Enabling Enterprise Agility

The Open Group TOGAF® Series Guide

Table of Contents

Enabling Enterprise Agility

The Open Group TOGAF® Series Guide

The Open Group TOGAF® Series Guide
Enabling Enterprise Agility
Document Number: G20F
ISBN: 1-947754-71-3

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

Built with asciidoctor, version 2.0.16. Backend: pdf Build date: 2022-04-19 14:56:14 UTC

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 a TOGAF® Series Guide: Enabling Enterprise Agility.

It has been developed and approved by The Open Group.

The high-level structure of this document is summarized as follows:

- Chapter 1 provides an introduction to this document, including what is meant by agility, the role of Enterprise Architecture, and its relation to agility
- Chapter 2 includes the terms and definitions for this document

- Chapter 3 describes the TOGAF Architecture Development Method (ADM) and how that relates to agility
- Chapter 4 looks at how architecture activities can be structured to support agility
- Chapter 5 considers how to execute Enterprise Architecture in an Agile environment

The audience for this document is Enterprise Architects requiring information on how to adapt and use the TOGAF framework to support an Agile enterprise.

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.

About the Author

See the Acknowledgements section for more information on contributions to this document. Please note that affiliations were current at the time of approval.

Christopher Frost, Fujitsu

Chris is a Principal Enterprise Architect in Fujitsu, working in the Application Technology Consulting Division. The Division provides guidelines, standards, and expert technical support for the global Fujitsu Group. Chris has worked for Fujitsu since 2005, in a variety of technical leadership roles, including CTO for various large business units. Before Fujitsu, Chris worked for EDS (now part of DXC) on several large contracts for the UK Ministry of Defence, and in earlier years worked for Ford, Shell, and a small startup software house called Shamrock Marketing. He is the lead author of this Guide.

Acknowledgments

The Open Group gratefully acknowledges the contribution of the following people in the development of this document (please note that affiliations were current at the time of approval):

- Michael Anniss, M. J. Anniss Ltd.
- David Gilmour, Mundo Cognito Ltd.
- Sonia Gonzalez, The Open Group
- Richard Gornitsky, Full-Stack Architecture International
- Mike Lambert, Fellow of The Open Group
- Venugopal Gomatham Venkata Narasimha, Nationwide
- Donald (“Troy”) Peterson, Nationwide
- Miroslaw Prywata, Asseco Academy
- Aaron Rorstrom, Capgemini
- Jason R. Sinclair, Nationwide

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

- Rolf Knoll, NovaTec Consulting GmbH
- Frederic Le, DXC
- Boitumelo Molete, University of the Witwatersrand
- Jim Rhyne, Business Architecture Guild

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.

DSDM is a registered trademark of Agile Business Consortium Limited.

Java is a trademark of Oracle America, Inc.

PRINCE is a registered trademark of AXELOS Limited.

SAFe is a registered trademark of Scaled Agile, Inc.

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.

Referenced Documents

The following documents are referenced in this TOGAF® Series Guide.

(Please note that the links below are good at the time of writing but cannot be guaranteed for the future.)

[1] *The TOGAF® Standard, 10th Edition, a standard of The Open Group (C220)*, April 2022, published by The Open Group; refer to: www.opengroup.org/library/c220

[2] *Agile Manifesto*, 2001, by Kent Beck, et al.; refer to agilemanifesto.org

[3] *Capability-Based Planning Supporting Project/Portfolio and Digital Capabilities Mapping Using the TOGAF® and ArchiMate® Standards (G193)*, July 2019, published by The Open Group; refer to: www.opengroup.org/library/g193

[4] *TOGAF® Series Guide: Applying the TOGAF ADM using Agile Sprints (G210)*, April 2022, published by The Open Group; refer to: www.opengroup.org/library/g210

[5] *Set-Based Concurrent Engineering in the Lean Lexicon*; published by the Lean Enterprise Institute; refer to: www.lean.org/lexicon/set-based-concurrent-engineering

[6] *Open Agile Architecture™, also known as the O-AA™ Standard, a standard of The Open Group (C208)*, September 2020, published by The Open Group; refer to: www.opengroup.org/library/c208

[7] *TOGAF® Series Guide Set: Business Architecture (T190)*, April 2022, published by The Open Group; refer to: www.opengroup.org/library/t190

[8] *TOGAF® Series Guide: Business Models (G18A)*, April 2022, published by The Open Group; refer to: www.opengroup.org/library/g18a

[9] *Digital Practitioner Body of Knowledge™ Standard, also known as the DPBoK™ Standard, a standard of The Open Group (C196)*, January 2020, published by The Open Group; refer to: www.opengroup.org/library/c196

[10] *TOGAF® Series Guide: Organization Mapping (G206)*, April 2022, published by The Open Group; refer to: www.opengroup.org/library/g206

[11] *TOGAF® Series Guide: The TOGAF® Leader's Guide to Establishing and Evolving an EA Capability (G184)*, April 2022, published by The Open Group; refer to: www.opengroup.org/library/g184

[12] *TOGAF® Series Guide: A Practitioners' Approach to Developing Enterprise Architecture Following the TOGAF® ADM (G186)*, April 2022, published by The Open Group; refer to www.opengroup.org/library/g186

Chapter 1. Introduction

This document describes in general terms how the TOGAF® Standard [1] can be adapted to support an "Agile enterprise". It is written to be applicable to any Agile delivery method that follows the commonly accepted Agile approach of iterative development through a series of sprints. It will be supported by other standards, guides, white papers, and case studies from The Open Group, which will provide more detail about specific approaches.

1.1. What is Meant by Agility and Why is it Important?

Enterprise agility is a commonly used term but the exact definition differs among practitioners. The most common characteristics include:

- **Responsiveness to change**: a flexible approach that anticipates and explicitly plans for change, typically involving short iterations and the frequent reprioritization of activities
- **Value-driven**: activity is driven by delivering value; priorities are continually re-assessed to deliver high-value items first and work on intermediate products and documentation is minimized
- **Practical experimentation**: a preference for trying things out and learning from experience as opposed to extensive theoretical analysis, sometimes characterized as "fail fast"
- **Empowered, autonomous teams**: skilled, multi-disciplinary teams work closely together, taking responsibility for their own decisions and outputs
- **Customer communication and collaboration**: working closely with the customer and adapting to their needs; valuing collaboration and feedback over formalized documentation and contracts
- **Continuous improvement**: the internal drive to improve the way an organization performs
- **Respect for people**: people are put first, above process and tools – they are treated with respect; flexibility, knowledge transfer, and personal development are high priorities

Regardless of how the term enterprise agility is defined, it is important because it enables an enterprise to better react to change by being more customer and product-centric, more efficient, and better able to ensure regulatory compliance.

The term Agile is typically associated with the Agile Software Development Process derived from the Manifesto for Agile Software Development [2]. While Agile principles and techniques can be applied to adapt the TOGAF framework, enterprise agility is a broader context than Agile. Therefore, additional techniques are employed in adapting the TOGAF framework to an Agile enterprise.

1.2. What is the Role of Enterprise Architecture?

Enterprise Architecture provides a framework for change, linked to both strategic direction and business value. It provides a sufficient view of the organization to manage complexity, support continuous change, and manage the risk of unanticipated consequences.

Enterprise Architecture is:

- A description of the elements within an organization, what they are meant to achieve, how they are arranged, how they perform in practice, and how they respond to change
- A framework (structure, approach, and process) for managing change to those elements and their arrangement; to continuously adapt to organizational change in line with strategy (goals and objectives) and circumstances (specific requirements)
- The practice of acting to manage and evolve the Enterprise Architecture at all levels of control, change, and pace

The TOGAF Standard is a framework for identifying and implementing change, and provides:

- A definition and description of a standard cycle of change, used to plan, develop, implement, govern, change, and sustain an architecture for an enterprise; see the TOGAF Architecture Development Method (ADM)
- A definition and description of the building blocks in an enterprise used to deliver business services and information systems (see the TOGAF Content Framework)
- A set of guidelines, techniques, and advice to create and maintain an effective Enterprise Architecture and deliver change through new Solution Architectures at all levels of scale, pace, and detail

1.3. The Demand for Agility is Not New!

The US Clinger-Cohen Act of 1996 was a major driver for the adoption of Enterprise Architecture because it required investment in new IT systems (in the US public sector) to demonstrate fit with existing systems. Almost immediately, it was recognized that it is not practical to develop a full Enterprise Architecture from the “top down”; it is necessary to structure work into smaller units to respond to the specific needs of the enterprise in a timely manner, and to have some form of “integration architecture” to ensure the pieces fit together and are aligned with the strategic direction of the enterprise.

The TOGAF framework has embraced the call to respond to the needs of the enterprise in a timely manner, through the concepts of “partitions” and “levels”. Partitions define how the work is broken down into multiple architecture initiatives. Levels define how the overall architecture can be developed at different levels of granularity and detail.

Each architecture initiative needs to be scoped to address the specific needs of the enterprise to be addressed.

The major scoping dimensions are:

- Breadth (subject matter): what is the full extent of the enterprise, and to what extent will this architecting effort deal with it?
- Depth (level of detail): to what level of detail should the architecting effort go?
- Time period: what is the time period that needs to be articulated for the Architecture Vision?

- Architecture domains: a complete Enterprise Architecture description should contain all four architecture domains (Business, Data, Application, Technology), but the realities of resource and time constraints often mean there is not enough time, funding, resources, or need to address them all

The TOGAF Standard allows users to use the TOGAF ADM in an iterative way. The ADM can be used to deliver value incrementally following different iteration approaches. The concept of iteration is deep-rooted within the TOGAF ADM; as described in the TOGAF Standard – Applying the ADM (Applying Iteration to the ADM) [1].

Iterative development, such as Agile practices like sprints with shorter iterations, is a useful technique to obtain early stakeholder feedback and results. It enables the Enterprise Architects to deliver value earlier and iteratively, whether in a planned or emergent manner.

Dividing work into sprints does not only mean dividing work into small pieces, but also learning by doing in short cycles and adapting the work accordingly.

1.4. How Does that Relate to Today's Imperative for Increased Agility?

There is an ongoing demand for the size and timescale of architecture segments and increments of capability to become ever shorter. This in turn is resulting in a tendency for enterprises to skip the development of Strategic and Segment Architectures, which in turn is resulting in some high-profile IT failures because of the unanticipated consequences of what appeared to be minor changes.

