reference Model

Any governance model should be viewed as a process and not a project; therefore, the phases of the Governance Reference Model should be viewed as a continuous improvement loop, whereby progress is measured and course-correction and updates to the processes are performed when needed, as indicated in Figure 3.

5.4 Initial Architecture Repository and the MSA Reference Architecture

The enterprise's Architecture Repository contains a collection of models, patterns, architecture descriptions, and other artifacts that are available for the development of its architectures. They may result from previous architecture work in the enterprise or from work in other enterprises, or

in industry bodies. The TOGAF Preliminary Phase includes the establishment of an Architecture Repository with an initial collection of material.

The Architecture Repository may contain numerous artifacts that are outside the scope of the MSA. For this layer, the primary artifacts of interest to the team are those which identify microservices that have already been developed, the provenance of those services, and how to interact with them. This is addressed further in Chapter 12. Although service repositories have largely been proven as ineffective in the past, solutions exist to provide for this sort of microservice discovery and prevent duplication of effort. Microservices are reachable only through a published API, which nowadays is typically managed through an API management framework, which encompasses both service meshes and API gateways. This is evolving with work such as the OpenAPI Specifications (OAS).[1]

An MSA Reference Architecture is in development. It will be focused, as has been identified in other MSA White Papers, on how an MSA is designed to meet the requirements of a specific business need, as contrasted with the focus on enterprise integration of the analogous SOA Reference Architecture (refer to ISO/IEC 18384 – see Referenced Documents). (As noted above, an MSA is not a complete Enterprise Architecture; it is an architecture only concerned with the layers at which the (micro) services of the architecture reside.) They are described as follows:

- Operational Systems Layer:

  — Programs and data of the operational systems of the enterprise

  — The new and existing infrastructure needed to support the SOA solution

- Microservice Components Layer:

  — Software components, each of which provides the implementation or "realization" for a microservice, or operation on a microservice, and binds the microservice contract to the implementation of the microservice in the operational systems layer

- Microservices Layer:

  — Microservices, with their descriptions, contracts, and policies, and the containers that contain the microservice components

  — Microservices and programs that support the application of the rules and the operation of the procedures

- Business Processes Layer:

  — Business processes, and compositions in which business processes are composed of other business processes and services

  In an MSA, we only worry about the relationship between a particular ABF and its implementation by the microservices. The requirements for these business processes will tend to be also much more distributed for an MSA implementation, since we are likely dealing with a federated organization as well as a federated architecture. Each part of that federated organization will be to a large extent autonomously responsible for its own requirements.

[1] Refer to: https://www.openapis.org/.

- Consumer Interfaces Layer:

  — The programs by which the users interface to the microservices

- Integration Layer:

  — Integration of and communication between other building blocks, including messaging, message transformation, complex event processing, microservice composition, and service discovery

  Service composition can only take place at the application layer, not within the MSA.

- Quality of Service Layer:

  — Monitoring and management of the quality of service of the architected system, including its performance, reliability, availability, scalability, security, and manageability

- Information Layer:

  — Management, analysis, interpretation, and transformation of data

- Governance Layer:

  — Governance rules and procedures

  — Microservices and programs that support the application of the rules and the operation of the procedures

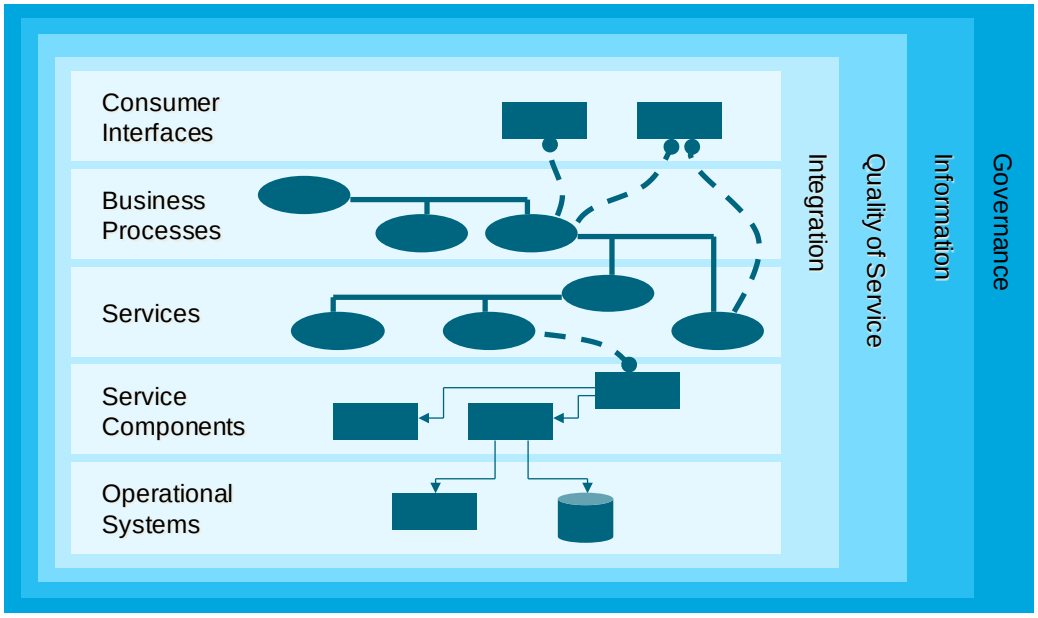

Figure 4: SOA Reference Architecture

5.5 Partitions and Distribution of Responsibility: Establishing the Architecture "Team"

The TOGAF framework establishes the architecture team and organization – team structure, roles, responsibilities, etc. – in the Preliminary Phase to support a desired architecture capability. With a distributed architecture such as MSA it is important for the organization to allow partitioning of the architectural elements among a variety of development teams. The federated nature of an MSA solution should be reflected in the federated nature of the organization.

