THE Open GROUP

Using Agile Practices in Enterprise Architecture

A White Paper by:

Serge Bouwens, ArchiXL & ANWB
Mats Gejnevall, minnovate & Biner Consulting
Piotr Papros, 4IT Group
Jacek Presz, Bank Pocztowy
Mirosław Prywata, Asseco Data Systems SA
Marcin Wizgrid, Accenture
Łukasz Wrześniewski, 4IT Group
Aleksander Wyka, ATE Enterprises

May 2019

Using Agile Practices in Enterprise Architecture

Document No.: W194

Published by The Open Group, May 2019.

Any 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 email to:

ogpubs@opengroup.org

Table of Contents

Boundaryless Information Flow ™
achieved through global interoperability
in a secure, reliable, and timely manner

Executive Summary

Enterprise Architecture and Agility are becoming more important than ever. We need to combine forces.

Agility is important because there is an increasing demand for enterprises to learn continuously and deliver benefits more quickly. There is an imperative to respond quickly to drivers for change, and to be able to change direction quickly and safely. Enterprise Architecture, on the other hand, focuses more on strategic direction and business value. Enterprise Architecture provides a framework for change of the organization, and a way to manage complexity, support continuous change, and manage the risk of unanticipated consequences.

This White Paper expresses the view that an enterprise needs both. It offers a pragmatic approach to help practitioners of both Enterprise Architecture and Agile to understand how and when these practices complement each other. It provides agile practices to the Enterprise Architect and Enterprise Architecture focus to the agile practitioner based on the experience of the authors.

The focus is on agile architecture practices and processes. To have a better understanding of the reasons why a change from a traditional style of practicing architecture to a more agile style of architecting is necessary, this paper explains the paradigm shift that is occurring in general, and translates that to the architecture context. From that, practical advice is offered to three Enterprise Architecture practice areas:

- Creating Enterprise Architecture artifacts in an agile way
- Engaging the strategic level in an agile way
- Engaging with agile development teams

Introduction

Time-to-market concerns of an enterprise have created a demand to deliver value in short intervals. Following the examples of new business start-ups and leaders in innovation, many enterprises have adopted new approaches to product design and system development to meet this demand.

Enterprise Architecture as a discipline is aimed at enabling businesses to undergo transformations to deliver value to investors and customers. The idea of delivering incremental value across the enterprise has long existed in the TOGAF® standard, and in other Enterprise Architecture standards and various bodies of knowledge.

However, by separating the efforts to understand the needs of the enterprise and guiding a transformation from managing cost, time, and resource deployment, many individuals and enterprises have taken the concept of "time-to-market" for granted in their Enterprise Architecture practices. This attitude has somewhat tarnished the image of the practice.

It is generally understood that Enterprise Architecture is expected to be about meeting the needs of the enterprise, whatever it may be at any point in time. In this light, if an enterprise needs agility and agile delivery at any stage, then good Enterprise Architecture will adapt itself to enable and support it as well. This could be happening informally in some organizations.

Guidance is required for Enterprise Architecture practitioners to not lose sight of the time-to-market concerns of the enterprise and for agile practitioners to leverage the acceleration, guidance, and an approach for the practice to be used along with the Enterprise Architecture practice. In the context of each enterprise, these two practices address overlapping concerns and have an opportunity to leverage evolving ideas from each other.

Purpose of this White Paper

This White Paper is written to deliver one simple message: agile practice should be incorporated into Enterprise Architecture. Agile and Enterprise Architecture share some values, but not all. And these values are appreciated differently, as the Agile Manifesto has expressed so eloquently.[1] From these different sets of values, a schism between agile and architecture has emerged. Only in recent years a new balance is being established, but only in some organizations that can be regarded as early adopters of a scaled-up form of agile. This paper is aimed at helping organizations – and especially architects – to bridge the gap between the Enterprise Architecture and agile practices, in order to find a balance that serves their goals better than either extremes.

[1] Note on the use of the word "agile": The word agile has evolved during the 30+ years it has been used in a software development context. In this paper, we do not add another definition. The Manifesto for Agile Software Development (http://agilemanifesto.org) refers to a set of practices, and a set of values and beliefs. Although the Manifesto is about software development, agile has become a widespread label that is used in many contexts; e.g., agile management and agile human resource management.

Relationship to the TOGAF Standard

This White Paper does not directly address practices related to the TOGAF standard. Nevertheless, ideas described here may be applied by practitioners who use the TOGAF standard as their Enterprise Architecture framework. The following sections of this paper will be very useful in a practical context: From Traditional Enterprise Architecture to Agile Enterprise Architecture: A Paradigm Shift (on page 11) and Agile Enterprise Architecture (on page 21). Concepts from this paper can be applied in all phases of the TOGAF Architecture Development Method (ADM). In the Preliminary Phase, the Enterprise Capability is established, the Enterprise Architecture team is formed, the metamodel is adapted, tools are selected, and the Enterprise Architecture processes are designed. This is the point in time when questions about style of work are answered. The Paradigm Shift section (on page 11) may be used to form the overall approach and to choose the way the architecture team is working. The Agile Best Practices section (on page 21) may be a source of practices and hints for how to apply them in Enterprise Architecture work. When doing this, it should be noted that this paper is not aiming to cover the complete standard. It describes the philosophy and some good practices in agile Enterprise Architecture that may be used. The agile practices are useful throughout the whole ADM cycle and the agile mindset is the way of thinking for approaching challenges that emerge during the whole Enterprise Architecture project or program.

Definitions

These definitions are specific in the context of this White Paper. We have included them to make sure the reader has a consistent interpretation of these terms while reading and understanding this paper. Some of the terms may have a different implied or explicit meaning in an enterprise or consulting practice. Definitions used in this White Paper are not intended to redefine industry definitions.

Agility

The ability to effectively react and adapt to changes in the environment.

Agile Approach

Employing the mindset of delivering incremental value in short intervals of time; i.e., iterations.

Agile Methods

An umbrella term for several iterative and incremental software development methodologies. While each of the agile methodologies is unique in its special approach, they all share a common vision and core values. Examples are: eXtreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Scaled Agile Framework (SAFe), and many others.

Agile Practice(s)

Multiple practices that are derived from a common philosophy. For the purpose of this paper, we are focused on the philosophy rather than a particular practice.

Agile Enterprise Architecture

Employing an agile approach to create and deliver useful architecture and support agile practices to deliver value incrementally and fast.

Minimum Viable Architecture (MVA)

An architecture that enables the delivery of the 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]

Scope and Structure of this White Paper

The scope of this White Paper is the approach used to create the Enterprise Architecture, and the approach to deliver that architecture to agile development teams. In other words: the focus is on the process rather than the architecture itself, as depicted in Figure 1.

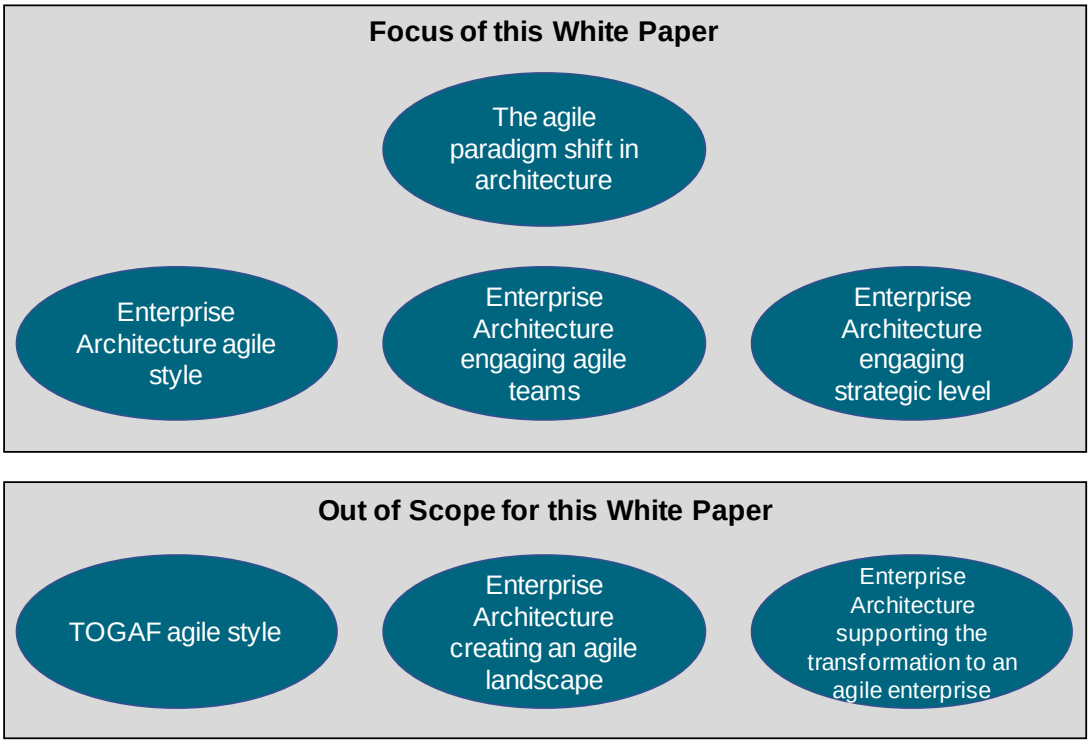

Figure 1: Scope of this White Paper

This paper focuses on delivering Enterprise Architecture using agile practice. We limited the scope of this paper to the processes. This alone is, of course, not enough to create a truly agile organization.

The overall aim of this White Paper is to reach out to different architect roles, enable them to work in a more agile way, and to upgrade their skills to deliver more value to the enterprise. After reading this paper, we

[2] Refer to: Continuous Architecture by Murat Erder and Pierre Pureur (see References).

believe the reader will be able to use the Enterprise Architecture and agile practice taxonomy with confidence in executive conversations or consulting engagements.