The TOGAF Standard recognizes the need to recursively break down the Enterprise Architecture into more granular levels that can be architected following an Agile approach.

Partitioning the architecture work is key for Agile delivery and implies the definition of segmentation based on the Strategic and Segment Architecture concepts as explained before.

These smaller pieces, that cover a specific area of the organization, can then be more easily specified and implemented following an Agile approach, enabling a cross-cutting view across the TOGAF domains.

The first step is to identify the segments scoped to address specific needs keeping the alignment with the Strategic Architecture. Once these segments have been identified they can be refined further into Capability Architectures that can be specified and implemented following an Agile approach.

These Capability Architectures, delivered in increments, can be used to iteratively construct the final outcome (product, service, or solution). These iterations can be conducted in parallel depending on the level of interdependency between them. And, as stated in the TOGAF Standard – Applying the ADM (Applying Iteration to the ADM), Capability Architectures corresponding to a Segment Architecture can also be performed in parallel.

The following chapters will explain these concepts in more detail; see in particular Section 4.1.

Chapter 2. Definitions

For the purposes of this document, the following terms and definitions apply. The definitions within the TOGAF Standard or the Merriam-Webster's Collegiate Dictionary should be referenced for terms not defined in this section.

2.1. Agile

To move/change quickly and easily, often to provide value-generating outcomes.

2.2. Agile Architecture

1. The "act" – the development of architecture that reacts quickly and easily to changes through the delivery of iterative architectures that provides incremental value-generating outcomes.
2. The "thing" – an architecture that is flexible; i.e., easy to change or adapt.

2.3. Agile Product Owner

A member of an Agile product team responsible for defining user stories and prioritizing the backlog, ensuring these are understood by other team members while maintaining the conceptual integrity of the features or components for the delivery team. In the TOGAF framework, product has a wider context, but is used here in the Agile product context.

2.4. Minimum Viable Architecture

An architecture that enables the delivery of product features with just enough content to be deployed in a given phase of a project and satisfies known requirements (especially quality attribute requirements), and no more.

2.5. Minimum Viable Product

The smallest possible outcome that generates acceptable learning, delivery of value to the customer (internal or external), and is a basis for future extension.

2.6. Product

An outcome generated by the business to be offered to customers. Products include materials and/or services.

Chapter 3. Overview of the TOGAF Architecture Development Method

The core of the TOGAF framework is the TOGAF ADM. At first sight, the iconic graphic of the TOGAF ADM reinforces the perception that Enterprise Architecture is a lengthy, “waterfall” process; see Figure 1.

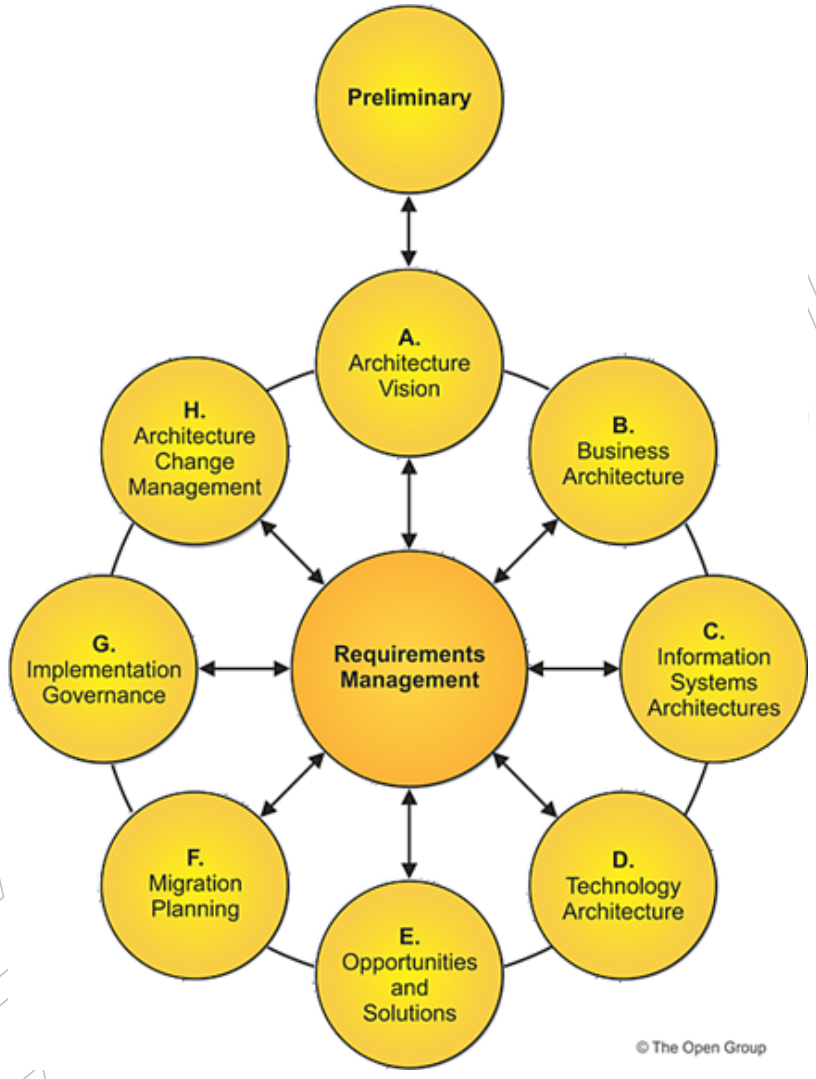

Figure 1. The TOGAF ADM

However, it is important not to infer statements from the TOGAF framework that are not present. The TOGAF ADM does not:

- Mandate that the steps must be performed in the sequence shown
- Mandate a “waterfall” process; i.e., that each phase must complete before the next begins
- Specify the duration of any phase or cycle of architecture development

The TOGAF framework *does* recommend that the ADM be adapted to meet the needs of the enterprise; agility is one such need.

The TOGAF framework shows how the ADM provides a tested and repeatable process for developing architectures. Rather than viewing the ADM graphic as a process model, it is helpful to view it as a reference model, which defines what has to be done in order to deliver solutions with a rational structure and to identify the interactions and relationships between components across the enterprise. Understanding these interactions and relationships is critical to reducing risk and optimizing the approach.

The core concepts of Enterprise Architecture shown in the ADM graphic are applicable in the most Agile of environments. Phases A-H around the circumference show how architecture is progressively developed and applied to the downstream delivery activities.

Fundamentally, the TOGAF Standard supports what architects do – they understand, specify, and govern. The phases of the ADM are:

- Understand
  - Phase A – Architecture Vision: understand the problem/opportunity, sketch the solution, and identify the broad transition approach
  - Phases B-D – Business/Information Systems/Technology Architecture: identify what is needed (Architecture Building Blocks (ABBs))

    During these phases, a recommended practice is to identify the potential solution implementations (Solution Building Blocks (SBBs))
- Specify
  - Phase E – Opportunities and Solutions: select from the candidate set of SBBs to best fit with the ABBs of Phases B to D and how they will interoperate to deliver the business service levels required, and the most appropriate implementation transitions
  - Phase F – Migration Planning: organize the resources to deliver the transitions in a controlled fashion
- Govern
  - Phase G – Implementation Governance: ensure the reuse/build/acquisition and deployment activities are properly organized and deployed in line with the agreed contract and specifications
  - Phase H – Architecture Change Management: ensure that the change is properly planned, structured, and delivers the business value that is expected

In the next chapter, more detailed guidance is given about how the ADM and other parts of the TOGAF framework can be applied in Agile environments.

Chapter 4. Developing Architecture in an Agile Way

4.1. Different Levels of Detail Enable Agility

The TOGAF framework presents a model identifying three levels of detail that can be used for partitioning architecture development:

- Enterprise Strategic Architecture
- Segment Architecture
- Capability Architecture

These levels are illustrated in Figure 2.

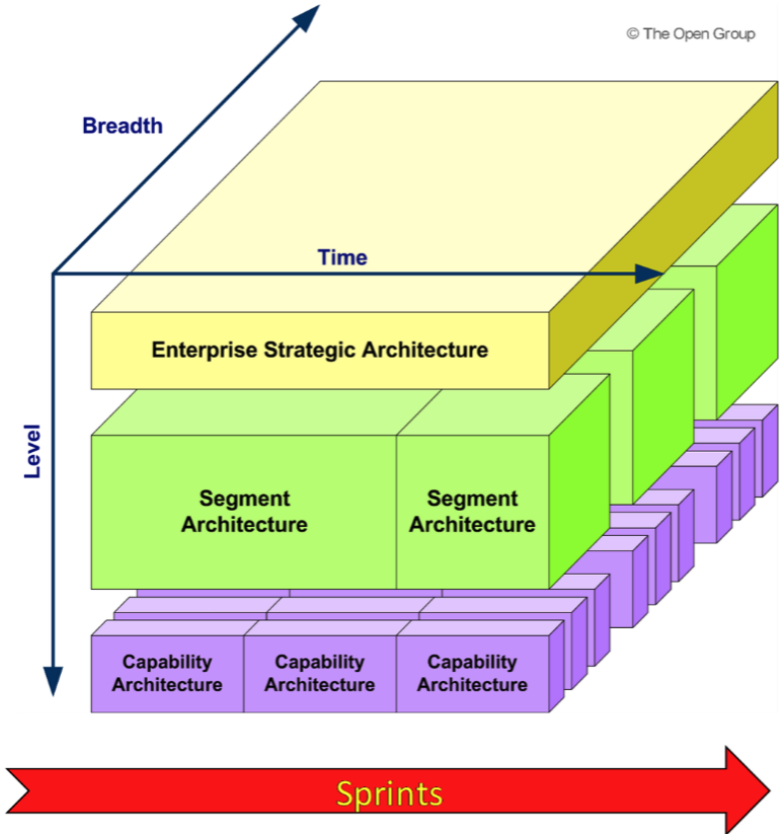

Figure 2. Summary Classification Model for Architecture Landscapes

This model must not be confused with the architecture domains; Business, Data, Application, and Technology Architectures may exist at all levels of detail.

Top-down, the *Enterprise Strategic Architecture* provides a high-level view of the area of the enterprise impacted by change. It enables understanding of the overall strategic direction of the enterprise at a high level, and must be sufficiently broad to establish the context within which the segments and

capabilities fit. It is necessary to plan and design the entire endeavor, and to avoid unanticipated consequences.