Different teams will work on different elements of architecture at the same time. Partitions allow for specific groups of architects to own and develop specific elements of the architecture (see partitions and scoping in Phase A). It is suggested that the team start with a focused initiative before implementing on a wide scale.

A successful architectural team organization will have several key attributes:

- A clear definition of each team's mission: why it exists, its scope of responsibility, and what the organization and the architecture practice should expect from the team
- Clear goals for the team including measurements and Key Performance Indicators (KPIs); it is important to ensure that the measures and KPIs of the team do not drive inappropriate selection of technologies and architectural choices
- The teams must always remain cognizant of the overall constraints and responsibilities laid down by the Enterprise Architecture team
- Interchange and dissemination of skills, experience, and capabilities across teams to the rest of the architecture practice is to be strongly encouraged
- Identify how members of the team and other architecture practitioners will be rewarded for success
- At the start, perform a skills assessment across the organization to determine the distribution of necessary skills to create fully functional teams
- The necessary skills and experience must be carefully identified, and where they are not present, acquired

  A fundamental skill for leading practitioners within the organization is the ability to mentor other practitioners transferring knowledge, skills, and experience.

5.6 Summary

In summary, when developing the Preliminary Phase, there are a number of methods, tools, and reference materials that have been developed by The Open Group to help the Enterprise Architecture team develop their MSA. These include:

- Principles: microservice-independence
- Determining organization readiness for MSA: OSIMM, independent team organization, Agile/DevOps methodologies
- Governance: The Open Group MSA Governance Reference Model

- Adapting reference architectures to the organization: the MSA Reference Architecture (in progress)

6 Phase A: Architecture Vision

The TOGAF Architecture Vision phase is concerned with establishing the architecture project and obtaining approval to proceed.

This phase captures the scope of the architectural initiative, which depends on the nature of the project and the level of detail of the implementation specification. It creates a compelling vision of what the organization will have at end-of-job, after all the projects necessary to instantiate the architecture have been completed. And it identifies the key stakeholders, concerns, and business requirements.

It must be noted that an MSA will only address one layer of an overall system or Enterprise Architecture. For example, the requirements for independence and self-containment which characterize a microservice mandate that any requirements for, for example, orchestration or choreography must take place at the application layer. Generally, an Enterprise Architecture will exist at a level substantially higher than the MSA.

At the enterprise level, we would expect that the Enterprise Architecture team would have developed a Strategic Architecture that gives a summary formal description of the enterprise, providing an organizing framework for operational and change activity, and an executive-level, long-term view for direction-setting. This might, for example, identify particular segments where microservices/cloud-native approaches should be used, and call for use of standards for interaction between segments, but it is highly unlikely to specify particular microservices or groups of microservices, or to prescribe a detailed infrastructure to support those microservices. The architect could then develop Segment Architectures, each of which gives a detailed, formal description of areas within an enterprise, used at the program or portfolio level to organize and align change activity. Below these Segment Architectures are the individual Capability Architectures, where the MSA layers would be expected to reside. This concept is depicted in Figure 5.

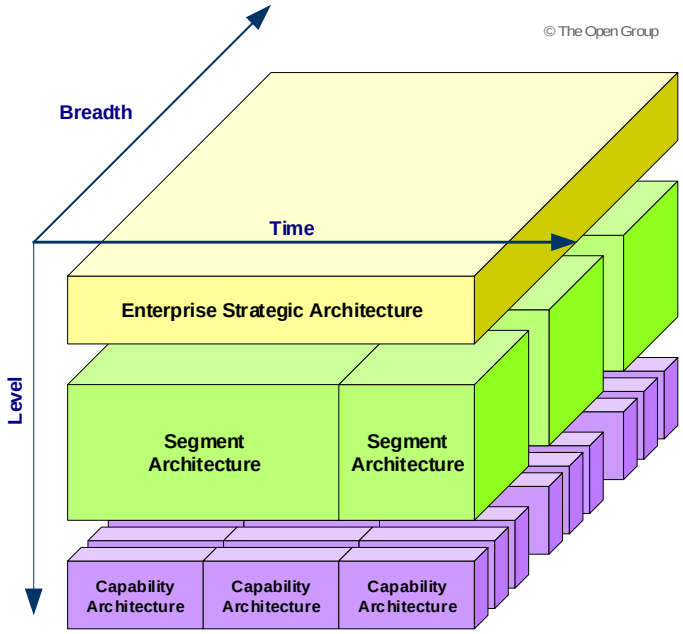

Figure 5: Scoping the Enterprise Architecture

6.1 The Nature of the Project

The scope of the MSA development depends on the size and structure of the project. The first key decision is to determine the focus of the architecture effort, by determining which business functions are to be implemented by the project.