A Historical Perspective on Agile and Architecture (on page 9) describes how the pendulum swung from a traditional architectural approach to an agile approach, and is now swinging back to a more balanced position.

From Traditional Enterprise Architecture to Agile Enterprise Architecture: A Paradigm Shift (on page 11) lays the foundation for applying agile practices for the Enterprise Architecture practice; it will focus on key agile concepts and apply them to the architecture context, as they relate to Enterprise Architecture delivery, governance, and stakeholder engagement.

Agile Enterprise Architecture (on page 21) provides practical guidance for delivering Enterprise Architecture using agile practices; it also provides guidance for engagement with agile solution delivery teams and the business. Agile architecture manifests itself on all levels. In this paper we grouped the architecture processes as follows:

- Developing Enterprise Architecture in an agile way (Agile Practices to Create the Enterprise Architecture, on page 22)
- Engaging solution development in an agile way (Agile Enterprise Architecture Engagement with Agile Solution Teams, on page 26)
- Engaging the business in an agile way (Agile Enterprise Architecture Engagement with the Business, on page 31)

Prior Knowledge

The practice of Enterprise Architecture has been in existence since the early 1970s. It is supported by material that is contemporary and has kept pace with the evolution of business needs. The reader of this paper need not know how the practice evolved, but is expected to have a reasonable understanding of Enterprise Architecture practice and its application. Likewise, the body of knowledge related to agile practice is vast and evolving. We suggest the reader refers to sources from respective practices to understand the nuances of the agile practice. It is expected that the reader is aware of the application of agile practice. This White Paper extracts key principles of the agile practice to guide the practitioner of the Enterprise Architecture and agile method to deliver and use Enterprise Architecture in short time intervals.

A Historical Perspective on Agile and Architecture

In the early 1970s, personnel from Information Systems and Technology researched ways to analyze the business roles and processes to guide investment in business information systems and technology. These efforts evolved over several decades to develop various frameworks. Most of these frameworks were developed to address a very specific set of strategic concerns and needs of the enterprise. These frameworks did not explicitly address the sequence in which the analysis, assessment, and evolution functions of the architecture development and consumption process were to be executed. It was normal for practitioners to use whatever delivery method was prevalent in their enterprise to deliver architecture work. Through the 1980s to date, the frameworks, now called Enterprise Architecture practice, have codified an approach to deliver such guidance.

In the early 1990s, delivery of software to automate and simplify business processes could not keep pace with the rate of change of business environments. Several attempts were made to reduce the time needed to analyze, design, develop, test, and deploy software solutions (e.g., Kern[3]). All of these efforts led to the creation of the Manifesto for Agile Software Development.[4]

Agile is optimized to incrementally deliver value to the business. Agile enables the implementation team to structure their work around their productivity, and time to incremental value. Enterprise Architecture is optimized to identify the strategic drivers of an organization, and remove any barriers that inhibit the enterprise from delivering value to its customers and investors (see the TOGAF standard). Both approaches have value, but even though both aim to serve the needs of the customer, the more agile bottom-up approach and the more architected top-down approach might lead to conflicts in execution. The mindsets and paradigms are different, and without a mutual understanding and a balanced approach it will be difficult to use them together to deliver value.

Agile's Growth towards Architecture

Though a good number of organizations are using agile practices for the overall benefit of the enterprise, many are still using it within the IT department to deliver technology solutions. Many practices derived from the Manifesto for Agile Software Development are still focused on software development.

Agile practice evangelists and leading consulting organizations proposing agile start to see the value of Enterprise Architecture in preparing the ground for agile initiatives without engaging in anti-patterns like Big Up-Front Design (BUFD), while remaining isolated from the delivery teams. Large transformations or product development need to be vision-led, and should facilitate roadmap development and agile planning. This introduces a whole new set of stakeholders that can benefit from a better understanding of the context, and who need to participate in the decision-making process at an early stage to support iterative architecture and solution development. They are often missing this high-level view of the enterprise ecosystem that should be provided by the Enterprise Architecture capability and find it difficult to see some dependencies or constraints impacting value delivery. When the agile approach reached mainstream, it proved to be difficult

[3] Matthew Kern: www.linkedin.com/pulse/agile-dead-matthew-kern and www.linkedin.com/pulse/agile-dead-2-matthew-kern.
[4] Refer to: http://agilemanifesto.org.

to ensure that large systems planning, governance, and financial controls can be executed consistently or are able to meet the enterprise needs. It is not without reason that most mature agile frameworks are now mapping their concepts and methods toward strategy and portfolio layers. Without having access to these views, we cannot expect agile teams to deliver full potential value for their enterprises, where ecosystems are complex, or where we can find many legacy systems in their application portfolio.

Enterprise Architecture's Growth towards Agile

As previously stated, for several years Enterprise Architecture practice was understood as Enterprise Information Technology Architecture (EITA) practice, and was confined to the IT departments of the enterprise. When the enterprise ecosystems became more and more complex, Enterprise Architecture changed to consider a bigger view and focus on business value delivery and strategic perspective, and also to shift their approach and strategy from the traditional approach based on moving from an as-is to a to-be architecture towards a more practical and agile approach.

The pace of change has increased due to the advent of new technologies and business trends. It called for a need to develop and deliver technology and business solutions at a pace that is commensurate with market demands. Over the last five years we have seen an increasing trend in the adoption of agile practices in many transformation initiatives and in delivery of new products. Agile practice is well suited to developing Minimum Viable Products (MVPs) – a working version of the solution – rather than on completing the whole project, which could take a lot of time to complete. Customers can shape the outcome better, as they can directly work with the creator of the solution and provide feedback on the working version of a solution. Time taken to get the MVP into the hands of the customer is much shorter than the traditional methods.

Synthesis

As enterprises adopt agile practices, the pace of delivering Enterprise Architecture by most of the practitioners creates a challenge. The outcome the customer wants may not be articulated clearly, thereby making it difficult to define the target state. Agile practices look for timely guidance to incrementally add value and define the target state, as guided by the use of the MVP by the customer. The disconnect between the need to define the target state first and arrive at the target state incrementally is the source of tension between Enterprise Architecture and agile practitioners.

There is as much a critical need to adapt the agile philosophy and to adopt agile practices into Enterprise Architecture, as there is a need to scale agile up to embrace practices that have developed in mature frameworks like, for example, the TOGAF standard. Reconciling bottom-up development with enterprise direction allows an organization to have the best of both worlds: the advantage of highly-responsive bottom-up incremental change, aligned with strategic guidance and led by shared purpose and vision.

From Traditional Enterprise Architecture to Agile Enterprise Architecture: A Paradigm Shift

This section discusses the deep changes that result from adopting the agile mindset to architecture. Using the phrase "From <traditional> to <agile>" model, we try to characterize these changes in terms of architecture approach and mindset, which – as we believe – altogether constitute a real paradigm shift.

Two remarks are important here:

1. We emphasize that the term "traditional" should not be identified with any management or architecture framework.
2. Both Enterprise Architecture and agile have a track record of failures and successes. Far too many people have read their biases and superstitions into both Enterprise Architecture and agile, often based on hearsay and bad examples. When an enterprise intends to close a gap between their goals and their customers' needs, both Enterprise Architecture and agile are essential components.

This section covers the main aspects of the shift from an architecture point of view. By cherry-picking parts of this architecture shift, the Enterprise Architecture team can enhance their ability to fulfill their charter.

To describe the paradigm shift we use this structure:

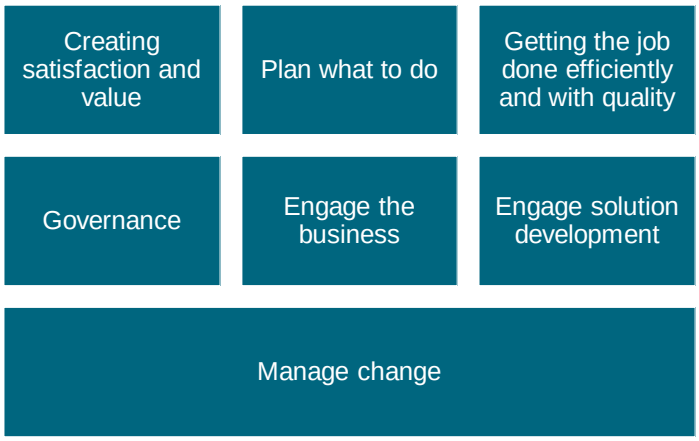

Figure 2: Aspects of the Agile Paradigm Shift in Enterprise Architecture

Even though the time-to-market and time-to-value concerns have not changed, the cycle times to achieve these targets have shrunk. In order to deliver more architectural or solution delivery value, there is a need to collaborate to gather information as quickly as possible. The environment and external triggers are also changing at a higher pace than a decade ago. There is a need to change the mindset and practices to meet this reduction in cycle times.

Creating Value

There is a clear need to start before a whole architecture is fleshed out. Instead of dropping the strategic view and delivering only based on trends without considering organization capabilities and culture, delivering

architecture in short cycles, can provide both timely results and long-term guidance. Delivering architecture in an iterative way enables fast feedback loops with solution development teams, business, and customers. It is important to understand that with architecture the iterative delivery of business value and short value-to-market cycle are improved.

Transform from:	**To:**
Inside out (company focus)	Outside in (ecosystem focus)
Completeness	Incremental value delivery
First time right	Learning by doing
IT value-based prioritization	Business value-based prioritization
Waste in architecture process	Lean architecture process

Transform from: Inside out (company focus) To: Outside in (ecosystem focus)

Due to market pressure and general trends to become more customer-centric, many organizations are transforming their business model and changing the focus of their strategy from optimizing the internal processes towards delivering value to the ecosystem and customer. This will shift the foundation of the target architecture, from internal business drivers to market demands.

Therefore, solution delivery is not limited or bounded by IT solutions, but also by outsourcing models and partnership agreements that should also be delivered following an agile approach.