The middle layer, the *Segment Architectures*, typically provide direction at the portfolio, program, or product level. These large-scale segments are often aligned to natural boundaries of functionality.

The bottom layer, the *Capability Architectures*, are detailed descriptions of (increments of) business capabilities [3]. These may align to delivery sprints, or multiple sprints may be needed to deliver a capability. They are sufficiently detailed to be handed to developers for action. Sprints may occur at any level, but are most commonly associated with the delivery of capabilities or increments of capability. Sprints can occur in parallel.

A key consideration is that sprints are time-boxed and aimed at addressing a set of bounded objectives. The Capability Architecture increments should be tightly scoped to be achievable within sprint time boxes. More detailed guidance relating to sprints can be found in the TOGAF® Series Guide: Applying the TOGAF ADM using Agile Sprints [4].

The higher-level Enterprise Strategic and Segment Architectures should show the relationships and dependencies between capabilities and capability increments and provide the framework for planning and design, and the management of risk. They then provide the information needed to assess the impact of a proposed change. The Capability Architectures then show the relationship between capability increments.

Bottom-up, there is feedback from the implementation of capability increments which influences the higher levels. The Capability, Segment, and Enterprise Strategic Architectures may evolve as a result of experience gained from the deployment of each capability (or capability increment).

The Strategic Architecture is not static. It must evolve as the strategy of the enterprise evolves. In Agile enterprises, this will be more frequent than a “traditional” long-term strategic business plan.

It is vital to have appropriate governance to maintain the link between the business needs, Enterprise Architectures, and the Agile solution developments of the enterprise.

Two major factors to achieving successful agility at an enterprise level are:

1. Managing the scope, understanding when a new capability is needed, how much of the enterprise is impacted, and how different parts of the enterprise interact.
2. Having a sufficient understanding of the overall strategic direction of the enterprise, key business capabilities, and the relationships between them in order to minimize the risks of unanticipated consequences and piecemeal development, and identify any change which would detract from the overall strategy for the enterprise. This understanding facilitates an impact assessment of any proposed change.

4.2. Transition Architectures

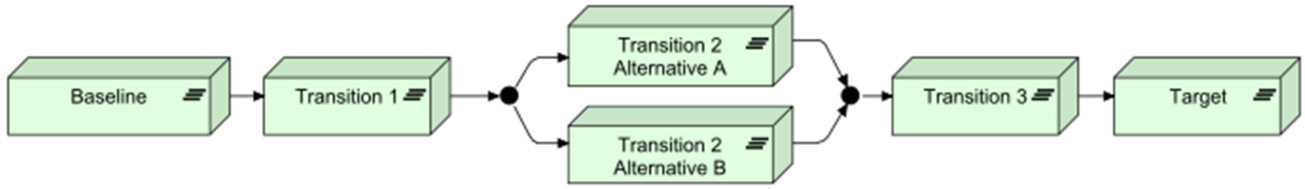

Figure 3. Transition Architectures

A *Transition Architecture* is a formal description of one state of the architecture at an architecturally significant point in time (see Figure 3), usually including a number of capability increments. It describes the roadmap to the desired outcome and ensures the stability of the complete system after its implementation. In Agile, the capability increments are usually implemented using sprints. Transition Architectures also provide a way of managing risk by helping to understand the incremental states of delivery. Transition Architectures are related to releases of business value to the stakeholders.

4.3. A Practical Approach to Structuring Agile Enterprise Architecture

A practical and proven approach to applying the multi-level approach through a hierarchy of ADM cycles is illustrated in Figure 4.

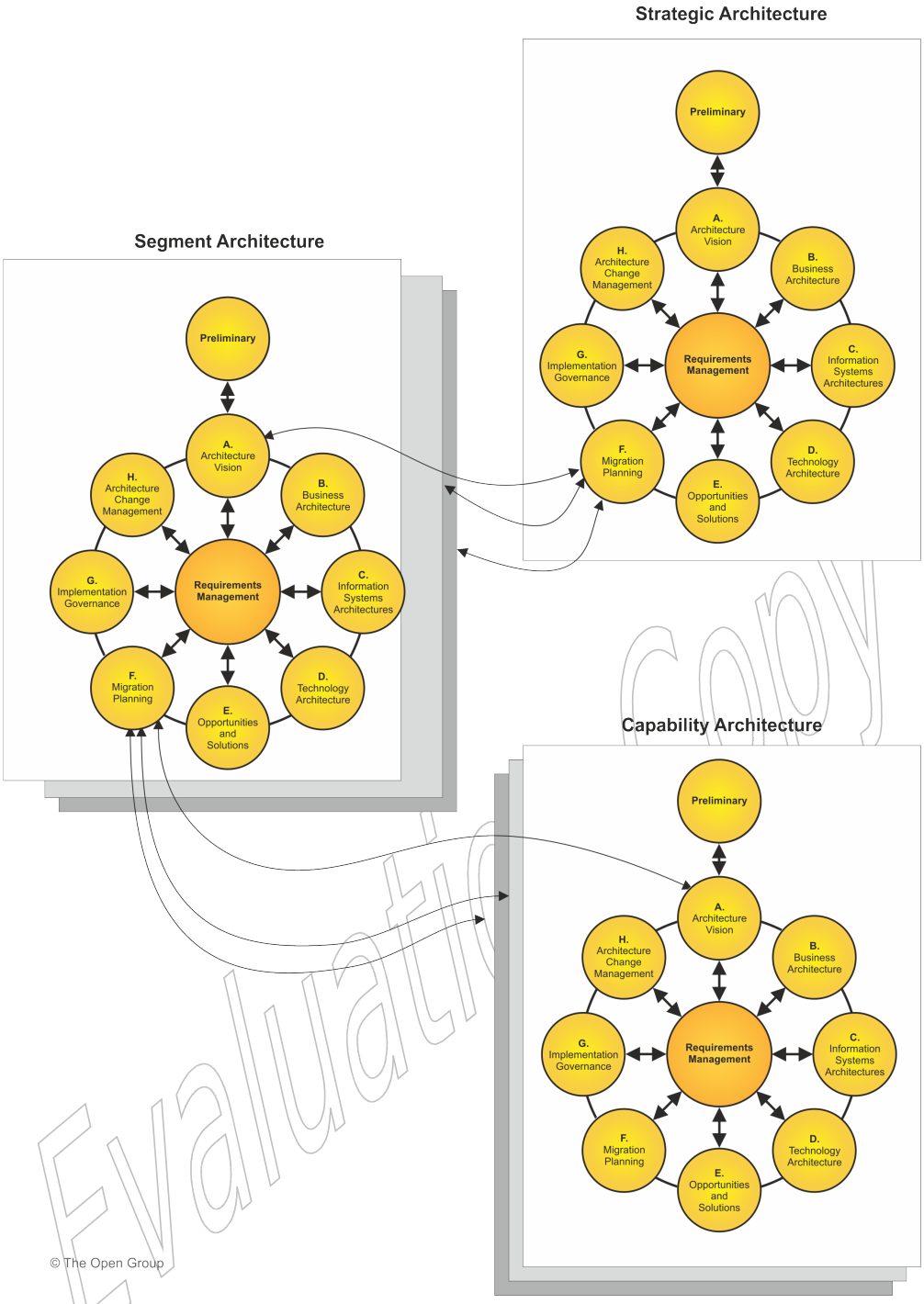

Figure 4. A Hierarchy of ADM Processes Example

As described in the previous section, TOGAF ADM phases do not have to proceed in sequence. The activities around defining Segment Architectures can start as soon as the relevant areas have been identified in the Strategic Architecture. Even if not all of these segments have been defined, architecture work can start in those that had already been identified. In a similar way, work on defining Capability Architectures need not wait until all Segment Architectures have been defined. Work on different segments and capabilities may proceed in parallel. The Strategic, Segment, and Capability Architectures are explained in more detail in Section 4.4.

Experience gained when developing a Capability Architecture should influence the higher-level

Segment Architecture. Experience gained when developing a Segment Architecture should influence the higher-level Strategic Architecture.

Agile delivery teams should collaborate closely with architects to ensure that the sprint teams understand and comply with architecture specifications (which may be expressed as guardrails or runways), and to enable rapid feedback to future architecture iterations.

It is important to keep in mind the big picture to ensure implementation teams remain aligned to the overall strategy. Therefore, a balance must be made between providing sufficient detail at each level, to give clarity and maintain alignment, *versus* exploring too much detail too soon ("Big Design Up Front" (BDUF)) without the benefit of feedback from delivery sprints. This is often referred to as the *Minimum Viable Architecture*.

4.4. Architecture Levels and Iterations

4.4.1. Strategic Architecture

In an Agile enterprise, Strategic Architecture is a high-level iteration supported by the TOGAF ADM Phase A, Architecture Vision. Strategic direction for the enterprise is defined in this iteration to support decision-making, and which may be further elaborated in Phase B to provide a high-level view of the organization landscape. It can also be used to identify required architecture and solution delivery capabilities.

Sometimes it is needed to also cover other TOGAF ADM phases, like B, C, and D, to have a clearer view on the whole landscape. In these cases, the iteration and architecture description should be high-level and not all TOGAF ADM phases will be needed.

This iteration provides identification and foundation for Segment Architectures, which can be delivered in parallel.

Key advantages for an Agile enterprise doing Strategic Architecture are:

- Provides understanding of the organization context needed to define the strategic themes, epics, and drivers; identify value streams, high-level requirements, and other broad features of the strategic direction and vision

  The Strategic Architecture provides a context for lower-level architectures.

- Confirms the basis to define guardrails for the product/service/solution delivery

- Identifies the high-level organizational capabilities necessary to deliver the entire endeavor: people skills, tooling, management tools, governance principles, etc.

- Provides an understanding of the organization landscape to shape migration planning roadmaps when loosely-coupled components are involved, thus providing the organization landscape to identify and implement the collaboration and integration needed between the relevant associated

teams

In the formation of these teams, consideration needs to be given to avoid following the organization's communication structure as the basis for the team structure. Instead the suggested approach is to consider shaping the enterprise's organization to make the Technology Architecture isomorphic with the Business Architecture.

- Input to define the backlog for the different segments (typically functional or organizational areas) that will be covered in subsequent iterations

The output of this iteration is used as input to define the backlog to be used in the different Segment Architectures.