The TOGAF Standard does not attempt to provide a definition of "project" as it does with "enterprise". However, it does note the increasing tendency for large-scale architectural developments to take the form of federated architectures, which are "independently developed, maintained, and managed architectures that are subsequently integrated within a meta-architectural framework". It is almost certainly true that it is impossible to have a single enterprise-level architecture that applies to every business function within the enterprise, unless that architecture itself is a federated one. The important implication is that each of those functional or project architectures must be consistent with the overall Enterprise Architecture, with overall governance and change management as discussed later in Chapter 12 and Chapter 13, respectively.

For the purposes of this document, we will assume that an overall Enterprise Architecture exists, of whatever complexity the enterprise requires, and concern ourselves with the development of the project-level architecture that is required for the particular system in question. Aside from general specifications of standards, interoperability requirements, and governance/change management, the Enterprise Architecture could also identify those particular business functions which could be potential applications for an MSA, although as an enterprise artifact it should not be specifying individual requirements for microservices or providing a detailed set of infrastructure requirements. The Enterprise Architecture may, for legacy applications, include additional constraints that apply to Capability Architectures, such as legacy interfaces or applications, as well as additional mandates that must be taken into account.

The development of the project architecture begins with a clear understanding of the business functions that are to be implemented via this architecture. It is at this point that the key business drivers which are mandating the use of an MSA should be enumerated: requirements for scalability, resiliency, failover, etc. The position of the MSA layer in the overall architecture must be made clear. What are the ABFs that are supplied by this layer? (The MSA Work Group understands an ABF to be the smallest unit of business function within an enterprise that delivers value, typically supporting a single business capability.) Interfaces where the microservice inputs and outputs are exchanged with other architectural layers should be identified. Governance and change management are inherited from the overall Enterprise Architecture. An example of a project architecture is shown in Figure 6.

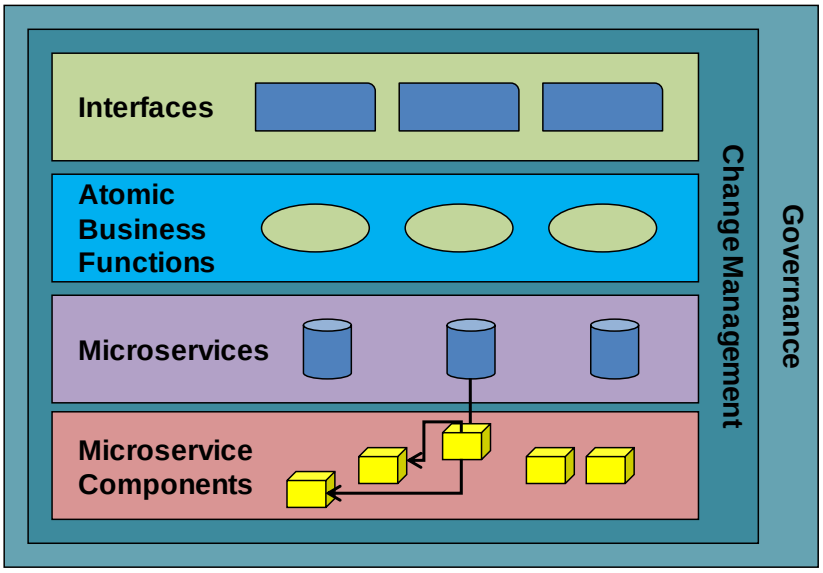

Figure 6: Project Architecture

This project-level architecture will be expanded into the Solution Architecture in the later phases of the architecture development process.

6.2 Level of Detail of Implementation Specification

How completely should the architecture define the implementation? In the case of a project-level architecture, it should be expected that the architecture will contain more detail than would typically be found in an enterprise-level architecture. This architecture is, after all, addressing the requirements of specific ABFs that need to be implemented. At the same time, it is not intended to be a Solution Architecture. Bear in mind that for a typically agile architectural paradigm such as MSA, frequent changes or modifications are to be expected; it is important that the architecture is not unnecessarily proscriptive in terms of, for example, technologies, although consistency across the enterprise and effective (albeit distributed) governance is still mandatory.

Solution project definition and planning is carried out in Phase E (Opportunities & Solutions) and Phase F (Migration Planning) of the TOGAF ADM, and the architecture team has a supervisory role for those projects in Phase G (Implementation Governance).

The Enterprise Architecture applies to each of the components of the enterprise and describes aspects that they have in common. Governance, change management, standards, and any

common required infrastructures can be defined in the Enterprise Architecture and inherited by each project architecture.

6.3 The Vision

The Architecture Vision includes a high-level description of the final architecture that is envisaged.

Like any SOA (of which MSA is a subset), the MSA description uses different language, with words such as "microservice", "composition", and "contract", and it has different models, such as matrices showing use of services by business processes and use of applications by services. The Microservice Ontology (https://webprotege.stanford.edu/ – requires registration and login) can provide taxonomical and ontological assistance with the language of a microservice-based architecture.

Although it may not include the kinds of detailed model produced in Phases B, C, and D, the high-level description produced in Phase A will reflect the service-oriented nature of the architecture that is envisaged.

6.4 Stakeholders, Concerns, and Business Requirements

Phase A is followed by the three TOGAF phases that produce detailed architecture descriptions for the Architecture Definition Document. In each of these phases, the architect:

- Develops models of the target system in the light of requirements
- Discusses concerns with stakeholders, using views of the system that are derived from the models
- Refines the models
- Identifies further requirements to be addressed

This is an iterative process, repeated until the architect is satisfied that the concerns relevant to the phase have been discussed and the requirements relevant to the phase are addressed.