Transform from: Completeness To: Incremental value delivery

One of the big selling points of an agile approach is that the business owner gets useful results as early as possible. However, many professionals – not only architects – are reluctant to show partial results early. There is uncertainty, and the first increments will be neither complete nor comply with their own professional standards. One of the significant agile phrases is "fail fast". It means checking different possibilities and different opportunities as early as possible – if they are wrong, it is better to find out sooner than later. For architects, this is true on the enterprise level, as well as on the solution level. Investigating possible scenarios with the stakeholders early has the advantage of creating a shared understanding, an understanding from which everyone can build further. The leads to a more effective process and better results.

Transform from: First time right To: Learning by doing

Another advantage of an incremental and iterative approach is that it gives us the opportunity to learn using feedback loops and course correction. In a one-shot up-front design approach, the feedback is pretty much absent to give any opportunity for course correction. Delivering incrementally, gaining feedback periodically, allows the design to be relevant and promotes continuous learning. The architecture is evolving at the speed of business and stays current with changing business needs. This is what makes many modern organizations so successful. As Arie de Geus said in 1998 (see References): "*The ability to learn faster than your competitors may be the only sustainable competitive advantage.*"

Transform from: IT value-based prioritization To: Business value-based prioritization

Even though most Enterprise Architecture frameworks state that Enterprise Architecture capability should look at every aspect of the enterprise, the reality in many organizations is that there is undue focus on IT. The customer and outcome-centricity of agile practices is a good reinforcement to the "whole business" view concept defined in the Enterprise Architecture frameworks. On the other hand, agile practices are also sometimes bounded to the IT/software space so there is an opportunity here for agile to be supported by the holistic Enterprise Architecture view.

It is all about business value. Even the long-term *versus* short-term discussion is not so much a business-IT alignment issue as a strategic decision that should be made in the proper context. Here we see agile[5] and architecture frameworks collaborate, when architecture effort is part of the planning cycle on different levels – from team level up to the portfolio level.

Transform from: Waste in architecture process To: Lean architecture process

Getting rid of waste by streamlining architecture processes takes up an essential element of lean development – the elimination of "waste". The essence of this is to analyze the Enterprise Architecture processes and streamline them by removing wasteful activities. Waste in Enterprise Architecture processes can be classified under the following categories:

- Architecture work could be overambitious, too much in one go
- Over-architecting: selecting the right scope is key and avoid going into too much detail
- Delays: people wait for information, feedback, or approvals

The removal of waste is inseparably linked to the analysis where value is generated. In other words, the Enterprise Architecture process needs to be analyzed in the context of the processes that consume the Enterprise Architecture outcomes and from the perspective of the consumers.

Planning

In any initiative it is useful to do some planning. But do we need to do a lot of planning up-front that will be changed down the line? Agile practices require different types of planning to be done to maintain the pace of delivery and value of the deliverable:

Transform from:	To:
Common budget overruns	Budgeted cost of sprint
Design for reuse	Design for use
Large time-consuming tasks	Small incremental tasks
Waterfall delivery	Agile delivery

[5] See www.scaledagileframework.com.

Transform from:	To:
Detailed planning	Rolling wave planning

Transform from: Common budget overruns To: Budgeted cost of sprint

What is true for projects as a whole is also true for the architecture effort in long-term initiatives. Initial guidance can be visionary and sketchy, but trying to do up-front architecture at any substantial level of detail is often underestimated, in time and in budget. An incremental approach gives a more realistic approach. Architecture guidance provided to the current and next episodes in a "just-in-time, just-enough" way has the typical advantages of agile: it is easier to plan (or time box); and focus on the essentials of that moment.

Transform from: Design for reuse To: Design for use

Architects are used to design with the longer term in mind. This leads to architectures that are designed for reuse, either by standardizing, or by design for change. There are many technologies and styles that can facilitate this. This approach is valuable, but it can lead to "analysis paralysis" as considering all possible future use is much harder than just dealing with current use. In an agile context, starting with an MVA and using this to gain insight and experience in what really works, will lead to better designs and greater acceptance. The price we pay is that we have to give up the "first time right" principle. Which is not that bad, given the speed of change today. It is important to consider, however, that this principle should be applied considering the organization context and specific industry. The MVA concept should be defined under this perspective to depict the minimum needed to avoid taking unnecessary risks.

With agile practices, the expectation is to deliver something of value and immediate use. This requires the architecture that supports solution delivery to be developed in increments for immediate use; refactored over time to promote reuse. This is a different mindset, however; not necessarily always better.

Transform from: Large time-consuming tasks To: Small incremental tasks

With agile Enterprise Architecture practices, the larger tasks are split into smaller manageable incremental tasks enabling transparency, adoption, and ease of change. The small tasks fit better into an overall agile way of working. A target architecture can be delivered as a detailed blueprint landscape, but also as a sequence of increments; e.g., a "big hairy audacious goal", a set of principles and guidelines, then by making the consequences explicit (this may take quite some time), then by fighting the battle to replace the MVP with a sustainable one, then by implementing the target architecture bit-by-bit, documenting it just-in-time, just-enough.

Transform from: Waterfall delivery To: Agile delivery

The idea that everything must be completed and released incrementally is a false explanation and understanding of an agile approach. Regardless of the method of planning or organizing work, or what the product is, use incremental delivery to maximum benefit. Define an appropriate time box for work increments based on the pace of change and develop the architecture that supports solution delivery within that time box. Maintain a sense of time box and incremental delivery of value that is commensurate with the rate of change of business need and subject. Not all the deliverables should be completely finished in every phase and it is not necessary to document more than is necessary in every cycle; also, depending on the cases, some Enterprise Architecture phases might not be needed and they do not necessarily have to be sequential.

Transform from: Detailed planning To: Rolling wave planning

Rolling wave planning is the process of project planning in waves as time elapses. The project team has the luxury of more information due to detailed elaboration, validated assumptions, and re-evaluated risks. Work is prioritized to address the unknowns to make the outcome milestone more reliable than the initial days of the project. Rolling wave planning is best used in an instance where there is an extremely tight schedule or timeline to adhere to with several unknowns that would have placed the schedule to result in unacceptable negative variance. Rolling wave planning[6] augments agile practices.

Getting the Job Done

Not all design decisions are architecture decisions. Architecture should be limited to the essential decisions and the detail should be left to the solution teams. What "essential" means is context-dependent. At least it should explain where the solutions fit into the bigger picture. Creating the right views to address stakeholder concerns will ensure the right levels of understanding are reached. Any architecture should be easy to explain to all stakeholders with the right views. Partitioning criteria around the level of detail is a valuable tool to employ.

The most important *raison d'être* of architecture is to be implemented and valuable. Therefore, the message it conveys and the outcome it delivers must be valued and understood by the intended audience, and it must be compelling. From a communications point of view this means it must appeal to stakeholder concerns, relate to the bigger picture, and focus on the current essential decisions to be made. In an agile context this is just as important. When the focus is on short-term value creation, longer-term concerns are easily disregarded. The architect's value is in helping stakeholders make informed decisions (what is known and unknown), to manage risk to acceptable levels. Selling the message requires subtle timing, dosing, and the right choice of medium, leaving enough room for stakeholders to connect the dots themselves.

Transform from:	**To:**
Big design up-front	Just-in-time, just-enough architecture
Decide as early as possible	Defer decisions until needed
Completeness	Deliver immediate value
Technical-oriented communication	Stakeholder-oriented communication
Working alone	Pair architecting

Transform from: Big design up-front To: Just-in-time, just-enough architecture

One of the required adaptations for architects is to let go of the idea that a complete target architecture must be architected. One of the reasons for this change is that in long transformation, business change is inevitable. The other reason is that business is able to specify details only when the system works. Between big design (and big redesign) up-front and no design up-front there is the perfect enough design up-front, which means

[6] Refer to https://en.wikipedia.org/wiki/Rolling_Wave_planning.

that some crucial things are stated and details emerge during progress. There does not have to be a big picture; often direction ("strategic intent") is good enough.

The architecture process should be "tailored" to ensure that all activities are focused on delivering valuable outputs. Cross-stakeholder collaboration should encourage a process for the ongoing monitoring and refinement of the architecture.

Transform from: Decide as early as possible To: Defer decisions until needed

From a central authority perspective, (the illusion of) complete control is ideal. However, organizational structure, geographical distribution of resources, and enabling the solution delivery to respond to changes make it imperative to delegate control to local teams. The architecture to support solution delivery should focus on communicating intent, and help the teams to arrive at new solutions within the boundaries of the accepted architecture guidelines. If a proposed solution does not comply, a decision should be made. In this way, a discussion on short-term and long-term business value can be made in a specific context.

Transform from: Completeness To: Deliver immediate value

Do we really need a complete architecture before the implementation is started? It may feel good to have a completely filled architecture repository, but this is seldom the case. There is often not enough time to reach such a state. So, focus on, for example, the initial product vision, major capabilities, value streams, and a high-level roadmap for the business transformation that enables delivery of immediate value. This is situation-dependent, and should be decided by architects and stakeholders in collaboration. Build a corporate memory (repository) that provides easy access to essential models, guidelines, and decisions, so that guidance can be provided without endlessly reinventing the wheel.

Transform from: Technical-oriented communication To: Stakeholder-oriented communication

It is possible that some of the Enterprise Architecture practices may make it hard to access the architectural work by non-architects. Good architects document the rationale behind their decisions and the artifacts. Good architecture work is supported by stakeholder analysis that results in creation of viewpoints and views that communicate the architectural work using the words of the stakeholders and the development is done collaboratively.

When the architectural work is not necessarily shared, it can be abstract, using specialized models and tools. Having a visual and transparent view on where the project is may be challenging even for a dedicated manager. An agile architecture team makes every effort to be transparent in what they do and where they stand. They do this by collaborating, communicating clearly and frequently, using techniques like Kanban boards, big visible charts, free-form architecture diagrams, and clearly articulated specifications.