4.4.2. Segment Architecture

A Segment Architecture is typically the specification of the product or business solution. It should be just enough architecture to identify features and functional and non-functional requirements – it may not be necessary to complete all ADM phases and steps, only the minimum needed to ensure outcomes are met. Several of these iterations can be delivered concurrently by different Agile teams. Delivery teams should be engaged during the definition of Segment Architectures to begin to build a common understanding of the solution to be delivered.

Key advantages for an Agile enterprise doing Segment Architecture are:

- Support for the definition of Capability-level backlogs
- Identify the capabilities/enablers and then the features and functionalities needed to deliver the product/service/solution
- Define the Key Performance Indicators (KPIs) needed ensure value is delivered in accordance with the vision and business objectives
- The final outcome should be the sufficient architecture to shape the products and solutions for every segment

  A segment consists of one or more products and SBBs. Just enough architecture should be delivered to enhance solution design, performance, and usability, and provide guidance for inter-team design and implementation.

- Specification at this level should be oriented to grouping things together at portfolio level to support Agile concepts that include epics, concurrent engineering, and planning for continuous delivery and integration of the target solution

Prototypes of the products or proofs of concept may be produced at this level to confirm that potential overarching solution patterns are possible.

Outputs of segment iterations are backlog items that will be the base upon which Agile teams can work

on the product and solution implementation.

This is consistent with one of the key topics mentioned in Section 1.1 about "practical experimentation": a preference for trying things out and learning from experience as opposed to extensive theoretical analysis, sometimes characterized as "fail fast".

4.4.3. Capability Architecture

Capability Architecture is the lowest level of architecture detail. It is intended to further elaborate the Segment Architecture to a level of detail sufficient to be used directly by delivery teams to implement the solution. Like Segment Architectures, many Capability Architectures may be developed in parallel.

Capability Architecture iterations support the definition of roadmaps based on architecture specifications considering the different elements and specifications for the target solution and addressing interdependencies. In a DevOps chain, Capability Architecture provides direct input to Continuous Integration (CI) and Continuous Deployment (CD). Capability Architecture outputs may be specific to one feature delivery and therefore to one delivery team (for example, a definition of some functional behavior), or may be cross-cutting across many features and teams (for example, a definition of an Application Programming Interface (API) management approach).

A Capability Architecture delivered by a sprint or even by a set of sprints depending on the scope should be incrementally and iteratively integrated into a delivery pipeline.

Key advantages for an Agile enterprise doing Capability Architecture are:

- Provides just enough detail from the higher-level architectures to define the implementation, and provides feedback to update the higher levels where necessary
- Developed Just-In-Time (JIT), to provide a forward "solution runway" for delivery sprints to consume, while avoiding unnecessary constraints
- Defines and refines the user stories that will be implemented by the different Agile teams
- Enables quality assurance and compliance activities for the solution deployment
- Enables traceability to confirm that the original objectives of the Strategic Architecture are being met

The final outcome from the Capability Architecture is the solution specification that will be constructed and deployed on demand following the architecture guidelines, metrics, and compliance considerations by Agile teams.

In this context, Agile delivery refers to delivering architecture specifications in an Agile way that will support subsequent Agile implementation (Agile solution delivery).

One important remark is the need to address the "fail fast" approach through the Strategic, Segment, and Capability Architectures; therefore, even though not explained in detail, the continuous review and retrospective of the specifications delivered should be tested so the adjustments needed can be addressed on time. For this retrospective to succeed it is key to have cross-level collaboration and

governance (this will be explained in detail in the next section). Therefore, having interdisciplinary autonomous teams is key. For more details on retrospective and sprint planning and reviews, see Section 3.2.2 in the TOGAF® Series Guide: Applying the TOGAF ADM using Agile Sprints [4].

So, at the level of Capability Architecture and based on the landscape and vision provided by the Strategic and Segment Architectures, the conception for a solution of any type can also be designed and prototyped with the objective to be tested in the implementation environment or internally very quickly. This is a good way to keep an active outside-in view and harvest input to put together a new and appealing solution. To succeed in this, it is very important to work with interdisciplinary teams to have not only a technical view or a new product but, above all, the business view.

4.4.4. Governance in Architecture Iterations

Architectural governance is necessary to ensure that solutions stay on track to achieve business targets (refer to KPIs from the Strategy level) and compliance and regulatory requirements. It also ensures the integrity of the overall endeavor through successive levels of architecture and solution implementation. Governance is also a key component of risk management. It is often through governance activities that wider impacts of local changes are recognized and addressed. Governance also has a role in providing feedback up through architecture levels where unforeseen difficulties have emerged at lower levels or during implementation.

The practical application of governance should be a collaborative and continuous effort between architecture teams at different levels, and between architects and delivery teams. In broad terms, the role of the governor is more that of an intelligent advisor embedded within a team rather than an occasionally visiting policeman. Governance should not be deferred until the final iteration before implementation, otherwise the benefits of an Agile approach are likely to be lost through rework and delays.

The value that Enterprise Architecture governance provides to Agile Architecture is to provide the set of principles and policies to guide the implementation (the guardrails mentioned in the Strategic level), the set of standards and compliance considerations. It also ensures that appropriate governance structures and approval are in place in a way that does not constrain the pace of Agile development. Again, the architect has a more consultative role in this space since they should also be checking that just the required level of architecture descriptions are in place to guide the further implementation (avoiding BDUF). Governance is also key to maintaining good communication and engagement between the different autonomous teams, providing guidance while giving them the required freedom to deliver.

4.5. ADM Levels and Phases Mapped to Agile Concepts

As previously described, the TOGAF ADM can be applied to deliver architecture iterations in parallel and partitioned across different levels of detail and change using Strategy, Segment, and Capability Architectures that can be also developed using techniques such as SAFe and Scrum.

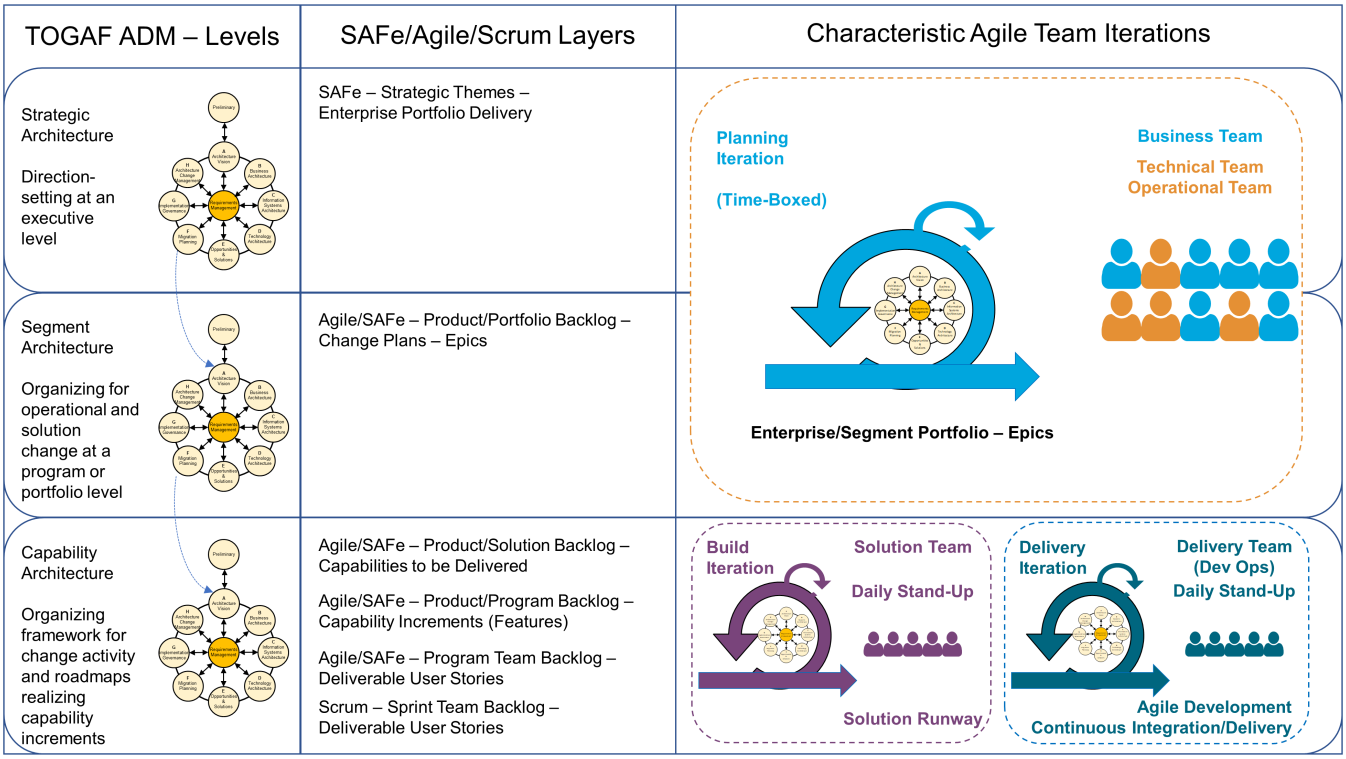

Figure 5. ADM Levels Mapped to Agile Delivery Concepts

Agile techniques consider that development and delivery work can progress based on one integrated team across all levels (business, technical, solution, build, and delivery) working in single connected sprints or across the types of boundaries shown in Figure 5[1] that may have connected but distinctly separate styles of sprints. The exact arrangement will depend on the complexity and scale of each enterprise and the implementation of the Agile approach.

In applying the TOGAF ADM, each level of planning and delivery may cycle through all of the TOGAF phases from A to G but each of the three levels will often focus on specific elements of the cycle. In the **Strategic level**, the focus is more on the Preliminary Phase (if architecture capability changes are needed) and Phases A and B to provide the basis to define the cross-enterprise and strategic change time horizon view. This generates a series of strategic high-level plans known as courses of action.

Agile techniques typically address this with concepts such as high-level strategic themes, and the highest level of an enterprise product portfolio backlog. In this level, interdisciplinary teams (business and technical teams and those that create, implement, and operate) must be involved to develop an Enterprise Architecture that meets both the business goals and objectives of the enterprise and is also potentially deliverable.

In the **Segment level**, the focus is on partitioning the courses of action across the relevant organization units (based on an understanding of the desired business capabilities and value streams and following each enterprise's chosen approach to partitioning – capability/product, service/process, or function etc.) such that the work of delivering the change can be effectively and efficiently organized. Where the information acquired by performing Phases A and B is insufficient for this activity there may be more emphasis on further exploring Phases B, C, and D in greater detail.