For MSA, the individual business capability and functions that are being addressed should be defined in the context of the overall enterprise. This is particularly important when considerations of lifecycle and governance are addressed; these would only apply to changes to that particular business function. The degree of change should be fairly small after the initial Baseline Architecture has been created.

The requirements to address, the stakeholders to consult, and the models and views to develop vary from one architecture engagement to another. In Phase A of each engagement the architect identifies the key stakeholders and their concerns, states the key business requirements to be addressed, and considers which architecture views and viewpoints to develop.

7 Phase B: Business Architecture

An MSA is normally targeted to a single business function within a larger organization. Business functions are subsets of the enterprise that provide services and products that are consumed by customers and partners. These customers and partners may be wholly contained within the business. An enterprise has many, sometimes overlapping business functions. When a decision has been made to commit to an MSA for the business function (an organization within the larger enterprise), an architecture effort should be started to create the initial architecture for the function. The Business Architecture for the function needs to provide context for the business function within the enterprise as well as an understanding of the services and operations that the business function provides. This initial architecture serves as the baseline for future architecture efforts.

Business Function

According to the TOGAF Standard, a business function is:

"A collection of business behavior based on a chosen set of criteria, closely aligned to an organization."

Atomic Business Function

The MSA Work Group defines an Atomic Business Function (ABF) to be the smallest unit of business function within an enterprise that delivers value. An ABF typically provides services supporting a single enterprise capability. Business capabilities are delivered by processes carried out by business functions which may be composed of other business functions. As an example, most organizations have a business function for marketing. The marketing function may contain a customer loyalty business function within it. If the loyalty function does not contain any other business functions, it is an ABF providing services around the capability of managing customer loyalty relationships.

Once the initial MSA for the business function has been established, future iterations should be focused on the changes related to the business function supported by the MSA and the context of the function within the overall organization.

An MSA is not created in a vacuum, but is constrained by the overall Enterprise Architecture. This helps to reduce redundancies that may occur when multiple business functions share a need for common capabilities.

When developing a Business Architecture for an MSA project, the practitioner should limit the scope of the Business Architecture to that of the business function. Once the Business Architecture has been established for the business function, it can be re-used for future iterations.

7.1 Business Function Decomposition

A business function can be decomposed into its constituent parts in a similar fashion to how the enterprise is decomposed into a full Enterprise Architecture (refer to the TOGAF Standard, Phase B: Business Architecture).

- Business Capability Mapping

  Identifies, categorizes, and decomposes the capabilities required for the function to have the ability to deliver value to one or more stakeholders. This essentially defines the services, operations, products, and other outputs that a business function needs to deliver to achieve its purpose.

- Information Mapping

  Collecting the information concepts and their relationships that matter most to the business.

- Organization Mapping

  In an MSA Business Architecture there are two goals: the first is to establish the position and role of the business function within the enterprise, and the second is to describe the internal organization of the business function. Normally these can be established once for the business function and re-used for future iterations.

- Value Stream Mapping

  The value stream is the stream for the business function putting the stream in the context of value to the enterprise as a stakeholder. For an MSA the value stream artifacts illustrate how a function delivers value in the context of the enterprise or a containing subset of the enterprise. These artifacts must align with the overall Target Architecture.

- Structured and Use-Case Analysis

  In the context of the business function, identify the key services or operations provided and within the scope. Actors and their roles are identified and mapped to the delivery of services within the business function.

- Process Modeling

  The primary scope of process modeling here is for the processes contained within the business function. Interfaces to external processes should be modeled as black boxes, reflecting the independence of the services offered by the business function with respect to external services. These interfaces can be modeled as inputs, outputs, and controls which can be affected by external business functions.

In an MSA, the business function essentially becomes an independent component or building block of the enterprise. It is important to establish the interfaces that define how the business function interacts with other organizations. This includes defining the services and operations provided by the function, as well as the contracts and other documents that govern the interfaces.

7.2 Constraints

An MSA should be constrained by the Enterprise Architecture of the enterprise. Architects supporting the MSA must adhere to the Enterprise Architecture constraints and follow the governance of the Enterprise Architecture when there is conflict. However, independence from governance is a potential advantage of an MSA; therefore, it is recommended that the Enterprise Architecture organization practice flexibility with MSA organizations within the enterprise.

8 Phase C: Information Systems Architectures (Data and Application)

Phase C: Information Systems Architectures of the TOGAF ADM consists of two architecture domains for the project: the Data Architecture and the Application Architecture. In the context of an MSA, the applications generally consist of the services and operations exposed to service consumers and utilities used to maintain the services. Data is constrained to the data managed by the business function being served by the MSA and external data sources that may be used as an offline source or repository for such data.

In general, once the baseline Information Systems Architectures for the MSA supporting a business function have been established, future efforts will typically be focused on the changes required to achieve a new Target Architecture.

8.1 Data Architecture

The Data Architecture of an MSA is constrained by the data being collected and distributed by the business function supported by the MSA. This means that the relevant entities, attributes, and relationships are also constrained.

- Data Entity Catalog

  Provides brief descriptions of the domain data entities within the context of the business function.

