TOPIC: ARCHITECTURE AND AGILE DEVSECOPS

Contributors:

Andre Schwan – Gijima
Darien Hirotsu -TachTech
Harish Mekerira - ADP
Krishna Jadhav – Tech Mahindra
Mark Williams – TachTech
Ryan Skipp—T-Systems
Shamir Charania – Keep Secure
Tom Scott—ADP

CONTENTS

EXECUTIVE SUMMARY

The IT environment has changed profoundly over the last decade. This includes changes to the way business operates and ultimately delivers value. In response to this, the technology arms of these businesses, big and small, have also had to adapt. The large separated monolithic systems view has generally been abandoned, and many technology departments have pivoted to embrace modern practices such as DevSecOps and Agile development, leveraging various cloud platforms. These approaches tend to leave the impression that architectures and solutions are "found" through trial and error, rather than proactively "designed" for fit.

Software development and operations run under Agile development, with sprints, and are evolved by "partially autonomous" DevSecOps teams. In many cases the traditional IT Architects are left scratching their heads when working with DevSecOps teams because they commonly only document the actual architecture once they have finished designing and testing. This arises from experimenting with components, functions, and patterns sourced from a multitude of options available from the internet / cloud providers.

It was the architects' role to ensure a solution was fit for purpose, described up-front, approved, planned, and budgeted. The question now becomes, "how does an architect function in a DevSecOps-driven group when the solution is unknown, or worse, unknowable", prior to implementation. Is there still a role for the architect? We identify that the nature of the architecture function must adapt. However, businesses often have neither planned the required changes nor determined how to globally ensure that their IT environment is still properly governed, sustainable, flexible, and interoperable.

With competitive expectations and the promise of "Agile development, business expects rapid solution development and feature/value delivery, even if the back-end isn't well architected up-front. So how do architects now reposition and deliver the stability, predictability, and sustainability that was always expected from their rigorous processes and governance disciplines?

This paper is written to help C-Level leaders understand and shape the evolution of the Architecture function in order that the architects continue to underpin and ensure sustained business value.

INTRODUCTION

WHAT IS AN ARCHITECT?

As with most definitions, there is a large degree of variance between the textbook definition of the architecture function and how it is used/implemented in different organizations around the world. A quick google of "What is an IT architect" will yield various definitions. For the purposes of this paper, we will look at the definition of architecture from various viewpoints. According to TOGAF, the role of an architect is to drive change that creates business opportunity through technology innovation (TheOpenGroup, 2020)

However, there are additional views to consider:

Viewpoint 1: The Business

From the business's point of view, an architect is someone that works to apply the policy or grand strategy of the business to the solutioning of problems and envision the future through the implementation of enabling technologies. They are expected to understand the "who" and "what" of the business and develop a set of capabilities (the "how") that the business will use to achieve its goals.

Typically, in an architectural framework model, these architects would be described as business or enterprise architects.

Viewpoint 2: The Stakeholders

At stakeholder level, architects typically outline a particular business problem (or opportunity) to address. This includes understanding what the stakeholder would like to achieve and the tools currently available (the "how") to solve the problem. Architecture from this viewpoint is about designing a solution to meet both the functional and non-functional requirements. It is about balancing the technical requirements of a solution with the dimensions of strategy, while considering Power, Fit, and Time as described in the Patterns of Strategy (FractalConsulting, 2022).

Typically, in an architectural framework model, these architects would be described as solution architects.

Viewpoint 3: The Team Member

From a team member perspective, the role of the architecture function is to ensure that a solution is coherent. These domain experts ensure that the solution incorporates appropriate patterns and practices as they relate to the technical decisions being made.

Typically, in an architectural framework model, these architects would be described as product or technical architects.

Viewpoint 4: IT Management

IT Leadership expects the architecture role to lead technical teams with respect to design and development of solutions, for the most efficient and economical achievement of business goals, improved frictionless technical implementations, and planning and roadmap development. This includes:

- Assistance with planning and defining roadmaps
- Assigning resources (technology, people/money/time) to various initiatives

Typically, in an architectural framework, these roles are described as enterprise architects and product owners.

Regardless of the viewpoint taken, or the prefix before "Architect", architects typically do the followings (Hewitt, 2018):

- Contain entropy by defining constraints
- Specify nonfunctional requirements for technology and services
- Determine trade-offs and facilitate decision making

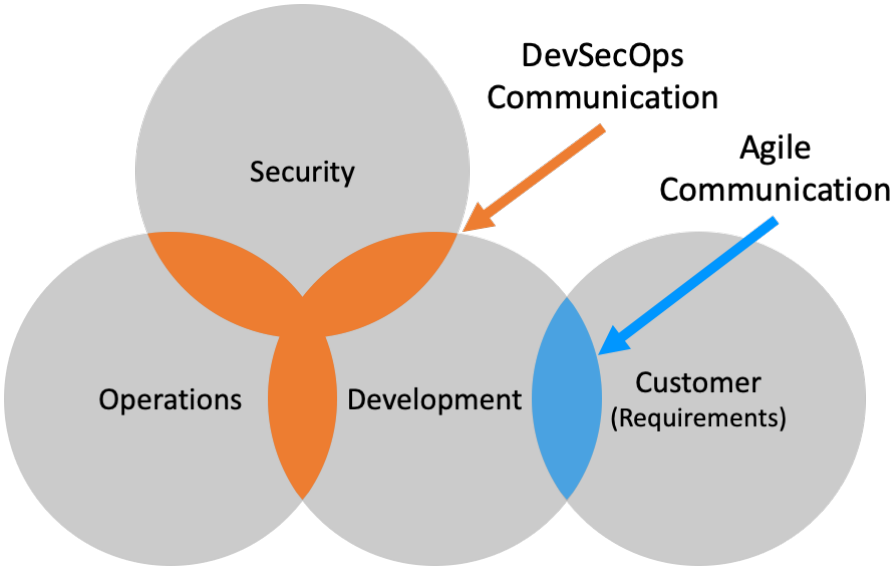
Operationally, an architect is trained to:

- Discover emerging technologies and relate them to business needs
- Determine current state, then plan and scope the "Future State"
- Translate business requirements into technology-based solutions
- Create blueprints
- Define methodologies, standards, and process
- Document a plan (up-front) for developers, operations, and engineering groups to base work or implementation on.
- Create an abstract view of the business, set up measures for success, and enable "safe-to-fail" probes and experiments for the team.

WHAT IS DEVSECOPS?

According to Red Hat, DevSecOps stands for development, security, and operations. It's an approach to culture, automation, and platform design that integrates security as a shared responsibility throughout the entire IT lifecycle (RedHat, 2018).

DevSecOps teams (where silos between traditional Development, Security and Operational teams are removed) commonly use variants of the Agile methodology which is an iterative approach to development and solution lifecycle delivery that helps teams deliver value to customers faster. Though both are different, they are highly interrelated. While Agile addresses the gap between Developers and the Business, DevSecOps aim to address the gaps between Security and the Operators. Instead of betting everything on a "big bang" launch and all designed up-front, Agile teams deliver work in small, consumable, evolutionary iterations. Requirements, plans, and results evolve and are evaluated continuously, enabling teams a natural mechanism for rapid response to change.