Transform from: Working alone To: Pair architecting

Pair programming is an agile technique that has proven to be effective. In this model, the person developing the code and the person observing and reviewing take turns and switch roles. Architecting in pairs, following the pair program model will improve consistency, accelerate delivery, and improve quality.

While reviewing, the observer also considers the strategic direction of the work, coming up with ideas for improvement and likely future problems to address. This frees the driver to focus all of their attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.

Governance

Enterprise Architecture is about looking at the enterprise from a holistic point of view and ensuring that all architectures are related to the short and long-term strategies of the enterprise. But be aware. There is a risk that as we try and be agile we may partition the work the wrong way. Sometimes we fall into the trap of working in business silos and thus perpetuating the problem Enterprise Architecture is trying to fix.

Transform from:	**To:**
Centralized governance	Self-managed governance
Mandatory compliance	Support

Transform from: Centralized governance To: Self-managed governance

To enable good agile governance, responsibility should be delegated to the agile teams so they can self-manage and decide the best approach to deliver within the agreed tolerance and compliance criteria. This is a general principle not applicable only to agile practice. It is key to consider, however, that compliance assessments are also a key responsibility to be applied appropriately. In this context, compliance and governance criteria should be agreed at a higher level and shared to the agile teams for application. Architects play a key role in the process of providing guidance and performing compliance assessment in a dynamic way.

This also means that the skill set of people taking part in DevOps or agile developments needs to be more comprehensive, in particular for the product owner and the Scrum master. Delegate the decision-making outside the comfort zone of executive management (risk appetite). At the same time, it is easier to cancel an initiative early on the basis of an outcome related to MVA/MVP than waterfall-type solution delivery after a long design and development cycle.

Transform from: Mandatory compliance To: Support

We learned that compliance to the target architecture is mandatory. That's the whole point of a target architecture, isn't it? End of discussion. Well, not entirely. A more agile position is that the target architecture must give guidance in decision-making, without trying to pre-determine the outcome. Of course, this requires a certain level of maturity and governance: not every explanation is good enough to receive a waiver.

The target architecture needs to be defined together with the solution designers to ensure that everyone has the same understanding. And the architects need to continue supporting the solution development teams understanding the reasoning behind the architecture.

On the other hand, developers need to be aware and understand the importance of metrics and compliance criteria, so the solution delivered would really provide business value and is aligned and properly integrated with the solution landscape.

Engage Full Scope of Enterprise Architecture

Alignment of the different actors from strategy, business, operations, and IT will happen in close collaboration between business line owners, strategists, solution delivery teams, and architects. Encourage the use of stakeholder concern analysis and constant reference to contributing to business benefit.

One key thing here is to speak in a language that business stakeholders understand. Business stakeholders will have to balance rational, data supported decision-making and emotive decision-making in response to customer and user preferences. Enterprise Architects should carefully select Enterprise Architecture models to communicate the case for investment and change to avoid being too abstract. Use models and deliverables that resonate well with business decision-makers.

Transform from:	**To:**
A gap between business and IT	Collaboration end-to-end
A gap between Enterprise Architecture and development	Collaborating end-to-end
Periodic alignment	Continuous business involvement
Communicate in writing	Face-to-face conversation

Transform from: A gap between business and IT To: Collaboration end-to-end

Connecting with business stakeholders is a particularly persistent challenge, not only for architecture but also for software development. Agile approaches like Scrum and SAFe are working to address this in an effective way. Architects on all levels – from Enterprise to Solution Architects – must be fully engaged in this collaboration, adopting the same cadence, and synchronizing with cross-domain planning.

Transform from: A gap between Enterprise Architecture and development To: Collaboration end-to-end

There is a difference between Enterprise Architects and solution designers. Normally, individuals have their comfort zone in either one, but not both.

Following an agile practice along with Enterprise Architecture will help close the gaps in the skill set and focus of the roles who support solution delivery, such as solution designers and the solution delivery team. Even though the most value of Enterprise Architecture is created in providing the necessary information to guide decision-making, it should not stop there. The hand-off to the teams needs attention too. The handing over of the architecture and enablement of the solution delivery team has its reward eventually, as the time to deliver the solution shrinks, with less change management efforts and the solution is robust in meeting the needs of the stakeholder.

This is not so much a process change as a change in attitude: a more personal engagement and less documentation. Work with the teams, not for the teams. In lean terminology: take a Gemba Walk. Discover what guidance is needed. When do they need it? In what form?

Transform from: Periodic alignment To: Continuous business involvement

In agile software development, the business-IT alignment gap is crossed by continuous involvement from the business in development work. The product owner makes sure that the development work is prioritized according to business needs. All Enterprise Architecture practices should also pursue consistent and continual involvement of business stakeholders to minimize the conflicts in work prioritization and short-term *versus* long-term objective alignment. How to achieve this is a continuous challenge for both practices.

Transform from: Communicate in writing To: Face-to-face conversation

Enterprise Architectures contain large volumes of complex and inter-dependent information. Effective communication of targeted information to the right stakeholders at the right time is a Critical Success Factor (CSF) for Enterprise Architecture.

Manage Change (at solution delivery level)

Keeping pace with agile development cycles without having an effective architecture repository is hard work, but it is a trap to regard a repository as a stable source of truth. It provides a starting point for discussions, not the conclusions. The Enterprise Architecture is as dynamic as the business requires.

Transform from:	**To:**
React to change	Change is welcome
Slow pace of change	Flexibility in change
Validation when done	Validate early

Transform from: React to change To: Change is welcome

In software development as well as in architecture that supports solution delivery, changes are mostly the effect of customer feedback. When the solution is validated by the customer late in the project lifecycle, the cost of changes is high. Agile practices emphasize early and periodic feedback from the customer making it less expensive and painful to address change. To adopt agile practices, the architecture process should also undergo a change to include frequent feedback cycles with the customer.

Transform from: Slow pace of change To: Flexibility in change

Enterprise Architectures are kept vital when they evolve over time with changing goals, strategies, and opportunities. Enterprise Architecture is sometimes related to longer increments of change. Sometimes, they are yearly-based and connected to the annual budget. In other cases, even though the general architecture view or overall target architecture can be planned for a longer time, segments and increments can also be identified and related to programs, portfolios, and programs and be delivered faster. Agile Enterprise Architecture deals with changes faster than yearly adaptations. The priorities in the product backlog are constantly updated and new tasks are added instead of waiting for the next transition architecture to be started – the size of changes between transition architectures becomes smaller.

Transform from: Validation when done To: Validate early

One of the more prominent agile concepts is integration of product validation with iterations of product development. An example of such an approach in agile software development is Test-Driven Development (TDD). This concept should be an integral part of your approach to architecture development. Being able to validate your architecture early allows you to find flaws in the architecture early. But how to validate an architecture concept? Obviously, not in the same way as a piece of software.

There are methods to validate architectures – walk-throughs are one common way. A walk-through can be a simulation of different scenarios validating the architecture concepts. Another is using the Value Assessment Method from ISO/IEC/IEEE FDIS 42030 (see References).

Also, in some cases a process or an IT proof-of-concept can work. Testing early by communicating the ideas underpinning the new solution to stakeholders can give early insight into how much work it will take to get acceptance. Inviting vendors to comment on an architecture concept is another way to test it. The key point is: fail fast and improve the architecture early.

Concluding Remarks

One of the major costs while using agile practices is refactoring. Refactoring is an approach for addressing the evolving needs of the customer while simplifying the design and maintaining existing behavior. As the solutions build on each other, the cost of refactoring grows rapidly. In developing just-enough architecture and working to deliver value incrementally, the effort required to evolve should not be forgotten. Every change request to the architecture is a potential trigger to change the target architecture. When not handled properly, the architecture either becomes brittle or takes longer to respond to complex questions. Enterprise Architecture has a key role in refactoring by supporting the improvement of design through the application of patterns and standards.

Support the improvement of design through the application of patterns and solution delivery best practices to also ensure standards application and interoperability. This is key, for example, if reference models have been applied.

Indeed, agile Enterprise Architecture can be expected to become a dominant style of doing architecture for the coming decade. It will not be a solution for each and every one. The real challenge will be to find the proper balance that fits your organization. How much direction is needed up-front, and what can better be created in an emergent way?

And last, there are organizations that combine a waterfall approach with the agile approach, depending on the type of innovation. The architecture process and practice should be able to support parts of the organization that follow the waterfall approach and the parts of the organization that follow an agile approach.

Agile Enterprise Architecture: Proposed Best Practices

In the previous section we characterized the paradigm shift from traditional to agile by a series of qualitative changes in the architecture realm. In this section we provide many examples of how to implement these changes. It provides the reader with a series of agile architecture practices. A more precise way of putting this would be: "a series of examples to do Enterprise Architecture in an agile style, that fits better in organizations that do innovation/development in an agile way". In this paper we use the former expression. The objective is to inspire the reader with actionable and easy-to-implement advice. Local needs and conditions can determine which practices are followed. There is no overall framework intended, although the presence of an architectural framework (like, for example, the TOGAF framework) and a defined agile approach will provide additional support.

Towards an Agile Architecture Practice

The agile movement is based on team autonomy, but this has to be aligned with the overall vision of the enterprise and objectives communicated by top management. One frequently asked question is: on what level do architecture and agile development engage with each other? We think that this engagement should occur on all levels, from the team level up to the strategy and portfolio level. This is in line with several agile frameworks; e.g., SAFe and Disciplined Agile (see References).