Work can be approached by factoring work to self-organizing teams at various levels (in line with the chosen organization unit structure) along with a high-level iteration through Phases C and D, that provides more detailed information for the product or solution delivery, going deeper into smaller organization areas (segments). Outputs of this iteration are the Epics that reflect large or long-running user stories, and the segment-based initial portfolio and/or backlog. The output from this level can be used to test and experiment with new products (if necessary), delivering descriptions for prototypes to test ideas into the relevant segment market.

In the **Capability level**, a more specific solution-oriented architecture specification, including the ABBs, is identified through Phases B, C, and D, covering both the functional and non-functional aspects of the solution to be implemented. These architecture specifications are then further developed in Phases E and F as the basis for the SBBs, and their integration into the desired solutions/services/products. These are finalized and their associated contracts are then developed to direct their reuse/acquisition/build and deployment. The implementation units are aligned with increments of capability that will deliver specific outcomes such that each "chunk" of work produces an implementation of agreed value with the relevant stakeholders and sponsors. This approach to fast continuous implementation at the smallest level of capability creates a Transition Architecture of deliverable units similar to the sprints in the lowest level of backlog in the Agile style approaches. This smallest level of capability is often referred to as *Minimum Viable Product* in Agile style approaches.

The backlogs are (if needed) usually refined down to an equivalent of the Scrum concept of a sprint backlog of deliverable sprints that will be ready for implementation in weeks, or at most one month. These sprints will take the specification from the equivalent of the TOGAF Phases A and/or B, C, D, and E through to implementation. The focus is on creating integrated teams and environments such that the further design, build, implementation, and operation processes interact seamlessly enabling continuous integration and implementation as delivery of each unit of the Minimum Viable Product is completed.

The Capability level is operationally completed in Phase G, Implementation Governance. This ensures that the agreed contracts have delivered the expected capability, in line with the contractual agreements, and that all of the required information for operating and changing the product/solution/service is properly created, stored, and made accessible to support faster and higher quality change in the future. The Capability level confirms benefits realization in Phase H, Architecture Change Management. This ensures that operational and business performance is evaluated to confirm that the value has in fact been delivered and that the users of the product/solution/services are satisfied with the business outcomes of that capability increment. It further ensures that the evolving or completed wider Segment and/or Strategic-level change projects are operating within the appropriate boundaries (or guardrails) set up when planning the change.

There may be pressure to move forward beyond the end of the runway defined by the Capability Architecture at a given point in time. This can store up problems in the management of poorly documented or architecturally dislocated products/solutions/services in the future. This is a type of technical debt, and like any debt, needs careful management to ensure that the debt does not get out of control. This should be addressed in Phase H, Architecture Change Management.

4.6. Set-Based Concurrent Engineering

Set-Based Concurrent Engineering (SBCE) from Lean Product and Process Development [5] can be accommodated by adapting the TOGAF framework. Both the Open Agile Architecture™ Standard [6] and SAFe® promote the practice of SBCE. In SBCE, the overall solution is divided into a number of sub-systems, broad targets are established for each sub-system, and multiple alternatives for reaching those targets are identified and developed concurrently. The set of alternative designs is evaluated iteratively over time, eliminating weaker options in each iteration, until the final, strongest design is selected. For each alternative, enough architecture is done to properly assess the competing alternatives, ensuring that good decisions are made. Architectural alternatives and elaborations are done using ADM iteration cycles and sprints. An ADM iteration is done for each alternative, with only enough architecture done within the iteration to evaluate the trade-offs for that alternative. When an alternative is selected, ADM iterations and sprints are done to further elaborate the architecture for the selected approach. Further explanation on how to handle alternatives using the TOGAF Standard can be found in The Open Group TOGAF Standard [1] – ADM Techniques, Chapter 10, “Architecture Alternatives”.

These iteration cycles with accommodation for SBCE support an Agile delivery style. For instance, a Segment Architecture could be defined in Agile sprints and each one of those sprints would be an ADM iteration cycle.

4.7. Selecting Delivery Styles

This document describes how to apply the TOGAF methodology and framework within an Agile delivery environment. However, different delivery styles exist, and it may be that some projects or programs are not appropriate for Agile delivery. The TOGAF Standard is intentionally flexible to accommodate these different styles.

The TOGAF ADM is a framework with a set of possible elements that can be applied to different scales of change, from meta to micro.

- The meta change is the evolving enterprise state space; large-scale change is the evolving backlog, project, or program in scope
- The micro change is the delivered transition; small-scale change is the delivery unit of each transition within the large-scale change

The meta/micro cycle distinction is usually of characteristic delivery style, dependent upon the needs of each change activity; such as:

- Rapid style: solution – iteration
- Agile style: backlog – sprint
- Robust style smaller-scale: PRINCE®: project – stage
- Robust style larger-scale: Managed Service Provider (MSP): program – project

4.7.1. The Three Delivery Styles

Agile is one of the three broad lifecycle delivery approaches which can work in concert with the TOGAF Standard. These provide for a proportionate and controlled approach for delivering change at an effective pace, and in response to the appetite for business transitions. However, agility is not just about using sprints; it is about the right balance of approaches based on the needs of the enterprise.

There is both a faster type of approach (*Rapid*), and a slower type of approach (*Robust*). It is normally the case that any meta change of scale and consequence (a backlog of sprints in Agile, a series of transitions in the TOGAF Standard) will have a series of iterations of packages of work that will often involve all three types of change lifecycle. Also note that the Robust approach incorporates elements of Agile where appropriate and the Agile approach incorporates elements of Rapid where appropriate. They are essentially used in a nested manner depending on the complexity and scale of a specific change.

Table 1 shows the characteristic profiles of projects related to each approach. The table is a general guide not a rigid statement, and acts as a guide for considering an effective approach for each meta/micro delivery cycle and each enterprise/change activity will decide how best to approach that change. Agile product management is discussed in Chapter 5.

The three main lifecycle delivery approaches that can be broadly identified are:

- Rapid – near immediate implementations of simple components (e.g., extended prototyping such as Rapid Application Development (RAD))
- Agile – fast cycles of component delivery for specific bounded functionality (e.g., Java™ Application Description (JAD), Dynamic Systems Development Method (DSDM®), Agile)
- Robust (risk and architecture-driven) – longer-term delivery of complex, large-scale components, interoperable across the breadth of a segment or an enterprise (e.g., managed projects or programs such as PRINCE2 and MSP)

Table 1. Characteristic Delivery Lifecycle Change Approaches and their Profiles

	Rapid	Agile	Robust
Small scale	Y	Y	N
Large scale	N	N	Y
Simple	Y	Y	N
Complex	N	N	Y
Tried technology	Y	Y	Y
New technology	N	Y	Y
Stable business process	Y	Y	Y
Changing business process	N	Y	Y
Skills already in place	Y	Y	Y

	Rapid	Agile	Robust
Skills needed to be developed	N	Y	Y
New solution	N	Y	Y
Extension of existing solution	Y	Y	Y
Integration of existing building blocks	Y	Y	Y
Integration of existing running services	Y	Y	Y
Short-term change horizon	Y	Y	Y
Long-term change horizon	N	N	Y
Time-driven project (shorter time frame)	Y	Y	N
Time-driven project (longer time frame)	N	N	Y
Cost-driven project	Y	Y	Y
Quality-driven project	N	N	Y
Function-driven project	N	Y	Y
Structure/architecture-driven project	N	N	Y
Low Risk	Y	Y	N
High Risk	N	N	Y

Note that for any project of some scale or complexity, each chunk (sprint, iteration, stage, transition, etc.) may require a different approach such that the project may well incorporate all of these approaches based on the specific circumstances.

In most changes there will be a significant, if not greater, number of transitions implemented using Agile style approaches. Occasionally pure speed is needed (Rapid). In some changes, really careful (Robust) approaches will be chosen when there is high risk or significant consequences associated with the change. When the Agile delivery lifecycle is the most appropriate approach it integrates well with the TOGAF Standard and the delivered sprints represent the transitions defined in the standard that move towards the evolving and changing Target Architecture. Again, the TOGAF Standard does not take an abstract position on which of these delivery lifecycle approaches is best for a meta or micro iteration. Rather, in its Architecture Change Management phase, it provides the structures and controls within which an enterprise can have true agility and choose the best approach for each specific change situation.

The concept of fast-focused change underlying Agile is powerful but not singular. There are many paces not just one pace. Often many dependent elements need to be put into place to enable even the simplest of functions in an effective manner. For example, in the world of athletics there are four main paces:

- Sprint
- Middle distance

- Long distance
- Marathon

Each of these paces has their place depending on the situation encountered. Using the wrong pace is not just less than good, it often means total failure. Within any one project, high levels of risk may require slower paces of change to ensure that each element has properly managed that risk; while lower levels of risk may enable a solution to be implemented more quickly. Within any change program, different iterations will often need to be addressed at different paces. It is expected that each iteration will be identified as having an appropriate pace and then be organized to deliver at that pace.

Early Agile approaches, such as Scrum, emphasized the delivery of small units of delivery in short sprints at a fast pace. (Note that later Agile approaches, such as SAFe, have broadened this to include larger units of delivery with more content working at different paces.) However, this does not apply in many situations. In large complex situations that are strongly-coupled, stateful, pessimistic, and controlled in nature (such as Nuclear Power Stations, Aircraft Control Systems, or non-repudiation situations) many teams may have to be involved and require significant levels of control and direction. There is a reason why the most important element of a successful large-scale military organization is logistics (organization, consistency, and delivery at scale – potentially in a hub-and-spoke architecture, as in efficient goods delivery) and not commando units (which do have their place and are very effective when used in the right situations, but not at scale).

4.8. Agility at the Highest Levels of Architecture

As Figure 6 shows, the highest level of architecture begins with an architecture vision and strategy at the enterprise level, which provides a long-term view of the target state and surrounding business landscape. The development of the vision and strategy can be done in an Agile style, through multiple iterations as architects collaborate with their business partners. Typically, architecture delivered at this level is visionary and conceptual, and often provides a lot of flexibility to architects who use this vision to prepare more detailed architectures. The release of such high-level deliverables can span a long duration, and an indicator of organizational agility is the speed with which vision and strategy can change.