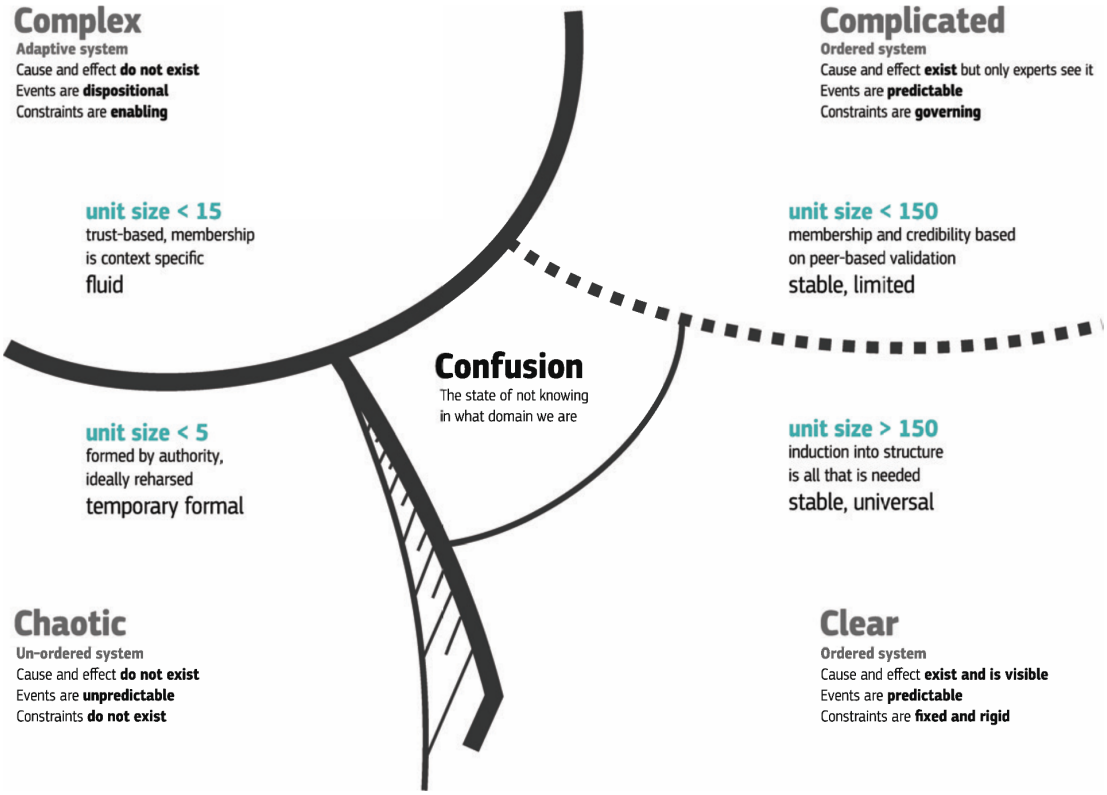


An important feature of the DevSecOps approach is the idea of creating/defining/enforcing guardrails on target solutions. This allows Agile teams to be flexible and autonomous within a defined set of boundaries that enforce security and other governance constraints.

DevSecOps favors an "as-code" approach to defining as many elements of the solution as possible. For example, tools to support DevSecOps have moved from traditional offline design and conceptual utilities (e.g., Aris) to integrated tools and Integrated Development Environment (IDE)-based visualization. These elements are then managed as code would be, benefiting from the rich set of tools and processes to govern its evolution. Policy as code can be applied on top of this paradigm and integrated with CI/CD toolchains to provide automatic attestations of code quality, compliance, and security. This leads naturally to the understanding that the architecture is *in the code* and DevSecOps teams tend to suggest the architecture can be understood by reading the code.

THE BUSINESS LANDSCAPE

Leading in to how the architecture function needs to adapt in a DevSecOps model, we need to consider that the architecture function changes depending on the business landscape or project being addressed. The first question to answer then is: "how to define what business landscape one is addressing, both overall (as an organization) and within a given solution/department (since landscapes can vary)?". Illustratively, we consider leveraging one possible framework to help with this – the Cynefin Framework:



The Cynefin Framework (Snowden and Boone, 2007)

The Cynefin framework is a decision-support framework. It is a way of determining what method or approach to adopt while critically assessing when to change it, considering the information at hand. It is based on the principle of bounded applicability: there are few if any context-free solutions but many valid context-specific ones.

Simple Contexts: The Domain of Best Practice
Simple contexts are characterized by stability and clear cause-and-effect relationships that are easily discernible by everyone. Often, the right answer is self-evident and undisputed. In this realm of "known knowns," decisions are unquestioned because all parties share an understanding. Areas that are little subject to change, such as problems with order processing and fulfillment, usually belong here.

Complicated Contexts: The Domain of Experts

Complicated contexts, unlike simple ones, may contain multiple right answers, and though there is a clear relationship between cause and effect, not everyone can see it. This is the realm of "known unknowns." While leaders in a simple context must sense, categorize, and respond to a situation, those in a complicated context must sense, analyze, and respond. This approach is not easy and often requires expertise: A motorist may know that something is wrong with a car because the engine is knocking, but the car must be taken to a mechanic to diagnose the problem.

Complex Contexts: The Domain of Emergence

In a complicated context, at least one right answer exists. In a complex context, however, right answers can't be ferreted out. It's like the difference between, say, a Ferrari and the Brazilian rainforest. Ferraris are complicated machines, but an expert mechanic can take one apart and reassemble it without changing a thing. The car is static, and the whole is the sum of its parts. The rainforest, on the other hand, is in constant flux—a species becomes extinct, weather patterns change, an agricultural project reroutes a water source—and the whole is far more than the sum of its parts. This is the realm of "unknown unknowns," and it is the domain to which much of contemporary business has shifted.

Chaotic Contexts: The Domain of Rapid Response

In a chaotic context, searching for right answers would be pointless: The relationships between cause and effect are impossible to determine because they shift constantly, and no manageable patterns exist—only turbulence. This is the realm of unknowables. Examples in this space would be the COVID-19 outbreak and the resultant lockdowns around the world that threw the business world into turmoil scrambling to find ways to continue to operate and remain in business, or a ransomware attack driving a pivot to alternate operations environments, although it quickly iterates towards "Complex". Another example could be sanctions being applied on a country or industry or an emergent technology (e.g., the advent of digital photography disrupted a global industry). The chaotic context is often in play when an organization is forced or desires to pivot significantly. A leader must first act to establish order, then sense where stability is present and from where it is absent, and then respond by working to transform the situation from chaos to complexity where the identification of emerging patterns can both help prevent future crises and discern new opportunities. Communication in the most direct "top-down" or "broadcast" manner is imperative; there is simply no time to ask for input.

THE "ARCHITECTS" CHALLENGE

Many of our current architectural frameworks and patterns emerged at the height of systems thinking and the analytical approach. Traditional processes approach design tasks by documenting customer needs and then decomposing (via the documentation of functional requirements and constraints) into design parameters. Design parameters can then be integrated (at an appropriate level of abstraction) to produce a product/solution that hopefully addresses the customer's needs.

Because systems thinking assumes that understanding the whole can be accomplished simply by having a full understanding of all of the parts [of a system], traditional enterprise's sustainability (usually) came from well-designed solutions with pre-planning, supporting processes, and governance carried out by specialized teams. They maintain products and services while carefully and slowly evolving and releasing a small amount of new appropriate capability.

There are challenges with the traditional approach. For example, how does one decompose customer needs when those customer needs cannot be described to sufficient level of detail up-front? Modern architecture must function in a business environment that promotes or requires rapid product/capability development, "fail fast-move fast" thinking, near-instantaneous response to feedback from the market, and rapid disposal of elements that do not represent business value (and which also consume valuable resources).

Another challenge focuses on the assumption that the analytical approach will work in the given context/landscape. What if understanding the parts of the system is not sufficient to understand the system as a whole? Agile methodologies are well suited here, relying on the idea that a solution can emerge over time. This causes friction with traditional architecture teams or approaches because specialist roles become frustrated with the Agile process tending towards experimentation with diverse options until an acceptable fit is found, rather than a pre-analyzed and documented design based on clear requirements, blueprints, and methodologies.

Rapid, innovative evolution of a new capability can be stifled by tightly managed processes and governance. On the other hand, agile teams may naïvely hope that a suitable architecture will gradually emerge out of weekly refactoring. This approach can sometimes invoke excessive redesign efforts, architectural divergence, and functional redundancy, increasing the complexity of the system's architecture and evolution costs.

Innovation scenarios involving solutions that are unknowable at the start are often a poor fit for traditional approaches. Solutioning within a narrow field (e.g., monolithic applications) where there is a narrow range of options and well-defined constraints such as predefined architecture patterns, interfaces, protocols, and methodologies, is very different than solutioning for an open business function where the whole public cloud's service options and elements are available for incorporation into the solution.