Adapting an agile approach only at the Solution Architecture level ("team level") will not work. Enterprise Architecture guidance will in many instances not reach the teams in time, or in a format that teams cannot absorb. And worse, feedback from the teams will not find its way to an Enterprise Architecture backlog. Hence, the whole architecture practice must adapt, as we argued before. It is likely that the Solution Architects and Enterprise Architects will work together more closely, and that roles will be redefined. For example, Enterprise Architects can support product owners and Scrum masters with a broader context and by spotting opportunities to collaborate with other teams to create Architecture Building Blocks (ABBs).

It may be tempting to architect an agile transformation top-down, but this defeats the agile paradigm in itself. Even John Kotter (who never refers to agile)[7] proposes a direction to let the transformation be a voyage of discovery. This leaves us with finding and maintaining an ever-changing balance between thinking ahead in order to provide guidance, and collaborating with stakeholders to discover what works and what else is needed.

So how to proceed? Several agile development frameworks have been developed, each with an emphasis on different agile values and practices. The role of architecture varies widely in these frameworks. Most agile method evangelists and leading consulting organizations proposing agile have recognized the value of Enterprise Architecture in preparing the ground for agile initiatives. Large transformations or product development need to be vision-led, and should facilitate roadmap development and agile planning. It is not without reason that most mature agile frameworks map their concepts and methods toward strategy, portfolio, project, or solution delivery levels. Without having access to these views, we cannot expect agile teams to

[7] Refer to Accelerate! (see References).

deliver full potential value for their enterprises, where ecosystems are complex and where we can find many legacy systems in their application portfolio. Having access to these views is a role of Enterprise Architecture.

An agile enterprise needs an agile Enterprise Architecture team in order to stay relevant. This means a team that adopts a set of agile practices. This White Paper is trying to support that change.

Although combining several practices may work out particularly well, there is no underlying framework intended. Practices can be cherry-picked according to need. What is important is to embrace the agile mindset as described in the previous section. The example agile Enterprise Architecture practices in this section are by no means meant to be complete. They are meant as inspiration; many other practices will be invented and adopted. There is not one recipe.

Agile Practices to Create the Enterprise Architecture

This section lists best practices for creating Enterprise Architecture artifacts in a more agile way.

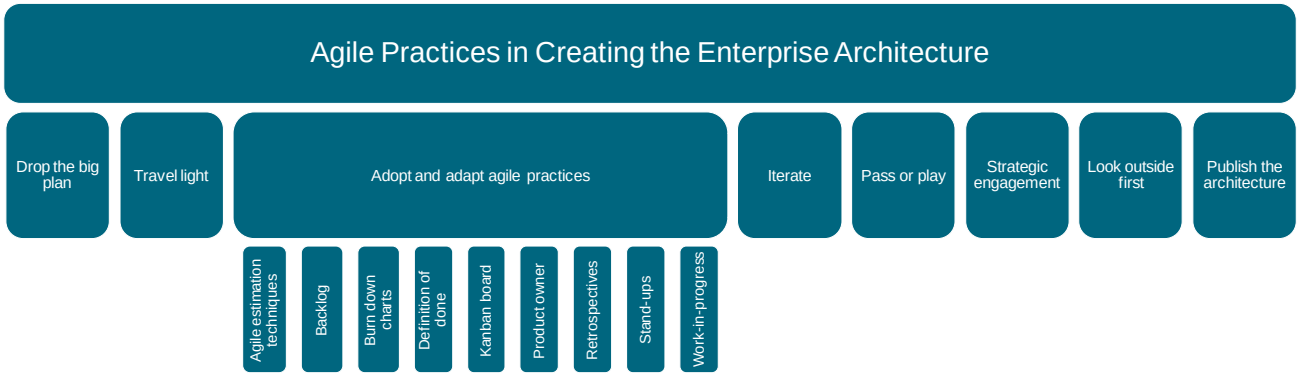

Figure 3: Agile Practices in Creating the Enterprise Architecture

Drop the Big Plan

It is important to clarify the real meaning of "Big Plan". A clear, general view of the organization landscape and capabilities is key to identifying critical milestones and to pursue effective business transformation, whether using agile practices or not. So, dropping the big plan does not mean losing the big picture, but rather avoiding delivering detailed blueprints of large-scale change programs covering a large scope.

It takes too long, and plans *will* change. What is needed is intention and direction. Enterprise Architecture can deliver both with much less effort than with BUFD. So, let's stop with designing a *Sagrada Familia*, unless we are really sure that there is a business owner for that.

Architects are well suited to define the boundaries of the effort scope, assess the complexity and high-level dependencies among key components, and then shift the scope of architectural activity, based on value and urgency. Managing complexity can be done effectively by the Enterprise Architecture maintaining backlogs at different levels of the enterprise (strategy, portfolio, project, and solution delivery).

Do not add any work to the architecture charter that does not deliver value to at least one business owner. If you think it is essential, convince the business, or put it in the backlog. If the charter is over-asking your current capacity: prioritize. Don't accept quests that keep you "out of the way" for too long. Stay on the ball!

Can you kill two birds with one stone? Can you create guidance with viewpoints that can be extended or even reused? It is hard, because communicating often requires custom-made views, but the more you limit to a small set of well-chosen and defined viewpoints, the better your chances that you can reuse them. It will make your stakeholders happier too, when they recognize the viewpoint. Of course, a good use of patterns and reference models will help to accomplish this, and even faster. Don't forget: completeness is often a non-issue: model only what is relevant for the current scope.

When stakeholders and other architecture consumers need guidance that is not available, first only investigate their motivation and objectives. Once this assessment has been done, plan carefully how the new content might be covered considering the approach defined in the points above. Then, you can put it in the backlog properly. Prioritization can be done in a regular way *later*. Stay on the ball! Work-in-progress should not be interrupted by new requests for guidance.

Most enterprises cannot afford constant updates to keep the Enterprise Architecture repository in sync with solutions developed. If we take into account the current trends of building microservices for new apps, and using cloud solutions – minimizing the impact of changes – the need for a complete repository with all the solution details diminishes. It is important, however, to have a good record of the different pieces of the ecosystem in order to conduct a proper impact assessment.

Adopt and Adapt Agile Development Practices

The collective lean and agile software development approaches provide us with an extensive set of practices that can be adopted and adapted in an Enterprise Architecture context in order to make the Enterprise Architecture practice more agile. It is our experience that, although these techniques may seem superficial at first glance, they are effective. Most of all, they make it easy for an Enterprise Architecture team that wants to transform to an agile team, to change their behavior. It also helps that the agile community understands these techniques; it tightens the gap. The reader is encouraged to experiment with these techniques.[8]

Below is a short overview of how these practices can be adopted in an Enterprise Architecture context.

- Agile estimation techniques

  The purpose of agile estimation techniques is to estimate the effort and complexity of backlog items. To prevent analysis paralysis from traditional function point analysis approaches, estimates rely more on the team's experience and group dynamics than on formal methods. Estimates get more accurate over time, and with refinement. Approaches like T-shirt sizes, Planning Poker (Scrum), or WSJF (SAFe) can be adopted for architecture work.

- Backlog

  A backlog is a prioritized list of work that should be done. The backlog development typically starts from a high-level vision or roadmap that defines a high-level view of the end architectural product and work that needs to be done. The roadmap is broken down into sets of epics, features, user stories, and work items. The Enterprise Architecture backlog should be combined or at least aligned with the top-level business backlog. In an agile context, it is especially important that the Enterprise Architecture backlog

[8] This White Paper is not an introduction to agile development practices. The reader is expected to be aware of the basic concepts of agile work.

not only contains top-down items (derived from a strategic business agenda), but also bottom-up items (feedback from teams).

- Burn-down charts

  These are charts that help to track the work that is done (burn-down). These charts also enable the architect to compare the ideal estimation with actual status. In general, the Y-axis represents the total workload still to be done; the X-axis represents time (e.g., in terms of iteration).

- Definition of done

  "Definition of done" is a useful concept for architects to make sure that both Enterprise Architect and client have a common understanding of what is needed. Often an MVA or limited increment will be sufficient, implying a process of refinement instead of a big-bang delivery.

- Kanban board

  A Kanban board is a useful tool to use in stand-ups. It also communicates the Enterprise Architecture work at-hand in a transparent way to everybody. A physical Kanban board is preferred unless the team is geographically spread.

- Pair architecting

  Pair programming is a concept from the eXtreme Programming (XP) approach. It states that the quality of software is much better when working in pairs of programmers. The same is true for architecture artifacts. Many architects feel that no architecture should be done in isolation. This is to prevent a tunnel view, to prevent a single-point-of-knowledge, and to increase credibility.

- Product owner

  Who is the product owner for the Enterprise Architecture product? Options are the CIO or CTO, but we think that the Chief Architect may be a pragmatic solution. The Chief Architect is intimately aware of the Enterprise Architecture team's chartered purpose; where different parts of the organization are in the business cycle; where there are holes in the Enterprise Architecture landscape. As the product owner, the Chief Architect can identify the architecture required by the organization.

  We do not suggest that the Chief Architect owns approval. Rather, the Chief Architect is responsible that the architecture products meet professional standards and provides guidance in line with strategic goals. They could also be the product owner, but in practice this construction fails easily, mostly due to conflicts of interest. When a conflict arises, an escalation board should be put in place. In this way checks and balances are secured.

- Retrospectives

  The retrospective is an agile project meeting organized for the team internally and it is focused on team performance and possible improvements. The retrospective is done at the end of the agile iteration. The retrospective should answer three commonly asked questions:

  — What went well (and what should we continue to do)?

  — What didn't go well (and how to improve)?

  — What is puzzling us (and how can we find answers)?

The retrospective is a best practice for any profession. Architects can participate in retrospectives of the teams they work with (recommended), but also have their own.

- Stand-ups

  For many architecture teams a daily stand-up is neither necessary nor feasible. But a weekly stand-up can help to stay informed as a team, and to maintain an Enterprise Architecture backlog. Architects that operate at large distances from each other can have virtual meetings, which are less effective, but better than none.