The speed of change also depends on the impact of the change. Changes that are non-disruptive to the current vision and strategy are relatively quick and easy to complete. Changes that create a huge pivot in the current strategy and vision should be assessed to see how much disruption they may cause to the currently derived or in-flight architectures. Such changes could have a trickle-down effect as many derived architectures at lower levels are impacted. The frequency and the amount of changes to the architectures at the highest levels should be less than those at the lower levels. How much change is feasible depends on the organization's tolerance to the amount of agility.

Business partners and architects collaborate constantly to produce transition state architectures and align on the aspirational target states. Architects should also align to business goals and evolve the architectures such that every transition state provides an incremental business value.

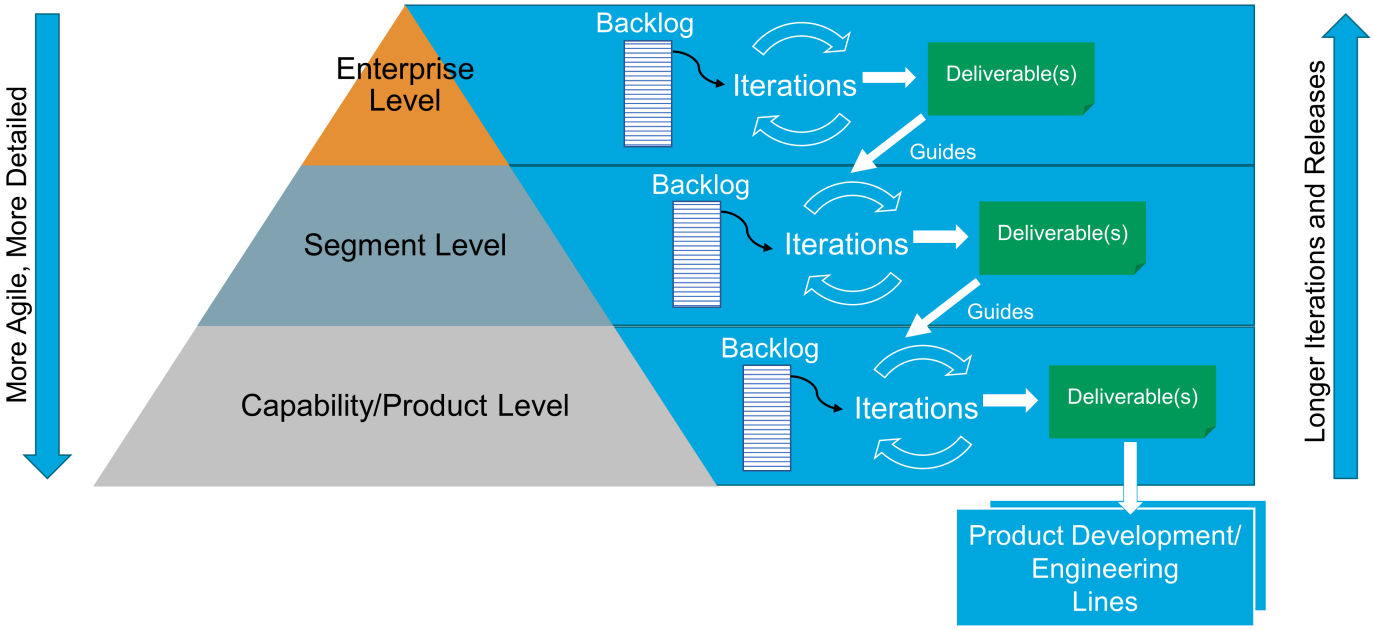

Figure 6. Agility at Different Levels of Architecture

[1] SAFe/Agile definitions taken from the Agile Alliance Glossary; refer to: https://www.agilealliance.org/agile101/agile-glossary

Chapter 5. Using Agile Product Management Techniques

Many techniques have become popular within Agile product management that can be beneficial when applied to the TOGAF ADM. This chapter describes some of those techniques and how they can be applied.

5.1. Establishing the Enterprise Architecture Capability

As discussed in the previous chapter, Agile Enterprise Architecture uses an iterative, incremental approach to create architecture artifacts. Specific skills and management approaches must be put in place for this to succeed. The primary objective of the Preliminary Phase of the ADM is to establish or adapt the Enterprise Architecture capability to the specific architecture work to be carried out. Some considerations for the Preliminary Phase in Agile delivery are:

- Agile Architecture requires a much closer focus on the outcomes, involving a shift from a project to a **product-centric** approach
- While most Agile efforts take place in the solution space, techniques that address the problem space, such as design thinking and business model canvassing, help to shape the architecture direction to achieve the solution

  Requirements often come on a JIT basis, requiring iterative evaluation to reaffirm the problem definition (TOGAF ADM Phase A) and the architecture approach to arrive at the desired solution (TOGAF ADM Phases B, C, and D).
- This produces an holistic architecture approach with less individual focus on the “domains” of architecture (Business, Data, Application, and Technology) and more emphasis on collaborative views across all domains focused around the required product
- This will require changes to the way that Enterprise Architecture and product development is managed and governed

The Preliminary Phase includes the establishment of a set of Architecture Principles (general rules and guidelines intended to be enduring and seldom amended) that inform and support the ways in which an organization sets about fulfilling its mission. The Architecture Principles for any style of architecture reflect the specific characteristics of that style. The principle of a single product owner should be established in this phase to ensure that requirements are clearly articulated and understood, and to prioritize the Agile development of the system components in later phases.

Chapter 5 of the Open Agile Architecture Standard [6], “Intentional Architecture”, sets out principles that intentional architecture within Agile delivery should follow.

5.2. Product Development and Architecture

As shown in Figure 7, a traditional, generalized view of Enterprise Architecture contains a number of stages.

Figure 7. Traditional Stages of Enterprise Architecture

In terms of the TOGAF ADM:

- Phase A defines the problem
- Phases B, C, and D define the baseline and target
- Phases E, F, and G deploy the target
- Phase H manages change

In an Agile environment it is likely that these activities will be a continuously managed process with activities proceeding in parallel, as shown in Figure 8.

It is not necessary to complete the problem definition before starting other activities. Enough of the problem needs to be defined to provide context for other activities. As other activities are initiated, work can proceed to further elaborate and extend the problem definition.

Clearly, there are dependencies. It is not possible to deploy any part of the target until the parts of the baseline and target describing that part of the target have been sufficiently defined.

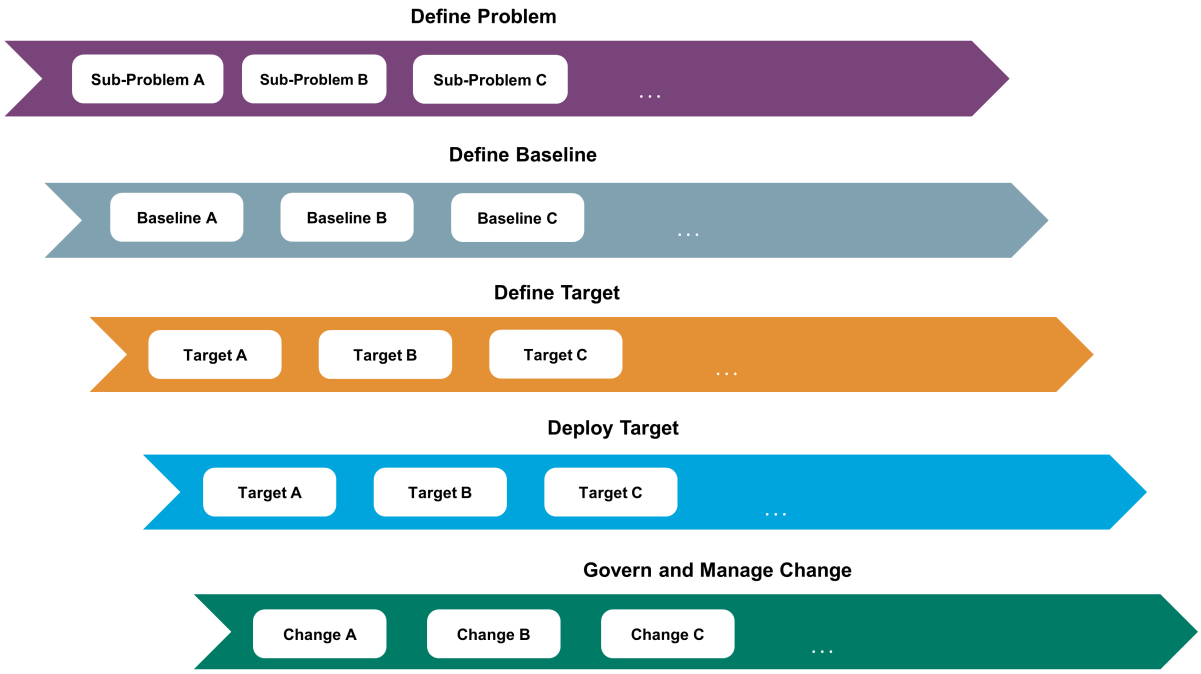

Figure 8. Stages of Enterprise Architecture in an Agile Environment

More practical guidance about different architecture styles of support can be found in the TOGAF Standard [1] – Applying the ADM.

5.2.1. Define Problem

> "Identify the key stakeholders and their concerns/objectives, and define the key business requirements to be addressed."
>
> — The TOGAF Standard

> "The level of detail addressed in Phase A will depend on the subset of scope and goals associated with this iteration of architecture development."
>
> — The TOGAF Standard

A sufficient understanding of the strategic goals of the enterprise is a prerequisite for any activity, to ensure that individual activities do not undermine those goals.

How to improve agility at this stage:

- Identify the product owner
- Seek to develop a high-level Strategic Architecture quickly, with only the detail required to plan more constrained activities
- Use one of the many available Agile tools and techniques to characterize the problem and define

the specific set of requirements to be addressed; such as:

- Business requirements
- Customer journeys
- Design thinking: an approach for defining a problem or finding novel and innovative solutions

  Table 2 describes the key characteristics of design thinking.