On the DevSecOps integration front, many organizations are allocating people from their well-structured specialist skills-based teams, into Agile groups, but often, it ends up being a superficial transition. Some Agile team members feel "lost", or the Agile team is not able to operate as a cohesive whole due to competing diverse thinking and objectives. In these scenarios one may observe short bursts of specialized activity, then disproportionate workload during subsequent sprints. This regularly happens in the task allocation for architects, where the first sprint may be considered an "Architecture Sprint", and the following sprints focus on function development tasks.

Architects can become isolated or disconnected within Agile teams, and the whole team does not carry equal load throughout and is somewhat dysfunctional. The Cynefin framework represents this problem by means of a "Complexity" illustration where only the "Clear" domain works according to the traditional "architect up-front basis"(Snowden, 2005). Architects may not yet have adjusted to the approach of providing and coordinating the use of frameworks, methodologies, and criteria (guardrails), rather than predefining final solutions.

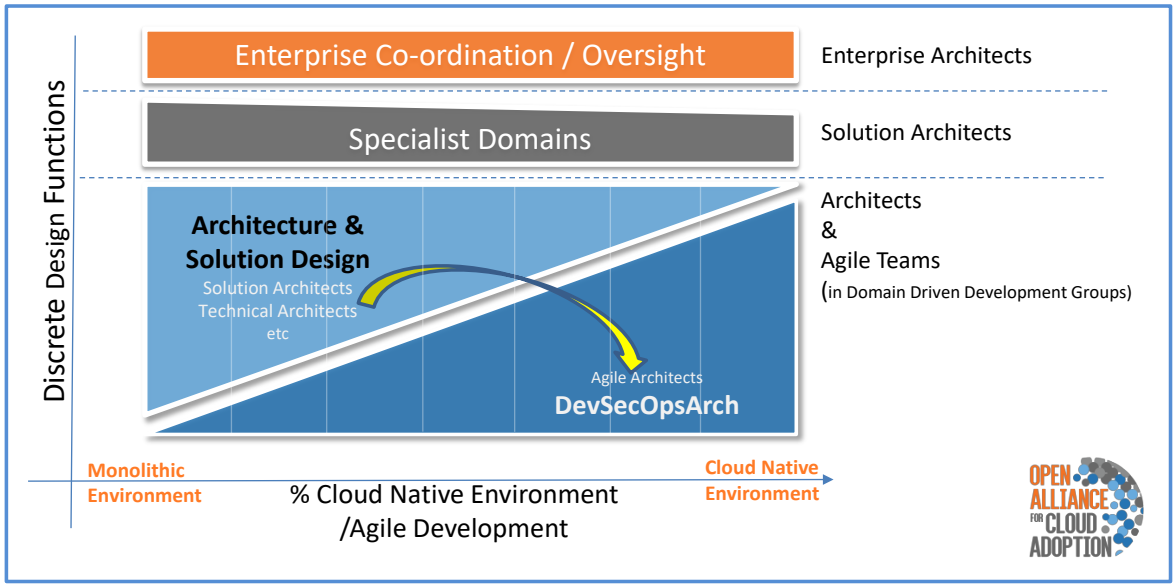
Lastly, what role do architects play when we scale the Agile approach? Some level of coordination (at the enterprise level perhaps) needs to occur across agile teams attempting to discover solutions in the same space. Further, some level of coordination is required to bring innovative ideas to enterprise scale, ensuring that solutions are architecturally sound and meet non-functional requirements (such as security and operability) prior to release to market. With large developments, architectural planning and governance becomes very important to ensure the alignment of Agile teams so that they achieve desirable organization-wide effects, align on a common target vision, and combine strategic considerations without excessive refactoring.

WHAT EVOLUTION MIGHT BE REQUIRED OF THE ARCHITECTURE FUNCTION?

While the fundamental goals of the architecture function need to be maintained in the modern enterprise, the approach to how this function is delivered must change to address the changing business landscape and embrace the DevSecOps mindset. We term the approach "Agile Architecture". Agile Architecture consists of a set of values, practices, and collaborations that support the active evolutionary design and architecture of a solution. This approach embraces the DevSecOps mindset, allowing the architecture of a system to evolve continuously over time, while simultaneously supporting the needs of existing users (Alzoubi et al., 2015)

An Agile architecture is *an architecture that satisfies the definition of agility by being able to be easily modified in response to changing requirements, is tolerant of change, and is incrementally and iteratively designed – the product of an Agile development process*, where agility is *the team's ability to create change, respond to change, and learn from change so that it can better deliver value (Waterman et al., 2015)*.

This means that Agile teams tend to orientate on business value and business functions and speed, and not on architecture principles and governance. The architects therefore need to learn to translate these artifacts into guardrails, frameworks, and methods that enable the Agile teams rather than as hurdles for them to cross.



1 – SKILLS & CULTURE:

Like the traditional approach, in Agile Architecture the Enterprise Architecture role remains crucial to the business and coordinates and facilitates alignment and common governance of multiple Agile teams and initiatives. Enterprise architecture, when performed in a disciplined Agile manner, is an important enabler of the Agile teams. Enterprise Architects should:

- Support the integration of bottom-up improvements in context of the larger IT landscape and the enterprise objectives rather than guiding a project on an overly detailed level (Proper and Lankhorst, 2014).
- Restrict EA artifacts (read "simplified") to only the essential ones and not create waste in a fast-changing environment (Hensema, 2015).
- Drive collaboration across programs and teams for following a common technological vision. Guide rather than dictate.
- Support the exchange of project experiences and lessons learned between the stakeholders.
- Maintain the design documentation and the base Enterprise Architecture model.
- Govern the EA in terms of aligned and established architectural guidelines.

There is an implied shift of power from enterprise architecture management (EAM) to the Agile teams, and the EAM role in individual teams is often only to provide a "supporting role" (Drews et al., 2017). The expectations of architects and Agile teams are not always frictionless as both exhibit antithetic ways of thinking and mindsets.

In this model, traditional Technical Architects transition towards an "Agile Architect" role. The agile architect's focus is to guide agile teams in the predetermined delivery style for a given project/initiative. This role is necessarily fluid and changes in scope/function during the solution delivery lifecycle. The Agile Architect balances intentional fostering of coaching and mentoring, enabling the development team to rapidly create products that can be easily iterated, with sustainability.

The Agile Architect must:

- adapt risk and cost-driven design methods, pragmatic modeling, and technical debt management to make effective decisions, since organizations value the architect for their ability to make the right decisions in an unclear context.
- have a clear architectural vision
- think beyond structure and technology
- deal with the structure of organization and production infrastructure
- be a good, broad communicator and display management skills which are valuable for large, distributed teams
- spend the most time with people living with the consequences of their decisions (!)
- spend time with the user to understand and interpret how the system will be used and derive the real requirements
- work closely with the product owner, who writes the final user stories for the team.
- be trained in the use of appropriate tooling such as GitOps, methodologies, process, API, ticketing & task boards, and pipeline delivery-orientated development approaches
- oversee the continuous improvement processes
- accept that "perfect" is always a target but never achieved, and near perfection may become redundant with the next iteration!
- move the team towards small, incremental development and capability evolution rather than large pivots
- understand the context, choose an implementation strategy (or combination of them), and choose appropriate tools and technologies in a combined ecosystem

- make use of continuous lean principles and practices
- focus on quality attributes and their prioritization
- exhibit deep technology understanding and insights to guide the decision making of the Agile teams
- possess sound reasoning and negotiation skills
- cede decision-making power to the team
- leave the ivory tower (!)

Solution Architects help coordinate between the Agile teams at a technical level. Examples include Security (where they help the Agile teams and developers learn and take responsibility for the security of their development).

Often, we see communication interfaces being overseen separately to coordinate data exchange and methodologies between multiple Agile teams, balancing intentionality, and emergence.