- Data Entity to Business Function Mapping (Matrix)

  An MSA is focused on a single business function, but may feed data to other functions and may be provided with data that originates externally. This matrix shows the external feeds and providers used to maintain the MSA.

- Data to Application Mapping (Matrix)

  This table matches the data collected or exposed to the individual services and operations used for data access.

- Conceptual and Logical Data Diagrams

  These diagrams represent the standard conceptual (business-oriented, domain entities, and core relationships only) and logical (entities and significant fields) in the context of the business function supported by the MSA.

- Data Distribution Models and Diagrams

  Most data stores in an MSA will use one form of distributed data model or another. This diagram needs to capture the distribution model used for each data store within the MSA. In data stores using shards the sharding strategy should be included as part of the discussion of this diagram.

- Data Security Mapping

  Maps the security levels to data entities or attributes depending on the requirements. If entity mapping is used, the entity should be mapped based on the highest security requirement of any attribute contained within the entity.

- Data Lifecycle Mapping

  This maps data entities to the expected duration of the existence of their instances. This may be driven by retention policies.

- Data Migration Mapping

  Data migration typically covers movement of data between retiring systems and new systems. In the MSA context, there may also be migration between MSA live systems and batch systems that are updated periodically. When this occurs it is important to show those systems that participate in this type of batch transfer and the frequencies or triggers of the transfers.

8.2 Application Architecture

In general, the Application Architecture is about the applications used by a business organization to deliver the capabilities of that organization. An MSA is more constrained and is focused on the delivered capabilities of a business function within a larger organization. In this context, "application" typically refers to services and service operations; however, there may be tools and utilities like configuration editors and such that are used to support and maintain the underlying services and the supporting infrastructure.

The common artifacts for the Application Architecture and how they map to an MSA include:

- Application Portfolio Catalog

  In the MSA context, the Application Portfolio Catalog becomes the list of available services and operations exposed to external and internal consumers of those services.

- Interface Catalog

  Describes the SOA contract information required to understand, interact with, and consume the services exposed by the MSA. This may also cover the requirements for establishing credentials and service levels for the consumers of each operation.

- Application to Role Matrix

  This is a matrix describing the services and operation available to consumers acting in various roles.

- API to Consumer (Interaction) Mapping

  This maps the various service interfaces to classes of expected consumers or to individual consumers as needed. It should be noted that in some cases this might need to be implemented using a report based on actual and planned consumers of the services.

- Application Use-Case Diagrams

  A collection of use-cases and/or business scenarios giving context and describing how specific services and operations are intended to be used.

- Process/Application Realization Diagram

  Descriptions and process diagrams showing how the various services and operations are expected to operate.

- Application Deployment/Distribution Model

  An MSA will always be a distributed system. This model describes the distribution mechanism for the nodes of the MSA and should also cover when the numbers or nodes are expected to grow or shrink in response to demand or failover situations. In general, the TOGAF Standard considers this an extension to the normal Information Systems Architecture artifacts, but this is critical for an MSA.

9 Phase D: Technology Architecture

The decision to use microservices to provide some or all of the application functionality is a technology decision. It may be made in principle in the Architecture Vision phase. It is confirmed, its scope of application is decided, and its supporting Technology Architecture is specified in Phase D of the TOGAF ADM.

As described in the TOGAF Standard, the objectives of this phase are to:

- Develop the Target Technology Architecture that enables the Architecture Vision, target business, data, and application building blocks to be delivered through technology components and technology services, in a way that addresses the Statement of Architecture Work and stakeholder concerns
- Identify candidate Architecture Roadmap components based upon gaps between the Baseline and Target Technology Architectures

The entire architecture need not be based on microservices. Support for microservices may only determine part of the overall Technology Architecture.

9.1 Microservices

A microservice is a service that performs a single ABF and is independent of other microservices. In particular, it is developed independently, and is independently deployable.

As a service, a microservice has a defined service contract, which includes the specification of an interface, usually termed an API. The microservices available to other architecture components are listed in a service catalog.

The development, deployment, and execution of microservices is illustrated in Figure 7.

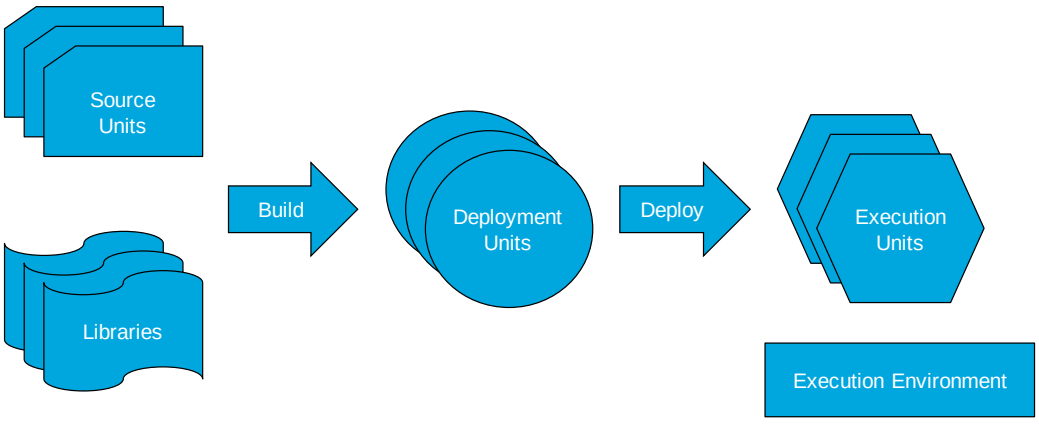

Figure 7: Development, Deployment, and Execution of Microservices

Each microservice has a source unit. This source unit contains the code that implements the microservice's ABF. It does not share code with other microservice source units, and cannot access their data. It may use libraries for functions that are not business-related. Each microservice also has a deployment unit that is built from its source unit and the libraries that the source unit uses. A deployment unit cannot include code for more than one microservice. Each microservice deployment unit is deployed as a single execution unit or, in a system at scale, multiple execution units running the same code.