  Table 2. Design Thinking

Practices	Thinking Styles	Mentality
Human-centered approach; e.g., people-based, user-centered, empathizing, ethnography, observation	**Thinking by doing**; e.g., early and fast prototyping, fast learning, rapid iterative development cycles	**Combination of divergent and convergent approaches**; e.g., ideation, pattern finding, creating multiple alternatives
Collaborative work style; e.g., multi-disciplinary collaboration, involving many stakeholders, interdisciplinary teams	**Abductive reasoning**; e.g., the logic of "what could be", finding new opportunities, the urge to create something new, challenging the norm	**Reflective reframing**; e.g., rephrasing the problem, going beyond what is obvious to see what lies behind the problem, challenging the given problem
Holistic view; e.g., systems thinking, 360 degree view on the issue	**Integrative thinking**; e.g., harmonious balance, creative resolution of tension, finding the balance between validity and reliability	**Experimental & explorative**; e.g., the license to explore possibilities, risking failure, failing fast
Ambiguity-tolerant; e.g., allowing for ambiguity, tolerance for ambiguity, comfortable with ambiguity, liquid and open process	**Optimistic**; e.g., viewing constraints as positive, optimism attitude, enjoying problem solving	**Future-oriented**; e.g., orientation towards the future, vision *versus* status quo, intuition as a driving force

- Epics
- Business Model Canvas: a management tool for understanding the fundamental characteristics of the business; see the template in Figure 9. For more details, refer to the TOGAF® Series Guide: Business Models [8].

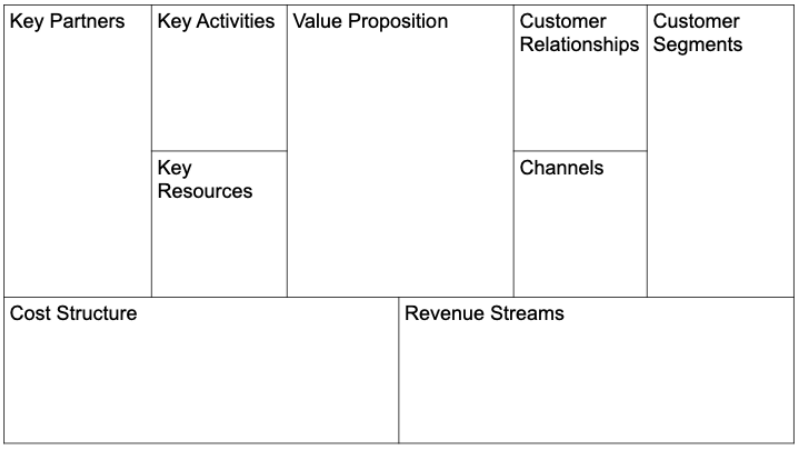

Figure 9. Business Model Canvas

To ensure that "just enough" architecture is done:

- Focus on the required outcomes
- Define the Minimum Viable Product that the enterprise can offer to its customers

Identify new business opportunities considering new trends in business and technology, taking an outside-in approach considering customer demands and competitor activities.

> "Agile Architecture shall use marketing and design methods to discover how customers are likely to use products and services. Jobs-to-be-done analysis, customer journey mapping, and design thinking are examples of methods used by Agile enterprises. Design thinking, which is a human-centered approach, incorporates human cognition and emotion as key aspects of the value definition."
>
> — The Open Group Open Agile Architecture Standard, Axiom 2: Outside-In Thinking

Value shall be specified from the standpoint of the customer. A value stream shall be identified for each product or service family from concept to launch and from order to delivery. Enterprise Architecture should support value stream definition and also the business capabilities needed to fulfil the need.

5.2.2. Define Baseline

The scope and level of detail to be defined will depend on the extent to which existing elements are likely to be carried over into the Target Architecture.

Agile development is high risk in the absence of a sufficient understanding of the baseline.

How to improve agility at this stage:

- Progressive development of the Baseline Architecture

- Understand the big picture
- Segment the problem
- Develop detail as required to support subsequent work
- Consider defining the target first to scope work on the baseline

The key measure is understanding the Minimum Viable Architecture necessary to manage trade-offs and risk.

5.2.3. Define Target

The scope and level of detail to be defined will depend on the relevance of the business elements to attaining the Target Architecture Vision, and on whether architecture descriptions exist.

How to improve agility at this stage:

- Progressive development of the Target Architecture within the context of the big picture
- Focus on the Minimum Viable Product
- Develop detail as required to support subsequent work – only the needed detail ("just enough architecture")
- Be guided by the enterprise strategy and prioritized backlog

Again, the key measure is understanding the Minimum Viable Architecture.

Once the problem space has been described then a more detailed architecture specification has to be delivered to describe the Business, Data, Application, and Technology building blocks needed. These will become the input items for a program backlog. Product managers and product owners are key stakeholders in this stage.

As described in Chapter 4, Segment and Capability Architecture concepts can be applied where necessary to decompose the problem into smaller pieces to provide a more progressive and iterative delivery. The functional and non-functional components of the architecture are decomposed into user stories which make up the architecture backlog. User stories are prioritized by the product owner in consultation with the Agile team. Those stories prioritized as "must have" collectively become the Minimum Viable Architecture.

Once some initial architecture specifications are available, these will define the SBBs and begin to build the sprint backlog; i.e., they become the intentional architecture. Intentional architecture is a set of purposeful, planned architectural strategies and initiatives, which enhance solution design, performance, and usability and provide guidance for inter-team design and implementation synchronization.

5.2.4. Develop Target

Ensure that the systems development method enables feedback to the architecture design.

Adopt one of the many successful Agile development methods:

- Make sure that people have the necessary skills
- Ensure that there is appropriate collaboration and governance to deliver the desired business value

How to manage risk:

- Maintain linkage to architecture descriptions
- Ensure sufficient feedback to assess conformance to architecture
- Maintain Transition Architectures through an efficient and Agile change management process

5.2.5. Govern and Manage Change

(Drivers for change) Experience with previously delivered project increments.

Governance should be embedded within sprints, and as part of sprint retrospectives and sprint planning:

- Provide guidance to solution construction to keep within guardrails and follow the solution runway
  - Guardrails may be technical boundaries or constraints such as specifying standard components to be used, performance and capacity limits, or technical policies
  - The runway is the architectural definitions or technical superstructure that will be consumed in future system construction sprints
- Assess the impact of change on higher-level architectures
- Gain feedback to evolve higher-level architectures based on experience of each sprint (capability increment)

Once solution implementation has started, the architecture team should refine the governance metrics that will be applied to test compliance against the Target Architecture. This compliance could be composed by Objectives and Key Results (OKRs) to ensure value is being delivered and also to ensure alignment with standards and regulations.

As defined in the O-AA™ Standard [6], OKRs spell out the company's priorities in terms of specific accomplishments and performance improvements. The objectives consist of a clearly defined qualitative change, while the key result is a specific, often quantitative performance target that must be met. Agile organizations should set OKRs annually and assess progress against them on a quarterly basis.

Change is an inherent part of Agile delivery, and architecture governance can help to maintain alignment between delivery teams and alignment with higher-level architectures. If a change is large, then a pivot may be necessary, but an assessment should be made at the first high-level iteration to be sure the final objective and value delivery is not lost, and if necessary to adjust business priorities and interdependencies.

5.3. Architecture, Product Development, and Delivery

In an Agile environment and within supporting organizations that have adopted an Agile delivery approach, it is likely that the activities presented in Section 5.2 will be a continuously managed process handled in parallel and focusing more on the outcome with architecture as the vehicle to support product delivery.

For an Agile approach the stages to follow are similar but performed in a different way. Enterprise Architecture iterations support this product definition and product architecture.

The problem is defined at the Strategic and Segment level, then at the Segment and Capability level a target is defined, and at the Capability level a target is deployed by Agile teams supported by the architect and the Capability Architecture definition.

As explained in Section 4.4, at the Strategic level a high-level architecture specification is defined – strategic themes, epics, that will guide the subsequent problem definition. At the Segment level, the backlog is defined in alignment with the strategic definition (problem definition). This backlog should be refined further to produce the sprint backlog that will be delivered using any Agile technique. This Agile delivery is supported by a Capability Architecture supporting Agile solution delivery.

5.3.1. Identification of a New Need – Define and Identify the Problem

Apply Enterprise Architecture principles and stakeholder assessment, supported by other techniques like design thinking, to identify new business opportunities considering the new trends in business and technology, taking an outside-in approach to consider customer demands, and assessing what competitors are doing.

As stated in Axiom 2, Outside-In Thinking, in the O-AA Standard [6], Agile Architecture shall use marketing and design methods to discover how customers are likely to use products and services. Jobs-to-be-done analysis, customer journey mapping, and design thinking are examples of methods used by Agile enterprises. Design thinking, which is a human-centered approach, incorporates human cognition and emotion as key aspects of the value definition.

Another important consideration is to perform a business readiness assessment, to adopt a new emerging technology that might be related with the new business need identified, and also to consider regulatory and compliance – adherence to standards and corporative principles is also a key consideration while defining the problem.

Agile Architecture should identify the enterprise value streams. Value shall be specified from the standpoint of the customer. A value stream has to be identified for each product or service family from concept to launch and from order to delivery. Enterprise Architecture should support the value stream definition and also the business capabilities needed to fulfill the need.

5.3.2. Define Target

Once the problem space has been described then a more detailed architecture specification has to be

delivered to describe the Business, Data, Application, and Technology building blocks needed – this specification will be the input for a product backlog definition. Product managers and product owners are key stakeholders in this process.

In order to provide an Agile approach, it is also necessary to segment the problem.

The Segment and Capability Architecture concepts should be applied to decouple the problem into smaller pieces so that the specifications will be delivered in an Agile way.

The features depicted in the architecture will be useful to define priorities for the product backlog and to identify any interdependencies that will facilitate the definition of the sprint backlog.

Once the architecture specifications have been finished, they will provide the input to define the SBBs and to shape the final solution following the partitions concept – this solution definition is the input for the sprint backlog definition and conforms the basis for the intentional architecture that will support Agile implementation.

Once the Agile teams start the delivery process based on the architecture descriptions, Enterprise Architecture teams should refine the governance metrics that will be applied to test compliance. This compliance could be composed through OKRs to ensure value is being delivered and also to ensure alignment with standards and regulatory compliance.

Governance activity should not be made in isolation, so every time an Agile team delivers an output a retrospective should be made, and the governance process should be part of that sprint iteration retrospective and planning.

On the other hand, change management should also continue during the whole process since changes could have an impact in any of the iterations described above. Therefore, these changes should be identified in a dynamic way so changes can be applied quickly to avoid further delays in the process.

If a change is too big or disruptive, then analysis at the first high-level iteration should be carried out to be sure the final objective and value delivery is not lost and the proper adjustments in priorities and interdependencies can be made.