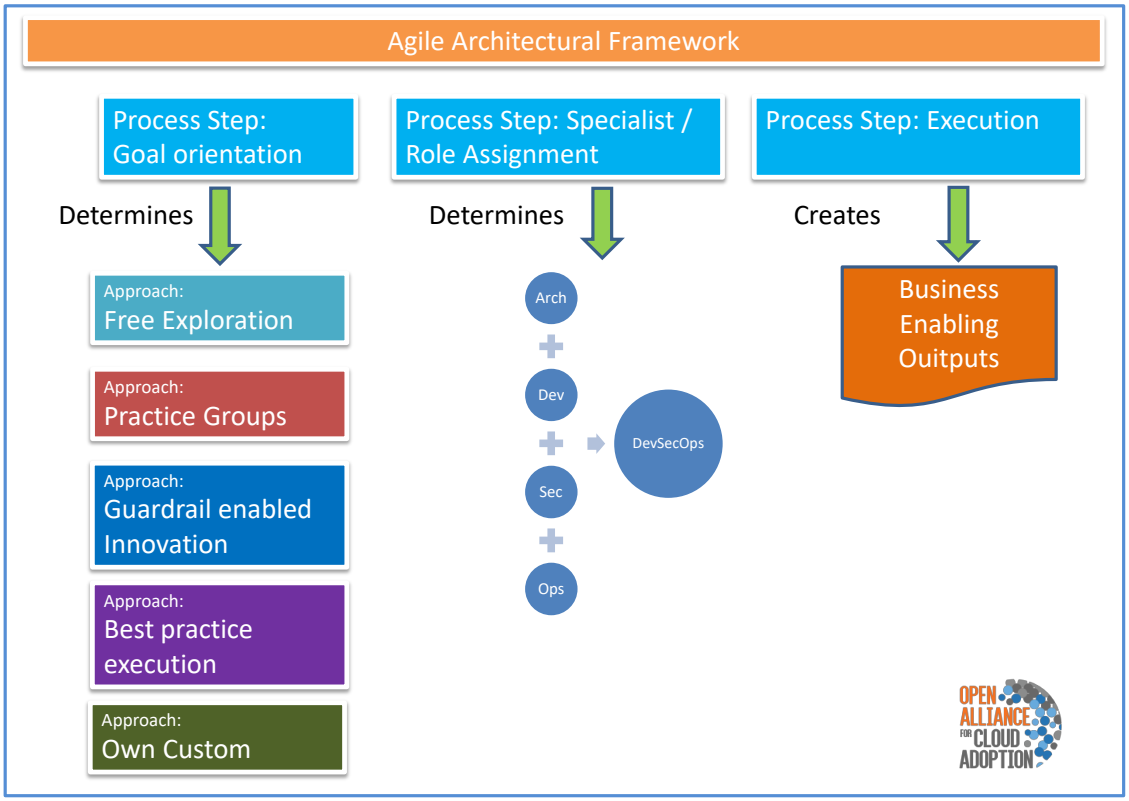
2 - ARTIFACTS:

In Agile Architecture, governance remains equally necessary, but Agile teams cannot wait for a monthly or fortnightly board approval. The architecture governance therefore needs to be rejigged to align with some of the Agile controls – the daily standups, the retrospectives, and the brainstorming processes. Further, architectural outputs on a given project may need to change. Traditionally these included defined blueprints, methodologies, and Review, Compliance & Architecture boards. Many of the controls that architecture may have used now must be adapted to support Agile teams by means of:

- defined guardrails, enabling autonomy for the Agile team between the guardrails
- compliance checks on development outputs (automated where possible, to be self-run)
- application landscape diagrams and business capability maps (sometimes referred to as domain maps) enriched by additional information, e.g., applications, technologies, infrastructure components - useful as primary architecture models. In Agile development, these often illustrate the business's transport, communications, and oversight layer for the developers that enable feedback, measurement, and reviews.
- data models, interface models, system communication models, and process models
- technical reference architectures, architecture blueprints, and architecture patterns to help Agile teams focus their thinking.

AGILE ARCHITECTURAL FRAMEWORK

There is much value in considering some of the approaches that have become common in this Architecture/DevSecOps intersection. The selection of approaches is usually led or strongly influenced by the particular problem or business landscape for which the architect and DevSecOps team is being triggered. These are illustrated with the usual progression flows as follows:



Each Approach is described in more detail below, and it is important to note that they should be adapted from organization to organization and project to project. More derivative approaches may evolve from the general approaches illustrated here.

PROCESS STEP 1: BUSINESS GOAL ORIENTATION

The aim of process step 1 is to define an appropriate solution delivery approach, and by extension, the role of the architectural function based on the business needs. When deciding the solution delivery approach, we must consider not only the business-enabling outputs required for the solution but also the business strategy and the current business landscape and goals.

EVALUATE STRATEGY

When considering business strategy, the following high-level dimensions of Power, Time and Fit should be considered (FractalConsulting, 2022).

- Power
    - Power represents “the differentials in our resource and capability and their deployment, relative to others”
    - Effectively a measure of how much ability an organization has to deliver the business enabling outputs
- Time
    - Time represents “differentials in our speed and rate of change relative to others”

    - Effectively a measure of how much time we have to deliver the business-enabling outputs. Need to consider rates of change relative to industry or competition

- Fit
    - Fit represents "the nature and attractiveness of our couplings with other actors in our environment"
    - Effectively, our ability/willingness to change to deliver the business-enabling outputs

Consider the strategy for the business and solutions at hand in terms of Power, Time, and Fit. Assess the strategy to determine constraints to the system at hand. For example, a strategic constraint might be that a solution needs to be delivered prior to a competitor. In this case, solutions should be weighted more heavily for time to market than other architectural factors.

CONTEXTUALIZE THE BUSINESS LANDSCAPE & GOALS

The next important part of this step is to contextualize the required development, to select an appropriate Approach

METHOD 1: CYNEFIN

The Cynefin framework was developed by David J. Snowden in 1999. It aims to help leaders understand that every situation is different and requires a unique approach to decision making. The framework outlines five situational domains that are defined by cause-and-effect relationships.

Establish a context and select a guiding framework for the initiative - adapted from (Snowden and Boone, 2007) Cynefin is a decision support framework. It is a way of determining what method or approach one should adopt while critically assessing when it should be changed. It is based on the principle of bounded applicability – there are few if any context-free solutions, but many valid context specific ones. This framework helps to generate a transitional path from a state of puzzlement, to a state of adaptive reaction, and then to transcend, innovate, and learn.

This (illustrative) framework helps to better understand effective approaches that one could take. For example, when an initiative is in the "Clear" domain, architecture becomes about applying the relevant best practice to the problem at hand. In this domain, a waterfall approach might be best, with a big design effort up-front (to capture various requirements) and a build phase that implements the "best practice" solution. This implementation can probably be well supported by Architecture Frameworks like TOGAF, Zachman, and others.

Contrast that with a requirement in the "Complex" domain. Here, the "right" solutions are not clear, and one needs to take a "probe-sense-respond" approach. This is in line with DevSecOps and incremental vs iterative design philosophies combined with continuous feedback processes. The role of the architect in this landscape changes from doing "big design and planning up-front" to:

- Determining safe-to-fail probes to run in parallel over a short time scale around any coherent idea or hypothesis. It may not be that any probe specifically succeeds or fails, but the result of them is to make a solution more easy to define (moving to complicated) type experiments to run (based on business criteria).
- Creating a measurement framework by which each experiment will be measured (the measure of success)
- Defining architectural guardrails
- Working with DevSecOps teams to set up and measure options and concepts

Once an experiment is deemed successful, the role of the architect shifts to focus on enterprise adoption. This includes defining the relevant architecture (or pattern) and applying artificial constraints on possible solution sets to attempt to move this particular business problem landscape onward from "Complex" towards the "Complicated" or "Clear" domains.