9.2 Steps

Development of a Technology Architecture to support microservices typically requires the following additional steps in order to complete the steps described in the TOGAF Standard for Phase D (Technology Architecture):

- Define the scope of the application functionality described in the Application Architecture that will be supported by microservices
- Review the use of data by applications within the chosen scope, as described in the Data Architecture
- Review the requirements, with particular reference to requirements for transactionality and scalability
- Specify the principles and patterns that will apply to the use of microservices to support the applications and their use of data and to meet the requirements
- Specify the standards that will apply to service contracts and interface definitions, and to the service catalog
- Specify the microservice deployment mechanisms and execution environment

Independent development of microservices is a feature of MSA. Decisions about programming languages, libraries, development tools, and build mechanisms can be left to the microservice DevOps teams. These decisions may, however, be constrained to some extent by the Technology Architecture.

10 Phase E: Opportunities & Solutions

10.1 Overview

The TOGAF Opportunities & Solutions phase identifies delivery vehicles (projects, programs, or portfolios) that effectively deliver the Target Architecture defined in previous phases. It reviews the target business objectives and capabilities, consolidates the gaps from Phases B to D, and organizes groups of building blocks to address these capabilities. It then generates an outline Implementation and Migration Strategy.

There are some nuances between a cloud-native solution and a microservices-based solution. Microservices are a type of cloud-native solution, but they can have non-cloud deployments as well. Cloud-native is exclusively designed for the cloud. The intent of cloud-native is to optimally use cloud-scale architecture and cloud services in the solution. Microservices have other constraints and motivations; refer to The Open Group White Paper: Microservices Architecture (see Referenced Documents) for a deeper understanding. The commonalities are that they are both very modular and highly distributed deployments. They are elastically scalable and resilient.

Cloud-native and MSA, as explored in depth in other chapters, bring the following key considerations to the Target Architecture, the Architecture Roadmap, and the Transition Plan:

- Scope transformation
- Organizational transformation
- Technology transformation

  — Data Transformation

10.2 Scope Transformation

A key tenet of cloud-native and MSA is that its target state is a modular, distributed, and highly decoupled solution. The current state may be a Greenfield scenario or a decomposable monolith solution. Microservices are independent and self-contained. This characteristic allows for the scope of each module to be managed independently, with its own Roadmap and Transition Plan.

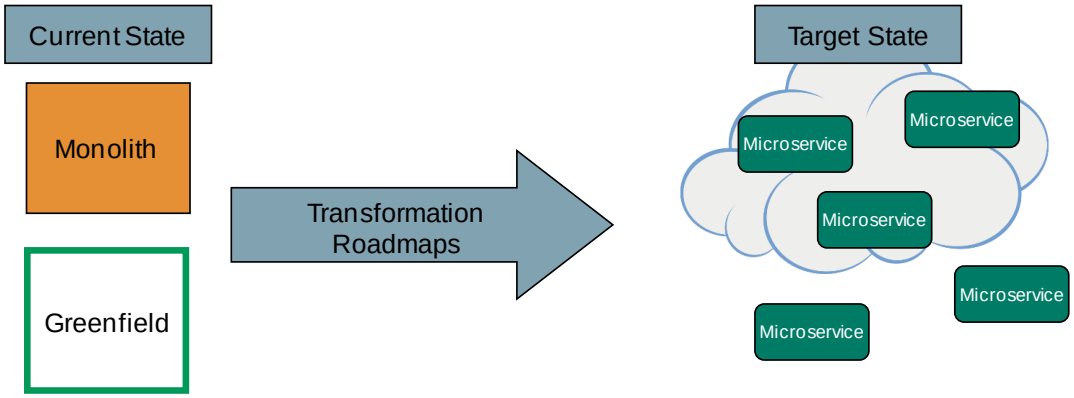

Figure 8: Scope Transformation

10.3 Organizational Transformation

To successfully embrace cloud-native and MSA requires significant organizational transformation. The target state demands single team ownership of each module (microservice) and a lean, federated governance structure. Each team will autonomously own the entire lifecycle of the microservice, and there is a need for coordination to enable all capabilities of the desired target state solution.

This new paradigm presents a good opportunity for adopting an Agile/DevOps methodology; see The Open Group White Paper: Benefits of DevOps Methodology for Microservices Solutions (see Referenced Documents).