- Work-in-progress limit

  To keep the work-in progress limit low is challenging for many Enterprise Architecture practices. To set a limit is to acknowledge that work needs to be prioritized. Switching between tasks introduces waste and creates false expectations.

Iterate

An agile approach supported by Enterprise Architecture is to start from scratch together with your stakeholders, and lead them through the process using whiteboards and post-its. Even if the target is already known, this may be a good approach, since you acquire support along the way. Start by communicating intent. The specifications come later. There is truth in the saying "the devil is in the details", but framing the discussion early is very valuable. Aiming for perfection early in the game is doomed to fail anyway. Focus on what is needed for decisions to be made. Break decisions down into sub-decisions that can be acted upon. Of course, minimal agreement on definitions and key requirements is necessary, but let's not fall into the trap of "definition sickness".

Working in time boxes works better in an agile setting: adjust quality requirements to the cycle. In early cycles the requirements focus on direction. Later, when the repository has to be filled, modeling requirements becomes more important.

Pass or Play

Enterprise Architecture teams can easily feel overwhelmed by an organization that starts to scale agile development. So much going on, so much energy unleashed, and so hard to keep track of everything, when bottom-up initiatives are stimulated and top-down structures are not yet in place, or even dismantled.

A good practice is "pass or play".[9] Let the impact on the business decide whether you spend time on an initiative or not. Don't try to deal with everything. Pass or play requires that the architecture team has a shared understanding of what has priority and what doesn't. In that way it can speak as one in portfolio priority-setting meetings. Criteria in deciding whether to pass or to play can include:

- Business value
- Impact on Enterprise Architecture of decisions to be made
- Inter-dependencies

[9] Chris Potts; "RecrEAtion" (see References).

- What is going on outside the company (mergers, disruption, regulations, …)
- What must be done in advance, and what can best be done with the teams
- Technical quality of the solutions under assessment

Engage at Strategic Level

Architecture's prime purpose is to enhance decision-making. Without architecture support, everything after decisions are made is damage control. Architecture should be involved in doing the right things before being involved in doing things right. If Enterprise Architecture can accomplish this, it is much easier to guide solution teams, or to decide on pass or play. So, although agile solution development does require more architecture attention than in a traditional setting, collaboration at a strategic level is necessary. Leverage there is much higher, and will repay itself later on many times. It should not be a matter of choice: both are needed. Do only one and you are risking either being irrelevant or invisible.

Collaborate with Professionals in Other Disciplines

In most organizations, the architecture role overlaps with many other disciplines; e.g., quality assurance, risk management, business analysis, information analysis, IT support, or solution design. Involving these other parties to the architecture effort, delegating some tasks – e.g., requirements capture, maintaining catalogs, or matrices – can bring additional value, alleviating the workload at the same time. It also makes the Enterprise Architecture a more collaborative and agile effort, more widely accepted, and probably of better quality, since more professionals have contributed to it. Take the open approach, not the exclusive one.

Start by Looking Outside

Agility and Enterprise Architecture are enhanced when there is an open mindset to what is happening outside the enterprise and in the enterprise ecosystem – market disruptions, technology adoptions, what the competition does, industry practices, IT solutions of leading vendors, … Keeping up with all that is hard work, but inventing it all yourself is impossible. Still, it is easy to drown in internal issues, leaving no time for education and development. If everybody is involved in this industry watching, it becomes a little easier.

Publish the Architecture

It was already argued that the Enterprise Architecture should be light. The following section contains a plea to assure a warm and personal hand-off of Enterprise Architecture guidelines to the teams. However, it is very useful for teams to have a reference available, once the hand-off has been successful. Available means: easily findable and searchable, up-to-date, and with support if need be. And with an easy way to communicate findings in the repository. A well-maintained repository, published as an intranet resource, can be of tremendous value to an organization. In an agile context it provides teams with a jump-start because it prevents reinventing the wheel over and over again. For the architects it means that many first-line support questions can now be answered by self-service, freeing up time.

Agile Enterprise Architecture Engagement with Agile Solution Teams

In the previous section, we focused on a set of agile Enterprise Architecture practices. In this section, we change focus from creating Enterprise Architecture artifacts to the hand-off process: the delivery of Enterprise Architecture to agile solution teams.

Using Agile Practices in Enterprise Architecture

Why does Agile Development Require Enterprise Architecture?

In most Enterprise Architecture practices, the hand-off of Enterprise Architecture concepts and artifacts (blueprints, principles, guidelines, policies) from the plan phase to the build phase gets relatively little attention. There is an ongoing argument whether this hand-off is in the scope of Enterprise Architecture, or whether it is a Solution Architecture concern. In a lean/agile context this argument evaporates: hand-offs are a source of waste. The border between Enterprise Architecture and Solution Architecture becomes fuzzy (although the responsibilities of each discipline stay quite different), as the architecture process increments tend to become shorter. In short, agile development implies the following changes that require a more agile way to hand off Enterprise Architecture:

- Objectives change faster; architecture guidelines shift from complete up-front to just-in-time, just-enough; one consequence is that the understanding and creation of a shared vision is becoming more iterative
- Development teams are challenged to create an architecture defined by developers to depict how the system works (emergent); the architect who collaborates with the team can infuse Enterprise Architecture guidance in the process, but new documentation formats and processes should be considered for this audience (see Light Hand-off Documentation)
- New ("disrupting") technologies are adopted in a discovery style approach; hence, Enterprise Architecture guidelines are not up-front available, and the need for new policies change the way the Enterprise Architecture can be delivered to support the agile backlog: more by means of proof-of-concepts than by extensive desk studies
- When more decisions are made downstream, interaction between the Enterprise Architecture team and the solution development teams must intensify
- Quite often there is the need for an architect to support related projects (sprints) to ensure that the projects understand each others' needs; the separation between "related" and "enterprise-wide" is blurring

One of the main reasons why many architects struggle with agile is that they have their comfort zone in either Enterprise Architecture or Solution Architecture. This separation of concerns is being challenged by the agile approach. Why? There are good and bad reasons:

- A bad reason is that many developers do not understand the distinction between software architecture and Enterprise Architecture

  They see disconnected ivory tower architects and dispose of architecture as a whole. Therefore, the architecture community should get rid of the ivory tower mentality, and engage with the development effort more directly.

- A good reason is that we want to avoid hand-offs

  As the completeness of architecture documentation becomes less (as does the time to consume it), the downstream information flow from Enterprise Architecture to Solution Architecture needs to become more personal. Also, the architecture practice needs to become more responsive to the feedback from agile teams, hence the upstream information flow has to become more interactive too.

From the argument above it cannot be concluded that the Enterprise Architecture and Solution Architecture teams should be joined. What *can* be concluded, however, is that the architecture teams should adopt agile practices to stay connected. In this section we provide some agile "rules of engagement" for the Enterprise Architecture practice.

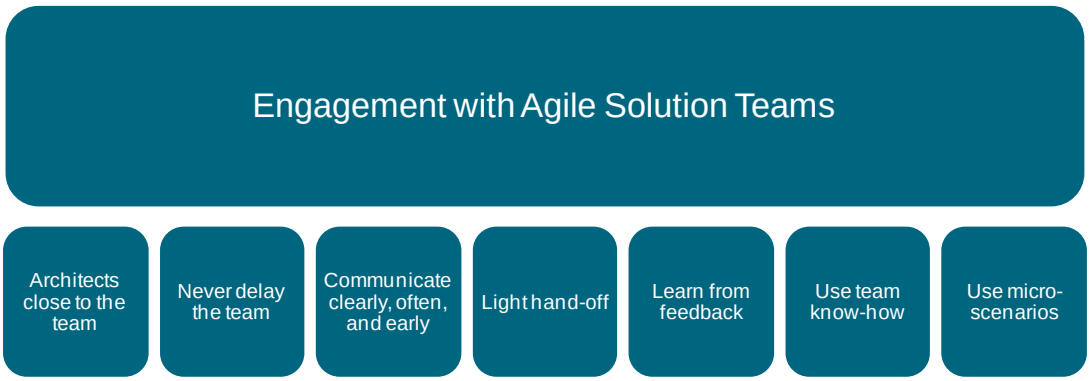

Figure 4: Engagement with Agile Solution Teams

Architects are Close to the Team

Whether architects should be *close to* the teams or *part of* a team depends mostly on the number of teams *versus* the number of Solution Architects, the maturity of the team in relation to the work, and – of course – the need for architectural guidance. For Enterprise Architects this rule of engagement is not so obvious. In organizations with few architects there may not even be a distinction between Enterprise and Solution Architecture.

In effective agile teams there is a strong team spirit and mutual trust. An architect needs to tap into that trust to be able to hand-off guidelines and participate as a trusted adviser. Trust does not come with title or rank; it must be earned. Architects can only tap into that by close collaboration. And again: close to the team can be taken literally: working in the same room or floor does help.

Never Delay the Team

In (scaled) agile circumstances teams work in a stable cadence. They need not wait for architecture decisions to be made in the same cadence. Architects must adapt to this cadence and way of working.

In this context, architects should support by:

- Explaining what they know, and what they want to learn soon
- Detecting upcoming issues well before they become blocking
- Helping the team to decompose the issue so that work can continue on other components
- Speeding up decision-making

Communicate Clearly, Often, and Early

A practice that was always important, but becomes even more so in an agile context, is to communicate clearly, often, and early. As has been shown in many research papers, teams want to understand the context

for which they develop their software, and they want to understand why there are these architectural constraints. Not to question them, but to be able to fill in the blanks. What helps:

- Inform people in smaller, helpful pieces of information
- Personal, instead of sending an architecture document or a hyperlink
- Be transparent and open (clear definition of done)
- Don't tell it, but sell it – give room for solutions to emerge
- Limit information to what they need now; don't tell everything *you* know

Light Hand-off Documentation

It is key to deliver artifacts aimed to support agile development teams – just enough architecture description to address requirements and concerns and delivered in a practical way, integrated with solution-oriented models. Here the combination of bottom-up and top-down approach is key. Key is also to scope the architecture to support only what is needed for each sprint and for every team to pursue designs that will diminish interdependencies, with less complexity so more solutions can be delivered at the same time at a faster speed.