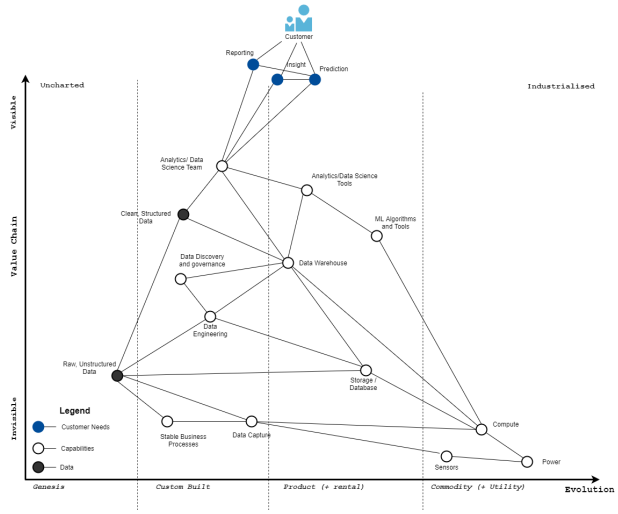
Using this method, architects need to:

- Participate in determining the context in which a particular solution needs to be delivered
- Define the constraints in place on a given solution, considering the business strategy
- Determine the best solution finding approach (example: probe-sense-respond)

METHOD 2: WARDLEY MAPPING

An option for classifying what is required to achieve the business goal could be leveraging a methodology such as Wardley maps (Wardley, 2020).

Using these maps, one can achieve a better picture of goals and classification for a solution, the forces in play, and therefore the role requirements. Depending on the objectives and components in scope, the role of architecture changes. Objectives and elements higher up on the vertical plane may involve Enterprise Architects. This offsets the Solution and Agile Architects' roles in designing component-to-component elements towards the middle of the stack. Components higher on the horizontal plane likely speak to the complexity of the objectives and the required competency of the architect, for example towards being more product focused, rather than more technically focused.



Example: Visual representation of a Wardley Mapped data landscape (Xheblati, 2022)

By making use of Wardley mapping, one can better understand the value chain of a given solution. The horizontal plane illustrates the natural evolution of components from inception where everything is uncertain, towards utility, and components becoming commoditized, i.e., from the uncharted to the industrialized.

Once the categorization, goals, and objectives have been defined, an Approach must be selected for the first iteration of the initiative. The Approach can change as the initiative goes through iterations (e.g., use of a temporary transitional

architecture when migrating from a legacy to a cloud-native application), and each approach through the iterations attracts different requirements from the architectural supporting role.

DEFINE OUTPUTS

The goals of the initiative will vary depending on the circumstances (environment) that a project/solution/initiative is being run in. It is important to note that business-enabling outputs are not necessarily full products and/or solutions. Depending on the approach taken, apart from the usual product development and software artifacts, with supporting documentation, outputs could also be results of multiple, safe-to-fail probes, or experiments which help to inform future iterations - the result of them is to make a solution easier to define (moving to “Complicated”).

A goal could also be to make the solution more ‘usable’ in the broader enterprise context. This could mean making the solution parametrized (or more generic) to support a wider range of use cases. Another example would be ensuring the solution is easily deployable and integrates with existing enterprise patterns. Doing this creates attractors towards the solution, which in turn helps increase enterprise adoption.

Examples of business-enabling outputs are:

- Vision documents
- Proofs of concept
- Safe-to-fail probes and experiments - designs / runs / outputs
- Working product

SELECTING AN APPROACH

Once the architectural goals are defined, an approach must be selected that best supports and enables the initiative. Approaches could include:

- Free exploration
- Practice groups
- Guardrail-enabled innovation
- Best-practice execution
- Own Custom

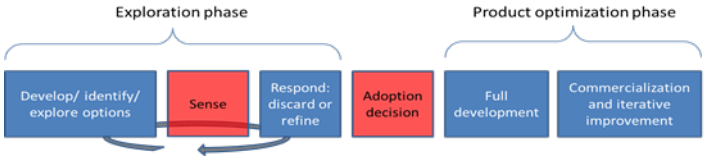
APPROACH: FREE EXPLORATION

The free exploration approach helps organizations reallocate resources and redesign processes with the goal of driving (or allowing) innovation. Free exploration aims to allow organizations to explore – meaning start journeys toward solution discovery (see cast around or side-casting in Cynefin) – as opposed to the traditional “goal-oriented”

approaches. Typically, it starts with acting (creating something new), then sensing the market response, and reacting by changing direction or developing it further. The four pillars of this approach are (Snowden and Rancati, 2021)

- Managing Constraints / Boundaries
    - Organizations must reduce hard constraints (or behaviors / routines / habits) and allow for more flexible boundaries. Flexible boundaries allow teams to explore options previously considered "out of bounds."
    - The degree to which boundaries are loosened will be determined by how radical [degree of newness (Cooper, 2011) to the organization or market] the required innovation is.
- Coordinate, do not decide
    - Organizations must shed the top-down "approval" approach of solution design, favoring instead to distribute decision making to responsible teams. Coordination between teams should still be centralized.
- Communicate by engagement
    - Create informal employee networks to help assess the situation and contribute to making decisions.
- Avoid premature convergence
    - Organizations need to resist narrowing perspective by anticipating and/or projecting the future.

As with the Agile methodology in general, innovation is managed on a succeed fast, fail fast philosophy to constrain the cost of failure. Because of this, the initial process (the exploration phase) is iterative, starting small and gaining complexity as sense and response affects the direction of development. During the exploration phase, there are very few architectural boundaries up-front and often very vague business requirements.



Because the nature of this development is purely explorative, the architectural development is completely emergent (develops from the bottom up). Once the solution is stable and final adoption decision made (product optimization phase) the architecture will be documented and formalized on the solutions and enterprise layer.

Business Challenge

Free exploration is often employed to facilitate innovation when a more radical solution is needed or to expand current products or explore potential markets and products or when the organization wants to pivot. The business identifies the need for the introduction of a completely new capability within the environment. This may be to replace a very old legacy system with completely new technology, to bring in a new business division or function, or to diversify into previously untapped opportunities.

Architectural Positioning

Managing the constraints and boundaries of the project falls under the domain of the Enterprise Architects. EAs decide how to adjust constraints/boundaries to help facilitate the project objectives along both the technical and non-technical domains. For example,

- EAs may remove or loosen boundaries around technology types, allowing developers to pick appropriate (or new) technologies.

- EAs may influence team make-up, forcing closer collaboration among different business units within the organization.
- EAs may introduce new boundaries, such as regulatory compliance requirements, to ensure compliance of the new solution.

Solution architects are key to coordination and facilitation activities across agile teams. Solution architects may "govern" multiple agile teams to help ensure there are no collisions or overlaps between them and facilitate the sharing of "learning" between the teams. As the innovations emerge through iterations, and are tested, the solutions architects evaluate the efficacy of the solution against business objectives and key metrics. (Act-Sense-Respond)

Agile architects, placed within DevSecOps teams, help to influence the initial technology and architectural decisions. They work within the constraints provided by the Enterprise Architects and take a hands-on role in solution delivery. Agile architects focus the team on addressing the problem by running safe-to-fail probes or experiments and abstracting out (but having plans for) non-functional requirements such as sustainability. As the solution emerges (and passes tests) the Agile architect works to formalize the solution, layering in non-functional requirements and ensuring the broader applicability of the solution.

APPROACH: PRACTICE GROUPS (OR COMMUNITIES OF PRACTICE)

This is a group of people who perform in similar roles or share a common interest, and it usually includes architects and developers. They meet on a regular basis to connect, align, and collaborate on methods, frameworks, standards, and ways to succeed despite working in different teams across an organization, with different mandates. Outputs from this group include the development of standards documents, guardrails, and constraints for the various Agile teams.

Practice groups primarily have a focus outside of the current scope of organizational delivery and focus on establishing capabilities or products that are new to the organization but not necessarily new to the market. Practice groups can facilitate the Probe-Sense-Respond (Complex domain) and the Sense-Analyze-Respond (Complicated domain) approaches to solution delivery. In the probe-sense-respond approach, practice groups focus on discussing plausible solutions and designing/commissioning safe-to-fail probes and experiments. In the Sense-Analyze-respond approach, practice groups work through more detailed architectural design, applying relevant practices and evaluating the solution approaches "on paper".