10.4 Technology Transformation

Cloud-native and microservice solutions adopt a polyglot paradigm. This opportunity introduces a heterogeneous technology landscape in the target state. A further complexity is that most of the distributed modular solutions have a dual reference architecture: one for the module or microservices (sometimes called inner architecture), and the other for the framework (*aka* outer architecture) in which it is deployed. Each of these has their unique characteristics and constraints. Hence the technology transformation plans and roadmaps need to account for the following:

- Transformation from (nearly) homogeneous to a heterogeneous technology landscape
- Transformation from a single reference architecture to a dual reference architecture
- Delivery and deployment infrastructure transformation

10.4.1 Data Transformation

Proper data transformation underpins a successful microservices solution. Persistent data storage will be part of the framework (outer architecture), but needs to be similarly distributed, elastically scalable, and resilient as the microservices themselves, to achieve the desired benefits of an MSA. This phase should define the transformations necessary in the data layer of the architecture to meet the needs listed above.

10.5 Artifacts

As with the previous phases, there are several models, artifacts, and guidelines that are cloud-native and microservice-specific. Those for Phase E might include those listed below.

Artifact	Purpose
Application Decomposition Methodology	This is the detailed document explaining the methodology (Domain-Driven Design (DDD), etc.) and how it is used to decompose the current state solution to the target state.
Technology Guidelines	This document provides the guidelines on how to use cloud-native and microservices infrastructure.

11 Phase F: Migration Planning

The objectives of TOGAF Phase F are to:

- Finalize the Architecture Roadmap and the supporting Implementation and Migration Plan
- Ensure that the Implementation and Migration Plan is co-ordinated with the enterprise's approach to managing and implementing change in the enterprise's overall change portfolio
- Ensure that the business value and cost of work packages and Transition Architectures is understood by key stakeholders

The Architecture Roadmap lists individual work packages that will realize the Target Architecture and lays them out on a timeline to show progression from the Baseline Architecture to the Target Architecture. The Architecture Roadmap highlights the business value of individual work packages at each stage. Transition Architectures necessary to effectively realize the Target Architecture are identified as intermediate steps. The Architecture Roadmap is incrementally developed throughout Phases E and F, and informed by readily identifiable roadmap components from Phase B, C, and D within the ADM.

The Implementation and Migration Plan provides a schedule of the projects that will realize the Target Architecture. The Implementation and Migration Plan includes executable projects grouped into managed portfolios and programs. The Implementation and Migration Strategy identifying the approach to change is a key element of the Implementation and Migration Plan.

11.1 Microservices Work Packages

Work packages or projects related to microservices that could appear in the Implementation and Migration Plan include:

- Creation of an environment in which microservices can be developed and deployed, and can execute
- Formation of development and operations teams, or DevOps teams, that will be responsible for one or more microservices
- Development of a microservice
- Operation of a microservice
- Modification of non-microservice-based systems to interface with microservices

A microservice implements an ABF. A business capability could be provided by a single microservice or a group of related microservices.

Formation of a development team, formation of an operations team, microservice development, microservice operation, and modification of other systems to work with microservices would

likely be specified as part of a Capability Architecture, which provides an organizing framework for change activity and the development of effective Architecture Roadmaps realizing capability increments.

Creation of a microservices environment might be specified as part of a Capability Architecture or as part of a Segment Architecture, which provides an organizing framework for operational and change activity and allows for direction-setting and the development of effective Architecture Roadmaps at a program or portfolio level.

11.2 Intentional and Emergent Architecture

The Open Agile Architecture™ Standard, also known as the O-AA™ Standard (see Referenced Documents) views architecture development as a combination of intentional and emergent architecture. The description of Phase F in the TOGAF Standard is written using the language of intentional architecture, but the TOGAF framework can be used for agile architecture developments, where there is a greater focus on emergent architecture. This will often be the case for cloud-native and MSA.

Creation of a microservices environment and formation of a DevOps team are activities that would normally be identified as work packages that appear in an Implementation and Migration Plan developed in Phase F, using an intentional approach. Microservice development and modification of other systems might also be intentionally planned activities. Alternatively, they might be emergent activities, with microservices created to meet business needs without being scheduled on the Implementation and Migration Plan. The guardrails defined for an agile development, supported by Enterprise Architecture governance, should make it clear whether this is allowed.

11.3 Value and Cost Reporting

The architecture team should maintain up-to-date catalogs of microservices that are implemented and microservices that are planned. A microservice implements an ABF, and the business functionality and value of each microservice should be clearly presented in its catalog description.

The architecture team should also maintain records of the resources allocated and planned to be allocated to the development and operation of each microservice.

These catalogs and records will ensure that the business value and cost of work packages and Transition Architectures can be understood by key stakeholders.

12 Phase G: Implementation Governance

A key activity in delivering any type of system is governing the implementation. This is how the business can be assured that what is delivered has not diverged inappropriately from what was requested and architected. The objectives of TOGAF Phase G (Implementation Governance) are to:

- Ensure conformance with the Target Architecture by implementation projects
- Perform appropriate Architecture Governance functions for the solution and any implementation-driven architecture Change Requests

It is important to note that the entire architecture of a solution which consumes microservices will almost certainly not be based solely on those microservices. Support for microservices may only determine part of the overall Technology Architecture. This reinforces the need for governance, so that those Solution Architectures can depend upon the microservices.

12.1 Microservices Defined

For the purpose of discussing governance, it is important to remember that a microservice is a service that performs a single ABF and is independent of other microservices. In particular, it is developed independently, and is independently deployable.

As a service, a microservice has a defined service contract, which includes the specification of an interface, usually termed an API. The microservices available to other architecture components are listed in a service catalog.