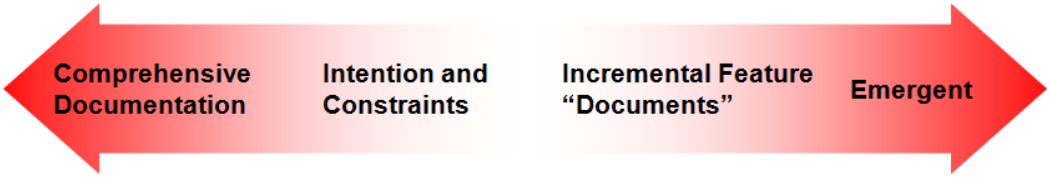

Figure 5: The Sweet Spot of Documentation is Probably in the Middle

Wherever the balance, a warm hand-off is necessary.

Good practices are:

- Limit the guidelines to the essence; if there are design specialists in the teams, use their skills to create acceptance, get better solutions, and accelerated learning
- Agree on what MVA the team needs
- Use the enterprise repository as a reference, not as a means of communication; for communication, models should be adapted to the stakeholders
- Documentation for reference by future analysts or architects (repository) should not be made by the solution team

  The team provides the design in an agreed manner. The architects take care of the modeling part; e.g., the teams do not have to know all modeling conventions, although it may help if some do.
- Deliver the architecture just-in-time, just-enough, in sprint-size chunks; the farther away, the sketchier
- Allow for some variation in the hand-off

  For example, experienced teams may need less guidance than new teams. Or, when new solutions

involve off-the-shelf software or Software as a Service (SaaS) products, discovering the "fit" in the landscape may be a joint effort.

Learn from the Feedback from Development

In the 1980s and 1990s it was considered good practice to organize a lessons learned meeting at the end of the project. Valuable insights were gathered in these meetings. In an agile context, this practice seems old-fashioned. One learning moment at the end of the project? The accelerated feedback loop in agile working is probably the most essential aspect of agile. Learning is constant, daily, per sprint, every (Enterprise Architecture) retrospective. "Fail fast" is the mantra of every innovative start-up these days. This means that both the architecture itself, as well as the architecture and governance processes, may improve continuously.

For architects, there is great value in the feedback from development teams:

- It puts issues on the radar that have been missed in the strategic top-down approach
- It shows what concepts should be clarified to the team
- When teams have a compliance issue, feedback triggers questions like: "Should we grant a waiver?", "Is this an incident or an early warning that things are changes", or "Should we amend existing guidelines or propose new ones?"

Use the Team Know-How

Projects are learning experiences. There is no reason why architects should be great in architecting or designing things they have never seen before. Respect the specialists: they are the masters of the development game. "Let's talk about how this works …" breeds better relations than "I have no experience with this, but this is the plan". Experienced architects ask the right questions rather than give all the answers.

Letting people from the agile development teams – e.g., lead developer, product owner, or Solution Architect – participate in Enterprise Architecture work, especially on new technologies, is another option to make the validation and hand-off to teams easier.

Use Micro-Scenarios

Giving up on BUFD doesn't mean that we give up on vision, strategy, or planning altogether. An incremental approach gives room to filling in the blanks just-in-time. The need for the Enterprise Architecture team to "think big" remains.

The advantage is that there is less need for big decisions. Along the way many small decisions can be taken. Architects that work a few sprints ahead of the teams can facilitate decision-making by presenting scenarios with pros and cons. An "implementation board" that decides on such matters in the same cadence as the team sprints allows for a smooth ride.

Agile working requires that development teams are empowered to define and create solutions. Even when architects and leaders are themselves subject matter experts, it is wise to give room to the team to "own" the solution themselves, and restrict guidance to intent and direction.

Agile Enterprise Architecture Engagement with the Business

In this section we focus on the engagement with the business by the architects using agile techniques. The agile doctrine is that the distinction between business, architecture, and development teams is blurring, but there is still little guidance on how to improve this engagement. A number of practices that can be adopted for collaboration with the business have been defined. We do recognize that this is a playing field where much more can be gained by adopting modern techniques like design thinking. This is, however, a body of knowledge on its own, which we address only briefly in this paper.

Visioning

Create a vision for the engagement in collaboration with the stakeholders. The vision will be based on the initial high-level epics developed in workshops with the business and solution stakeholders. An agile approach asks for direction – direction with a lot of open space where people can buy in and add to. Often, business modeling in an agile context uses a more informal approach, with more emphasis on interaction and mutual understanding, and less on comprehensive documentation and formal decision-making. Architects should identify the key choices to be made, and the consequences of the alternatives.

Accelerated Learning

Treat your enterprise initiatives as a managed backlog. Reprioritize the backlog when new initiatives are added. Focus on a small number of items in the backlog. Create small, talent-rich self-steering teams working *outside the hierarchy* to explore new knowledge areas in short initiatives. Use external experts to import knowledge fast. Architects can be useful to scope these initiatives and to recognize key learning opportunities and impediments.

Slicing the Elephant – Partitioning

Larger initiatives that require many teams and many sprints have a common challenge. How to divide the enormous amount of work into workable smaller pieces? Traditionally, slicing the elephant can be done along many different patterns: function-by-function, highest business value first, resource availability, user groups, platform-by-platform, etc. Working agile adds a new requirement: the work has to be sliced in chunks that can be handled by teams, working in parallel, in an iterative and incremental way, without creating too many interdependencies. For the architect there are even more considerations. No-regret chunks – that have value in more scenarios – first. Balancing new functionality with technical debt. This is a multi-dimensional puzzle, that is different every time, and often highly political. Architects should know the slicing patterns, since they help to influence the slicing decision.

Design Architecture for Change

The architecture needs to be built for handling change. Of course not all change can be handled, but working together with the business the architects can foresee future business developments, and combined with an understanding of future technology innovations a more resilient architecture can be created. As mentioned before, this is worth a White Paper in itself, covering technologies, principles, and patterns. Thinking about possible futures is a mature strategic capability where business visionaries and IT visionaries collaborate closely.

Collaborate Across the Lifecycle

The collaboration with the business and solution teams has to survive across the full lifecycle of the engagement from early visioning, architecting, cross-iterative solution deliveries, and into maintenance. The goal is to always understand what the business wants both in the short and long term and how that will affect the architecture and the solution.[10]

Test your Architecture with the Stakeholders (Customers)

When defining the user stories, create walk-throughs in a cross-functional setting, including business, operations, and architects. Walk-throughs where the stakeholders play different roles in the user stories can greatly enhance the understanding of the problem/opportunity. These roles can be the actors in the stories including IT system actors.

When walking through the epics and user stories it is very useful to define the high-level acceptance criteria for these. That will avoid some misunderstandings in the communication between the stakeholders and will also be a great input to the lower level tests in each increment.[11]

Agile Contracting[12]

Contracts for solution development are a complex matter, in which the legal context is by definition the dominating perspective. In many cases, business owners together with Legal and Procurement are leading in contracting a solution provider. Architecture should be involved in (strategic) contracts. The role of an architect in creating "agile contracts" can be:

- Creating awareness of the conflicts of the agile way of working *versus* waterfall type contracts
- Making sure that essential non-functionals, guidelines, and capabilities make it into the contract, and later into the solution
- Helping the business to define targeted outcomes in terms of Key Performance Indicators (KPIs) and agility CSFs

Working agile is adding another level of complexity: agile is about trust, while contracts are about distrust. Or, as the Manifesto for Agile Software Development puts it: "customer collaboration over contract negotiation". Nevertheless, over the last years many attempts were made to create contracts that suit better in an agile context. Working agile with contractors in the context of a non-agile contract is a recipe for failure. Although there is no silver bullet, there are lessons learned.

In general, an agile contract should specify:

- Goals
- Budget

[10] Refer to www.kanomodel.com.

[11] See www.linkedin.com/pulse/two-ways-add-detail-user-stories-mike-cohn.

[12] Agile contracting: www.agilebusiness.org/resources/templates-and-tools/dsdm-agile-project-framework-contract-template.

- Delivery dates
- How to collaborate
- Iteration duration
- How to deal with changes
- How to deal with contract termination

Some alternative ideas for making contracts more agile are:

- Proof of capability, then hiring teams

  You can use first delivery on a fixed-price basis as proof of the contractor's capability to deliver the ultimate solution. By using this approach, you can test your supplier and create win-win situations for both parties. Keep in mind that the win-win paradigm also assumes that the ordering enterprise will deliver its part of the deal and the contractor can require an agility assessment and business readiness of the enterprise to reach an acceptable trust level on both sides.

  Another option is to let supplier teams compete against each other. The winning supplier provides the same winning team and guarantees continuity during a longer period. The client gets the best team, but maybe at a premium price.

- Money for nothing: early ending

  One way to balance the risk for both parties is to keep the budget and time variable. When the contract is terminated early, the contractor gets an agreed fraction of the remaining contract fee. This limits the risk of the client, and allows for some relocation period of the contractor's teams.

- Change for free: fixed budget

  One way to assure team capacity, while keeping the options open for reprioritization, is to hire capacity. This allows for requirements to be refined – or dropped! – along the way.

- Small contracts

  Keep the contracts small. Large long-term contracts may have advantages in budget acquisition or priority setting, but the failure rate increases, according to the statistics. A master contract offers the flexibility to issue a new annex per release or for a few sprints, and with defined deliverables per sprint.

- No cure no pay

  For small – sprint-based – contracts, working on a no cure no pay basis may be an option.