Practice groups can include subspecialists from a single department, but they more often include specialists from across the organization to pool learning, experience, and insight to rapidly develop capability and a control framework for the product development or the technology adoption.

Business Challenge

Practice groups are typically adopted by businesses that want to understand the best way forward (in terms of fit for the current organization) based on the current state of the organization in terms of skill, time, capabilities, etc. and the set

of practices within the industry at large. This approach is often applied when a business or a business function needs to catch up with others in the market – for example bleeding-edge organizations. During the initiation of the task, the team will need to consider how the business landscape will be influenced or changed and the required technology responses to those changes.

Architectural Positioning

The architect role in this approach is that of a facilitator: identifying the problem space, facilitating idea and option collection, then facilitating analysis, impacts, and opportunity identification. The architect will need to work with and align multiple groups, and therefore needs to display sound reasoning capability and articulate the organization's imperatives and big goals. They will also need to understand the impacts of decisions on the various business functions.

Once choices have been made, the Agile Architect then moves to supporting the implementation by the various DevSecOps teams and bringing back problems and opportunities for the practice group to consider.

APPROACH: GUARDRAIL-ENABLED INNOVATION:

In essence, guardrail-enabled innovation leverages a set of controls including tools, process, and people to ensure a programmatic approach to solving a particular business problem. One can consider the problem to be in both contexts:

- Chaotic – lots of options to choose from and skillsets available
- Complex – multiple ways to achieve the desired outcome, but with tradeoffs to be made

Guardrails provide a mechanism for teams to continue innovating while operating within the constraints defined by architects. Guardrails can take different shapes depending on the organization.

- Porous: Porous guardrails enable an organization to override them, depending on factors such as the competitive landscape. An engineering organization may choose to assume some risk such as releasing a known vulnerability to meet a commitment such as the public launch of a new service while actively pursuing mitigations.
- Boundaries: A hard boundary is one that cannot be crossed. As an example, financial institution may present boundaries such as codified policy (policy-as-code) enforced by admission control tools that prevent engineers from releasing vulnerable software altogether.

Like most architectural options, positioning of guardrail-enabled innovation is dependent on organization, use case, and risk. Consider the following organizations and teams as example scenarios.

a. A software team developing an application that processes credit card information for a global financial institution
b. A cloud operations team deploying infrastructure for a SaaS start up that only handles customer encrypted data
c. A security team tasked with software supply chain security for all developers in a Fortune 100

Porous guardrails are better positioned for scenarios (b) and (c). Scenario (a) likely has too much financial risk, so security teams will likely be enforcing boundaries that prevent developers from making mistakes given the risks

associated with not adhering to a particular architectural principle or requirement. Scenario (b) has some risk due to lost sales and reputation, but a start-up likely needs to find "product market fit," so innovation takes priority over strict adherence to architectural principles. Scenario (c) outlines a common challenge in large teams. Large enterprises have a wide variety of applications that have varying risks associated with them. As such, a security team may have to take a programmatic approach to enabling guardrails which enable teams to make decisions and accept risk as needed based on the appropriate business environment.

Business Challenge

Guardrail-enabled innovation is typically adopted by businesses who want to embrace the agile DevSecOps development process for the gradual improvement of a product/service within the organization. This method strikes a delicate balance between ensuring systems and services meet non-functional requirements (such as security, sustainability, etc.) while allowing teams to explore (within reason) novel solution delivery approaches.

Architectural Positioning

In this approach, the role of an Agile Architect is principally around developing and aligning the appropriate people, process, and technology to help developers continue meeting their objectives while enabling risk-based decision making and tradeoff analysis to eliminate chaos and complexity. However, the Agile Architect must provide more than just tool recommendations to ensure appropriate guardrails are in place.

Architecture can take various forms in this method. For example:

1. Architecture review boards that require approval of solutions prior to releasing to production environments
2. Requests for Comment (RFCs) that allow teams to review solutions to ensure architectural integrity
3. Programmatic guardrails that blend people, process, and technology to ensure adherence to a common set of architectural practices across teams
4. Codified policy (policy-as-code) managed through the software development lifecycle and implemented through admission control platforms

Once the requirement for change has been mapped based on the business landscape and requirements, Agile Architects need to decide on a viable balance between free exploration and the constraints of the current architecture. This will help to determine the type of architect to involve and whether a specific role is to be allocated or if the developers must derive an architecture after experimentation. Sometimes you need a "bus driver architect" to "take you straight there" (read: experience). and sometimes you need a "cab driver architect" who "knows the map" and helps you to make decisions as you encounter "decision opportunities".

APPROACH: BEST PRACTICE EXECUTION

This approach is usually chosen when dealing with a well-known standard environment, product set, or technology set, and it follows the more traditional architecture frameworks and methodologies. For example, it could address a well-structured upgrade, migration, or replacement of an existing business function, leveraging well documented blueprints and methodologies.

Once a target solution or state has been identified and the analysis has been done against the business requirements, the architectural function should move to implementation considerations. This includes:

- Define the target solution components based on various non-functional requirements (security, scalability, availability, etc.)
- Define the recommended deployment pattern, component setup, etc.
- Define the migration strategy

Business Challenge

The business may have identified that an older or base technology is to be enhanced, retired, too expensive to maintain, or lacking key functionality / capability. The business therefore decides to invest in changes, according to well-known techniques that others have performed previously.

Architectural Positioning

In this scenario, Solution and Agile Architects will play leading roles in enabling the DevSecOps teams. They will work with the team to define the key deliverables, target states and methods, functions, and outcomes, and how those will be achieved through a set of well-defined steps. They will also define success criteria and specific requirements to be fulfilled.

From an architectural perspective, this business problem appears to be in the 'Complicated' domain. Based on the Cynefin framework, the approach here should be sense-analyze-respond. In this case, the architectural function should focus on the following key tasks:

- (Enterprise Architects) Identify and document key target features and states based on an analysis of existing business process (however ad hoc)
- (Solution Architects) Perform a market review to determine what other businesses with this profile are doing. Which features are key in those business? What features exist in the current vendor offerings? Which vendors already meet or exceed current security/governance requirements?
- (Solution Architects) After some target architectures have been drawn up, a fit analysis is done with perceived requirements mapped against the solution features. This exercise can be largely paper based, although some trials and proofs-of-concept can be run to help address softer requirements such as usability. The outcome is a down-selection on options.
- (Solution Architects) Architectural document produced defining key requirements, target solution recommendation, costs, and deployment / release steps etc.
- (Agile Architects) Review the target design, features, and facilitate the team-driven implementation to achieve the target state, supporting and helping to identify and address hurdles.

Having analyzed the initial scenario with its requirements and objectives, and then having decided on an appropriate Approach for the particular iteration of the initiative, one needs to move on to the next Process Step – assigning skills and resources to the initiative.

PROCESS STEP 2: SPECIALIST ROLE ASSIGNMENT:

The goal of process step 2 is to ensure that the correct types of resources are assigned to the initiative. Since this paper focuses on the architectural aspects of DevSecOps, we will limit our discussion to how to determine the right resource for the architectural roles (Enterprise, Solution, Agile). Given that the enterprise and solution roles are quite static (across different projects and outcomes) we will limit our discussion only as it applies to the agile architect role.

The selected delivery approach can affect the specialist role assignment in two key ways. The first relates to the "amount" of architecture that may be required. Architecture should be an emergent property of a team/solution and may be accomplished by one or more architects (or none, depending on the constraints of the project). The second relates to the functions that the architect must perform.

For example, in "Clear" situations, architecture may be mostly done by the team lead, who effectively picks the best practice options from a set of approved solutions, and then facilitates the team to run with them to completion. In the "Complex" or "Chaotic" problem spaces, the architect may lead/run multiple DevSecOps pods aimed at producing and executing various safe-to-fail probes and experiments. While these experiments are running, the agile architect would be working on the list of trade-offs between the various solutions and measuring the experimental outcomes against the defined objectives for the initiative.