12.2 Governance Approach

This document draws heavily on the TOGAF Standard. Specifically, Phase G (Implementation Governance) describes the overall approach as follows:

- Establish an implementation program that will enable the delivery of the Transition Architectures agreed for implementation during the Migration Planning phase
- Adopt a phased deployment schedule that reflects the business priorities embodied in the Architecture Roadmap
- Follow the organization's standard for corporate, IT, and architecture governance
- Use the organization's established portfolio/program management approach, where this exists
- Define an operations framework to ensure the effective long life of the deployed solution

It is also important to ensure that arbitrary choices are not made by the various DevOps teams creating microservices, leading to technology divergence and the attendant costs and risks

associated with having too many different approaches and tools. That said, it is also important to recognize that different problems may require different tools for their solutions. The point of governance is to ensure that these differences are resolved in a reasonable, managed fashion and not arbitrarily.

Consequently, the governance team will need to involve itself in such decisions, at least at the oversight level.

One implication of having a proper governance process is the need for solutions to be able to consume existing services. In turn, this means that it must be possible to know just what services exist in the organization and their provenance(s). It is also important to know what expectations can be had about their use, in particular non-functional expectations in relation to their performance, availability, and security. This may be done using a service catalog product. There are several commercial products in this space but other approaches, including manual cataloging, can also be used. Another issue will be the ability to dynamically bind to/invoke services which will mean their addresses need to be known and stable. The ability to consume microservices across the organization will also require that the authentication and authorization processes are understandable.

13 Phase H: Architecture Change Management

There are two focus areas for Phase H (Architecture Change Management) as an organization transitions to and employs microservices. Those are changes in the overall, organizational architecture processes and then, at the solution level, change management for an approved Solution Architecture.

13.1 Organizational Architecture Processes

As an organization moves toward use of microservices as a preferred architectural style, this will require changes to the overall architecture process. Issues will include:

- Reviewing Solution Architectures for opportunities to use microservices

  — And, conversely, inappropriate use of microservices

- Reviewing the organization's catalog of existing microservices, microservices under development, and retired microservices
- Ensuring that microservices in development plan for use beyond the initial solution for which they are intended
- Defining the authentication and authorization approach to be used and ensuring that it is in harmony with the overall organization's authentication and authorization methodology and tooling
- Ensuring that a service contract is created for each service being developed, including all the non-functional characteristics for the service, plus expectations upon the infrastructure for the service

The TOGAF Standard discusses organizational architecture management in Phase H.

13.2 Solution Architecture

The architecture of a solution which consumes or provides microservices must consider that other organizational solutions might and, in fact, are likely to consume or provide microservices. This means that care must be taken in a number of areas, including:

- Explicitly documenting the characteristics, including the non-functional characteristics, of a given microservice; these should be documented in the form of a service contract, to the standards of the organization
- Not arbitrarily changing a microservice, given that other parties are likely to depend upon that microservice; this will require understanding who those other parties are and communicating with them about changes

14 Summary

In summary, there are a number of methods, tools, and reference materials available to help the Enterprise Architect develop Microservices Architecture (MSA) for distributed applications. The Open Group standards and publications are suggested. Some are directly focused on SOAs and the readiness of an organization to adopt them, such as OSIMM; others are not directly focused but regularly useful, such as outputs of The Open Group Security Forum.

The adoption of an architecture capability to support an MSA requires considerable activity in the TOGAF Preliminary Phase. These activities and service-specific Work Group tools include:

- Adapting the principles of MSA for appropriate levels of Enterprise Architecture practices – recognize that an MSA will be a significant part of the Enterprise Architecture, but will never be a complete Enterprise Architecture in itself
- Determining organizational readiness for service-orientation (OSIMM)
- Governance: adapting The Open Group SOA Governance Vitality Method (SGVM), recognizing that governance of an MSA in a distributed environment requires distributed, federated governance, as detailed in this document
- Organization: ensuring that the organization is skilled, arranged, and governed in such a way as to support distributed MSA applications; compatible practices such as DevOps are suggested

The changes in the rest of the TOGAF ADM phases cover how an architecture is described, analyzed, and documented. During an iteration of the ADM the practitioner needs to consider the relationship between the MSA layers and the overall enterprise, and the artifacts necessary to document the interfaces between them.

At different levels of granularity the purpose of the ADM cycle will vary. In strategic-level work, the purpose is identifying whether MSA is needed, and in which segments. In segment-level work, describing the structure and capability requirements of the MSA takes place. Finally, in capability-level work, the purpose is to identify and describe the requirements of the MSA services that will be implemented.

When delivering MSA with the TOGAF framework, the practitioner should never lose sight of the final objective: MSA solutions that address management of the enterprise's complexity and provide business agility in efficiently meeting the needs of specific business capabilities.

Acronyms & Abbreviations

ABB	Architecture Building Block
ABF	Atomic Business Function
ADM	Architecture Development Method
API	Application Program Interface
DDD	Domain-Driven Design
IoT	Internet of Things
IT	Information Technology
KPI	Key Performance Indicator
MSA	Microservices Architecture
O-AA	Open Agile Architecture
OAS	OpenAPI Specifications
OSIMM	The Open Group Service Integration Maturity Model
SaaS	Software as a Service
SGVM	SOA Governance Vitality Method
SOA	Service-Oriented Architecture

Index