An important concept to follow for Agile Architecture is intentional architecture, which specifies a set of purposeful, planned architectural strategies and initiatives to enhance solution design, performance, and usability and provide guidance for inter-team design and implementation synchronization.

In Figure 10, intentional architecture is delivered as part of the Capability Architecture specification and it is aimed to provide guardrails to support implementation.

The output of the Strategic Architecture gives input to define the product backlog that is defined in further detail in the Segment Architecture. At the Capability Architecture level, the spring backlog is defined and also the intentional architecture that will guide implementation (deploy target).

Change management will address any new requirements and features and will support the refinement

and prioritization for the product backlog, for them to be distributed in different releases.

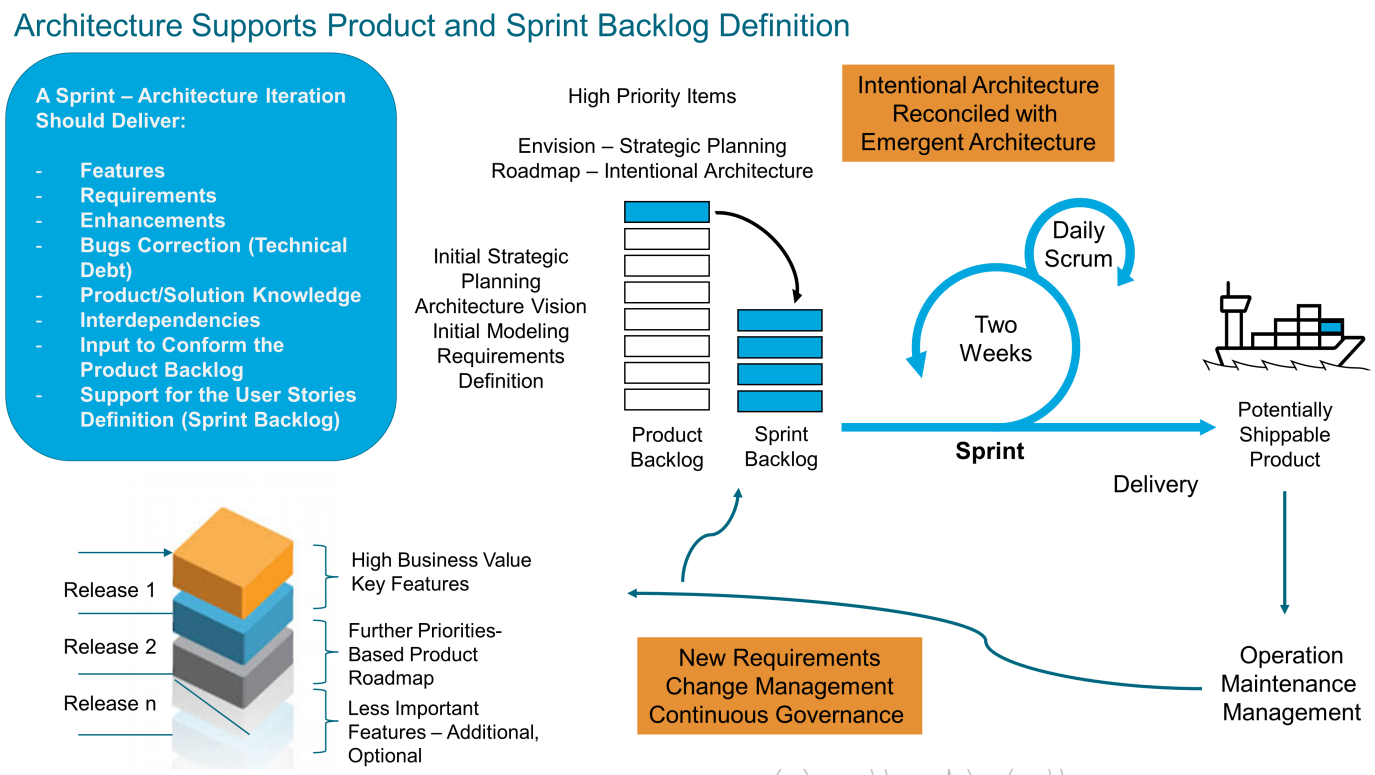

Figure 10. Agile Architecture and Product Development

5.3.3. Projects *versus* Products

Project and product management are closely related, but in general *product* management is a broader discipline than *project* management. Product management covers the full lifecycle of discovery, development, delivery, long-term support and maintenance, and disposal, although disposal is often a relatively trivial step with digital products. Within this product lifecycle, there are likely to be a number of projects and processes invoked; e.g., a project for each major product deliverable, and a process to handle support requests from product users. Figure 11 shows the relationship between products and projects.

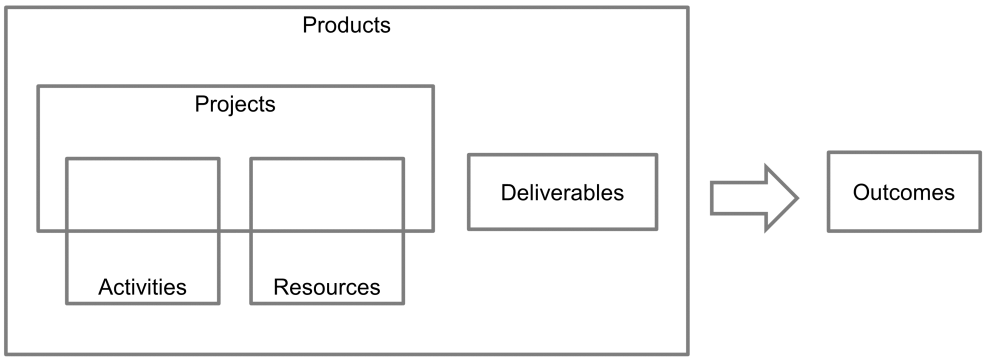

Figure 11. Products and Projects, based on Figure 43 from the Digital Practitioner Body of Knowledge™ Standard

For further information on product management techniques, refer to the DPBoK™ Standard [9].

5.4. Architecture Artifacts

The TOGAF Standard – Architecture Content describes the TOGAF Content Framework. In general, the artifacts and their relationships apply equally to Agile delivery as to other delivery lifecycle approaches. However, the tooling and format may need to be adapted for Agile delivery.

5.4.1. Automation of Enterprise Architecture

The implementation of TOGAF architecture artifacts can be accelerated with the use of Enterprise Architecture tools. These tools can collect business operational data in real-time and build/enhance architecture models. Using Artificial Intelligence (AI) algorithms, they can optimize the models. The tools also enable ideation and collaboration between the users by showing the impact of adopting various digital technologies. Most importantly, these tools enable the Enterprise Architecture to quantify and measure transformation changes to the architecture.

5.4.2. Minimalistic Artifacts

Artifacts and their templates should contain the minimum content that is consistent with their purpose, and no more. The progressive, iterative development of architecture described in previous sections of this document can be realized by producing the artifacts using a series of templates of increasing detail and/or scope. For example, a first artifact from Phase B, Business Architecture, might contain just the Baseline Architecture, or for large projects, this could be further divided into an overview artifact followed by a series of artifacts with more detailed descriptions.

5.5. TOGAF ADM Phases and Artifacts Supporting Product Architecture

- **Preliminary Phase**
  - Define the Enterprise Architecture organization capabilities and maturity model to support an optimized Business, Data, Application, and Technology Architecture, especially if they respond to disruptive trends from the market; for example, new value propositions to evolve the organization into digital product offerings and the customer journey mapping for their delivery
  - Define the Enterprise Architecture principles and governance framework to support the Architecture Vision
  - Define the changes needed at the organizational level to adapt the organization capabilities and structure to implement the business value and to support development and deployment in an efficient and effective way; for example, adapting to fulfill Agile methodologies for application/system/portfolio delivery when and if necessary

    A new model for organizational structure might be needed which implies new roles, skills, and capabilities and a new organizational model cross-cutting organizational units and aimed at having autonomous self-empowered Agile teams. This will allow organizations to scale their Agile development.

- Artifacts needed:
  - Organization map; refer to the TOGAF® Series Guide: Organization Mapping [10]
  - How to set an Enterprise Architecture capability; refer to the TOGAF® Series Guide: The TOGAF® Leader's Guide to Establishing and Evolving an EA Capability [11]
- Organizational principles – review and adapt if needed

- **Phase A: Architecture Vision**
  - Define the business value proposition and strategy in alignment with the organization strategy and mission
  - If the whole company strategy needs to be redefined to fulfill the new trends this needs to be addressed in the Architecture Vision phase
  - Artifacts needed:
    - Strategy and motivation models, depicting product strategy to support Capability-level decision-making
  - Additional support:
    - How to adapt the ADM to support strategy architecture and decision taking; refer to the TOGAF® Series Guide: A Practitioners' Approach to Developing Enterprise Architecture Following the TOGAF® ADM [12]
    - Value stream and business capability modeling, business model guides for high-level model of the product; refer to the TOGAF® Series Guide Set: Business Architecture [7]
  - At this stage it is very important to vision the Enterprise Architecture as modular so its delivery and support will be more efficient and easier to change
- **Phase B: Business Architecture**
  - Applied to define the key features and benefits aimed to solve the problem space and build the solution space
  - Requirements definition and translation into functional and non-functional requirements – and define the product target market, features, and value proposition for each
  - Design the value streams and business capabilities needed to deliver the product
  - Artifacts needed:
    - Value stream, business capabilities, connected with the organizational map, information map to identify key information needed – business models like BMC and also the business operational model
  - Model the customer journey for the different audiences; refer to the TOGAF® Series Guide Set: Business Architecture [7]
- **Phase C: Information Systems Architectures**
  - Information needed to deliver the product features and to handle relevant customer segments – information that will be needed to build and support the product in its different delivery stages

and also to market and sell the product to the target audiences and customers: customer profiling

- Refine the customer journey map with the information needed and the data analytics needed

- **Phase C: Application Architecture**
  - Map the product functional features with the capabilities needed for the product to be delivered into the market
  - Artifacts needed:
    - Mapping the functions with application components and services
  - The key to applying architectural patterns based in micro-services and loosely-coupled components, making them easier to develop, deliver, and maintain and with clear integration points, is to pursue effective integration and delivery
- **Phase D: Technology Architecture**
  - Technology or technical platforms to support product development, delivery, and maintenance – the approach depends on the kind of product: if it is an application or software system then it will be the technological platforms supporting the application components and services; if the approach is a cloud-based service then the approach should be adapted to fulfill that

Index