Organizing the Innovation Dialog

- Facilitated workshop

  In complex transformations there are many stakeholders and we do not always have the luxury of clear one-person decisions to be made throughout the transformation. In fact many decisions and directions are the effect of direct interaction between different parties. A facilitated workshop is an agile practice also used in Enterprise Architecture that allows creation of high-quality decisions based on the stakeholder engagement, which gives then proper support to those decisions. This is a group process of achieving

objectives and it is led by a qualified workshop facilitator, who is independent and neutral to the workshop objectives but still focused on the workshop outcome. The products of the workshop are decisions, actions specifically assigned, and any open issues. A facilitated workshop is a practice that can significantly help to build consensus among different stakeholders and buy-in from the business.

- Other dialog formats

  Agile has a number of collaboration/brain storming patterns that can be used cross the lifecycle. Some are for handling large collaborating groups – such as Fish Bowl, 6-3-5 Brainwriting, Open Space Technology (OST), and the World Café Method[13] – and some are for structuring and selecting the input, such as Kano Models.[14]

- Design thinking

  Design thinking[15] is a creative approach for designing innovative products and services. It is based on a deep understanding of the needs and problems of the customers.

  The design thinking solution approach is predicated on the idea that identifying the right problem to solve is the single most important factor for success. Identifying the right problem to solve finds the team's intent. The "design" is a rendering of the intent and "design thinking" articulates it. But the intended outcome has to be achieved at the speed and scale required by the user. In short, design thinking is: articulating intent at the speed and scale the enterprise requires. Its principles for viewing problems are a focus on user outcomes, multi-disciplinary teams, and continuous reinvention. Its methodical foundations are (a variation of) "Inspiration – Ideation – Implementation".[16]

  Design thinking is a respected option for agile outcomes and customer satisfaction. It can support both agile Enterprise Architecture and agile delivery and therefore deserves a lot more attention than this paper provides.

[13] Refer to https://en.wikipedia.org/wiki/6-3-5_Brainwriting, https://en.wikipedia.org/wiki/Open_Space_Technology, and www.theworldcafe.com/key-concepts-resources/world-cafe-method.

[14] Refer to www.kanomodel.com.

[15] Refer to: https://en.wikipedia.org/wiki/Design_thinking.

[16] Refer to: Design Thinking, Tim Brown and IBM on Design Thinking (see References).

References

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

The following documents and links are referenced in this White Paper:

- Accelerate!, John Kotter, Harvard Business Review, 2012; refer to: https://hbr.org/2012/11/accelerate
- Arie de Geus (1998), quoted (p.128) in Creating Competitive Advantage with HRM, Lesley Partridge, 1999
- Design Thinking. Tim Brown, Harvard Business Review, June 2008; refer to: https://hbr.org/2008/06/design-thinking
- IBM on Design Thinking: www.ibm.com/design/thinking
- ISO/IEC/IEEE FDIS 42030: Software, Systems, and Enterprise – Architecture Evaluation Framework; refer to: https://www.iso.org/standard/73436.html
- Manifesto for Agile Software Development; refer to: http://agilemanifesto.org
- recrEAtion: Realizing the Extraordinary Contribution of your Enterprise Architects, Chris Potts, 2010
- Scaled Agile Framework (SAFe); refer to: www.scaledagileframework.com
- The Disciplined Agile Framework; refer to: www.disciplinedagiledelivery.com and www.disciplinedagiledelivery.com/agility-at-scale/enterprise-architecture
- The TOGAF® Standard, Version 9.2 (C182), April 2018, published by The Open Group; refer to: www.opengroup.org/library/c182
- World-Class EA: A Leader's Approach to Establishing and Evolving an EA Capability, White Paper (W160), January 2016, published by The Open Group; refer to: www.opengroup.org/library/w160

The following documents and links provide useful background information:

- Agile Fluency Model: https://martinfowler.com/articles/agileFluency.html
- Agile Study for Achieving Outcome-Based (Value) Delivery Models, CSRA: www.csra.com/sites/default/files/csra_obd_delivery_models.pdf
- Continuous Architecture, Murat Erder, Pierre Pureur, published by Morgan Kaufmann, 2015; refer to: https://www.elsevier.com/books/continuous-architecture/erder/978-0-12-803284-8
- Enterprise Transformation using a Lean and Agile Toolbox, Stefan Bente, Uwe Bombosch, Shailendra Langade: http://eam-initiative.org/file/1m30p0f28s1sk/EAM-Initiative/All-Files/20160416%20Bente,%20Bombosch,%20Langage%20Enterprise%20Transformation%20using%20a%20Lean%20and%20Agile%20Toolbox.pdf
- How to Make Agile Work for the C-Suite, Eric Garton, Andy Noble, July 2017: https://hbr.org/2017/07/how-to-make-agile-work-for-the-c-suite

- How the TOGAF® Standard Enables Agility, The Open Group Blog; https://blog.opengroup.org/2018/06/19/how-the-togaf-standard-enables-agility/
- Philippe Kruchten Blog: https://philippe.kruchten.com
- ProjectConnections Blog: http://blog.projectconnections.com/project_practitioners/2017/06/if-agile-is-better-why-is-adoption-so-low.html
- Project Outcomes: www.cio.com/article/3151122/project-management/dont-confuse-project-outcomes-with-business-outcomes.html
- RFP Patterns and Techniques for Successful Agile Contracting, SEI CMU, November 2016: https://resources.sei.cmu.edu/asset_files/SpecialReport/2016_003_001_484063.pdf
- Scrum Alliance: www.scrumalliance.org
- Test-Driven Architecture – Use your Tests to Inform Architecture, Mike Brown, June 2015: http://itsadeliverything.com/test-driven-architecture-use-your-tests-to-inform-architecture
- The End of Agile: Death by Over-Simplification: https://effectivesoftwaredesign.com/2014/03/17/the-end-of-agile-death-by-over-simplification
- The Five Dimensions of Supply Chain Agility, David M. Gligor; refer to: www.supplychainquarterly.com/topics/Strategy/20151022-the-five-dimensions-of-supply-chain-agility
- What is the Minimum Viable Product? Lessons Learned, Eric Ries, Venture Hacks Interview, March 23, 2009
- Why Agile Development Failed for Universal Credit: www.computerweekly.com/news/2240187478/Why-agile-development-failed-for-Universal-Credit
- World-Class EA: The Agile Enterprise, White Paper (W123), April 2012, published by The Open Group; refer to: www.opengroup.org/library/w123

About the Authors

Serge Bouwens, ArchiXL & ANWB

Serge Bouwens is an experienced certified architect within ANWB, the Dutch Roadside Assistance company. He has been an agile practitioner and architecture consultant for many years with several consultancy organizations. He has been working in the Netherlands helping client organizations to get their architecture practice to a higher level. As a practitioner he has gained a lot of experience in architecture, from Enterprise to Solution Architecture, traditional as well as agile. For cibit academy he has developed masterclasses in Enterprise Architecture and agile architecture. Serge's motivation for working on this paper is founded in his belief that the architecture profession needs a solid standard, and that the TOGAF® standard should embrace agile in order to remain the leading standard.

Mats Gejnevall, minnovate & Biner Consulting

Mats Gejnevall is a long-time member of The Open Group and has been involved in many different Open Group standards and guides around architecture. He has been a co-chair of The Open Group SOA Work Group since 2006. Mats works at minnovate as a mentor and architect helping organization to set up their Enterprise Architecture capabilities, creating architectures and roadmaps to transform their enterprises to adaptive future states. His work has been in all kinds of industries, from Government to Industry, from Telco to Defense.

Piotr Papros, 4IT Group

Piotr is an IT Manager with almost 20 years of experience. He has studied, worked, experienced, and improved the way teams and individuals execute and deliver their work. Piotr has worked with all parts of the IT value chain from strategy, product creation, its delivery to the client, and support and maintenance. Piotr is a big enthusiast of lean and agile methodologies. He received an IBM Agile Thought Leader award for his contribution towards agile transformation at IBM where he was acting as Agile Champion.

Mirosław Prywata, Asseco Data Systems SA

Mirosław Prywata (https://www.linkedin.com/in/miroslawprywata/) is an Enterprise Architecture and project management expert in Asseco Data Systems. He has worked in several transformation projects as architect and lead architect helping organizations to perform change. He is also an expert and accredited trainer in project management methodologies, both agile and traditional.

Łukasz Wrześniewski, 4IT Group

Łukasz Wrześniewski (http://linkedin.com/in/lukaszwrzesniewski) works as an Agile Transformation and Enterprise Architecture Consultant. He specializes in agile Enterprise Architecture and Agile Program Management. He is also an active trainer who provides TOGAF®, ArchiMate®, IT4IT™, and Scaled Agile training courses. He participates in the Agile Enterprise Architecture team and in the Architecture, IT4IT, ArchiMate Harmonization Project as an invited expert.

Using Agile Practices in Enterprise Architecture

Aleksander Wyka, ATE Enterprises

Aleksander Wyka is an ICT veteran who started his professional career in Paris in 1972. He has occupied many senior IT management positions in a number of industries including banking, telecom, distribution, manufacturing, and medical services. For the last six years he has focused on teaching Enterprise Architecture, consulting, and coaching. A long-time proponent of the stage-gate and push model of delivery, he has evolved progressively towards the pull model of delivery. Aleksander is a weathered Enterprise Architect and SPC (scaledagileframework) Certified Consultant providing advice to enterprises engaging in scaling agile methods.

Acknowledgements

In addition to the Authors (see About the Authors), The Open Group gratefully acknowledges the contribution of the following individuals in the development of this document:

- Sonia Gonzales, The Open Group
- Walters Obenson, The Open Group
- Jacek Presz, Bank Pocztowy
- Marcin Wizgrid, Accenture

About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards. Our diverse membership of more than 600 organizations includes customers, systems and solutions suppliers, tools vendors, integrators, academics, and consultants across multiple industries.

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.