In either case above, a different set of skills will be required from the architectural role to help facilitate the project. It is important to note that the approach doesn't just influence the "role" of the architect (or the tasks that an architect does) but likely also the competence (or competencies) of the Agile Architect, which may need to be sufficiently broad to cover all technologies and elements that the solution may leverage. Further, in certain situations, Agile architects will likely need to engage specialists rather than make own decisions and need to know who to engage and how to enable them in decision making by providing context, background, and appropriate requirements per specialist area. This means that they need to have at least some knowledge and experience in almost all the possible technologies that the overall initiative will address and work with teams to discuss and influence tradeoffs.

It is common for organizations to slice themselves into smaller teams vertically and horizontally (even contracting out some layers) and create collaborating manageable (democratic) groups containing specialist capabilities and knowledge. Multiple modern terms exist for these, from "Squads, Tribes & Guilds", to "Products and Divisions", and "Industries and Functions". An example is illustrated below:

	Finance	**Manufacturing**	**Supply Chain**
Strategic: Enterprise/ Business Level	Team F1	Team M1	Team S1
Tactical: Solution Level	Team F2	Team M2	Team S2
Operational: Agile Development Level	Team F3	Team M3	Team S3

One must specify and accurately allocate skills and capabilities to each of these collaborating teams so that each is effective in its space, coordinated, and not overly staffed, overly expensive, over skilled, or out of their depth. This helps to ensure that each team is "fit for purpose" at Agile, Solution, and Enterprise levels. (This may seem obvious, but often

"generic" DevSecOps teams are assumed to be able to deal with any problem space simply because they are a good team, which might not be the case).

Effectively, when performing specialist role assignment, we are trying to understand the nature of the team (architects, DevSecOps, etc.) that will hopefully have the best chance of success given the context.

For smaller environments and less-complex projects this process might be intuitive based on understanding of the current personnel compliment and the project at hand, but in larger environments and complex projects, the most effective way to do this repeatedly is to have standardized approaches in place.

To speed up the process, establish a clear view of the characteristics, experience, skills, and areas of expertise of all the architects in the environment. This can be established through typical talent management assessment models, adapted to architectural management.

Knowing what expertise exists can then help determine what profile is needed in each layer of the architectural structuring and if gaps exist. This can be achieved by using approaches like ASHEN (Snowden, 2005) or having standard domain-based profiles (in the case of Domain Driven Development models).

Below is a short summary of each of the methods mentioned.

ASHEN

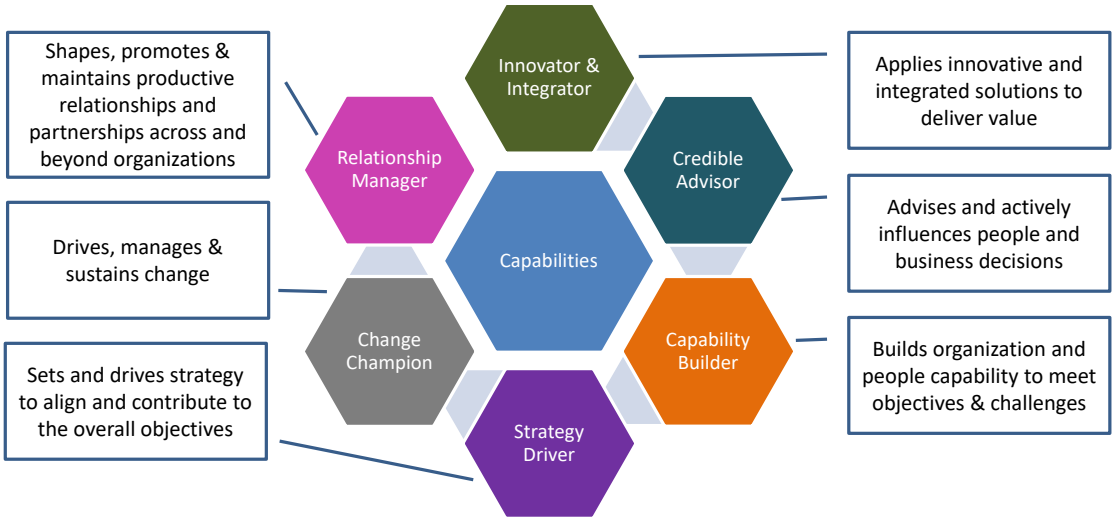
One method that can be used in this space is the ASHEN framework. ASHEN is a mnemonic that stands for artifacts, skills, heuristics, experience, and natural talent. One could use this framework to define the requirements for each ASHEN category, per team, and then set those as selection criteria for participating team members, and for supporting functions to the initiative or task. For example, when working in a novel space (approach: free exploration), it would be advisable that the team exhibit more diversity in the ASHEN elements, particularly the ones related to experience and natural talent (Snowden, 2005).

DOMAIN DRIVEN DEVELOPMENT

Another method in this space is Domain Driven Development (DDD). In this method, work is done to functionally break down a problem/project into a small enough domain that it can be addressed by a single team. The company may already have or may establish Domain Groups, with members participating from various teams, to discuss domain-specific ideas and challenges. Although scalable Agile frameworks support cross-team coordination and communication, they tend to lack detailed advice on how to do architecting in large scale Agile programs. DDD can provide basic ideas, concepts, and options for the architecture that can be beneficial not only to the Agile teams but the program overall. Business can profit from strategic DDD where architects are involved, while Agile teams profit mostly from the tactical component (Uludağ et al., 2018). Every domain can have standard architectural resource profiles predefined that are most likely to be required.

TALENT MANAGEMENT

Effectively talent management can be utilized when adding singular architecture roles to an already existing team (team augmentation). Multiple frameworks exist for specifying skills & capabilities, which may be used, but they generally consider the following requirement profiling characteristics:



Adapted from the NTPS Capabilities-and-Leadership-Framework (NTPS, 2018),

From there, identify more detailed characteristics of the architect (and for the team members), such as "What kind of activities must be lead and performed?":

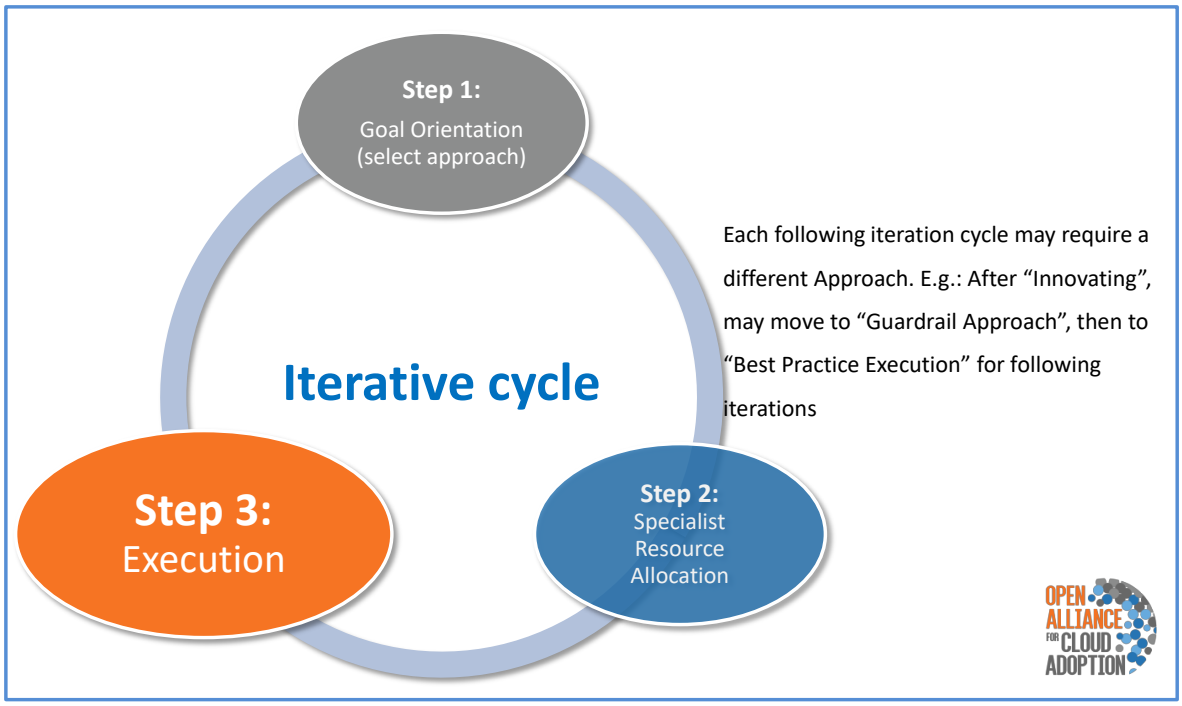
- Identify core competencies needed for the overall initiative – this helps support the businesses values and missions. This will apply to all roles in the team.
- Identify common competencies – this will relate to specific roles/jobs within the team. For example, the core competencies required in management roles would be different to a graduate starter. They would include strategic awareness, team leadership, or how they manage performance.
- Specify role-specific competencies – this is usually related to technical expertise that is required for a specific role. Essentially, this means assessing the depth and breadth required for that skill, capability, and knowledge. For example, software engineers will be versed in different programming languages – how many different languages and systems and how long they've utilized them will be a consideration.
- Define the required leadership competencies – this means the required skills and behaviors that contribute to performing as a leader or team facilitator and identifying if the candidate has the skills to potentially become one. One of the key areas of development in recent years has been about creating leadership opportunities internally. Understanding which leadership competencies are specific to the company's culture will help guide the framework and tailor it to provide a wider company advantage over time.

Identify "meta" competencies - these are competencies that relate to high-potential individuals, also considering where their competencies would be required in the future. This is when organizations are looking for something very specific in the next five to ten years, typically to fulfill long-term senior posts.

By carefully analyzing the structure of the supporting and participating teams, one is able to define and create a capability in the DevSecOps team that should be able to achieve the objectives of the initiative cost effectively, efficiently, and with reduced need for iterative evolutions.

PROCESS STEP 3: EXECUTION

Execution may consist of several cycles or "sprints", a la Agile. It is important to note that this may be an iterative process, and as iterations occur and the maturity of the solution/initiative evolves, so too the Approach being used may change for each iteration.



Earlier iterations usually provide more opportunity for innovation and exploration, while later iterations often leverage the capability that was created in the earlier cycles, but within more well-defined boundaries. The choice of approaches over multiple iterations is sometimes influenced by the strategy of initially releasing a Minimum Viable Product (MVP), then releasing more complex features later – taking more time and effort to develop depending on market response to the MVP. As the maturity of the business function increases, it is common for the approach applied to be adjusted to achieve speed of capability release and development efficiency. So, for each iteration or release, consider whether the same Approach from the Framework is still relevant or whether an alternative Approach may be more effective or appropriate.

CONCLUSION

A transformation of skills, culture, and people are usually required within an organization to enable them to effectively adopt cloud technology and the new development methods that accompany it. To really recognize all the benefits, leadership need to strategize, plan, and prepare their organization for it. Failing to transform the culture, roles, and skills of the organization may result in isolated or competing functions and less effective strategic achievement.

When leaders understand the changes required to achieve the target state as the organization adopts Cloud (Native) and DevSecOps working models, they realize that the Agile Architect also must be part of the DevSecOps model for it to be successful. This empowers architects to be better enabled to carefully prepare and place resources so that value can

be maximized. Just as the expectation is that Cloud Services enable increased agility, competitiveness, and business functionality, it must also be realized that these things are achieved by the architects within the organization working with the new technologies and methods and that they need to be positioned, prepared, and enabled for it.

What we see is that the approach to architectural support, processes, and management cannot be static anymore. It must become more fluid and adapt to the environment and project at hand to be more effective. Each business is different, and depending on whether the organization is (illustratively) in the banking or the baking world and depending on their specific application types and mixes, the "methods" applied will influence or even dictate the most appropriate architectural "Approach" that should be considered. The process proposed, combined with the methods and approaches described in this document, should help guide decision makers and leaders towards achieving increased success.

Improvements in operational effectiveness arise for a business in conjunction with their technology advancements. Agile Development and DevSecOps are effective when the business proactively considers and develops the dynamic capabilities needed to realize the potential of the methods and technologies they adopt. The architects and their skills must adapt – just putting them together in new groups does not automatically produce new results. They need to learn the new techniques, and each of their functions needs to evolve with them as the capability of the business is expected to evolve. This requires deliberate "change management" to be applied throughout the organizational layers. The new paradigms require improved feedback mechanisms to enable the teams to actively navigate the complexities and make effective choices, while not slowing down or inhibiting growth.

Sub-layer specialists, such as network, compute, and storage architects are still required. The Agile Architect is a broad role rather than a deep role, facilitating team effectiveness, navigating barriers, enabling appropriate decision making, solving problems, and empowering the team/s by providing decision frameworks. This enables the teams to build relevant and sustainable solutions to meet the evolving business requirements.

Organizations can replace the illustrative methods and approaches illustrated in this paper with others more suited to their reality. What is critical is that the organization realizes that the architectural process is not a "one size fits all" and that they need a process through which to decide which approaches to apply to a specific project or business initiative. Each scenario might require a different combination of skills and experience in the architectural layers and adapt accordingly.

This capability and dynamic approach are core to the future success of the organization, their decision making, planning, and effectiveness going forward.

REFERENCES

Alzoubi, Y. I., Gill, A. Q. & Al-Ani, A. 2015. Distributed Agile Development Communication: An Agile Architecture Driven Framework. *J. Softw.,* 10(6)**,** pp 681-694.

Buckl, S., Matthes, F., Monahov, I., Roth, S., Schulz, C. & Schweda, C. M. Towards an agile design of the enterprise architecture management function. 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops, 2011. IEEE, 322-329.

Cammin, P., Heilig, L. & Voß, S. Assessing Requirements for Agile Enterprise Architecture Management: A Multiple-Case Study. Proceedings of the 54th Hawaii International Conference on System Sciences, 2021. 6007.

Cooper, R. G. 2011. *Winning at new products: Creating value through innovation*, Basic Books.

Dragičević, Z. & Bošnjak, S. 2019. Agile architecture in the digital era: Trends and practices. *Strategic Management,* 24(2)**,** pp 12-33.

Drews, P., Schirmer, I., Horlach, B. & Tekaat, C. Bimodal enterprise architecture management: the emergence of a new EAM function for a BizDevOps-based fast IT. 2017 IEEE 21st International Enterprise Distributed Object Computing Workshop (EDOCW), 2017. IEEE, 57-64.

FractalConsulting. 2022. *Patterns of Strategy* [Online]. Fractal Consulting. Available: https://www.patternsofstrategy.com/ 2022].

Hensema, M. 2015. *Applying agile in enterprise architecture.* University of Twente.

Hewitt, E. 2018. *Technology Strategy Patterns: Architecture as Strategy*, O'Reilly Media.

Horlach, B., Drechsler, A., Schirmer, I. & Drews, P. Everyone's Going to be an Architect: Design Principles for Architectural Thinking in Agile Organizations. HICSS, 2020. 1-10.

NTPS. 2018. *Capabilities-and-Leadership-Framework* [Online]. Northern Territory Government. Available: https://ocpe.nt.gov.au/__data/assets/pdf_file/0014/243302/NTPS-Capabilities-and-Leadership-Framework.pdf.

Proper, H. & Lankhorst, M. M. 2014. Enterprise architecture-towards essential sensemaking. *Enterprise Modelling and Information Systems Architectures (EMISAJ),* 9(1)**,** pp 5-21.

RedHat. 2018. *What is DevSecOps?* [Online]. Available: https://www.redhat.com/en/topics/devops/what-is-devsecops#:~:text=DevSecOps%20stands%20for%20development%2C%20security,Integrate%20DevSecOps%20into%20Kubernetes%20environments.

Snowden, D. & Rancati, A. 2021. *Managing complexity (and chaos) in times of crisis. A field guide for decision makers inspired by the Cynefin framework*, Luxembourg: Publications Office of the European Union.

Snowden, D. J. 2005. *The ASHEN Model: and enabler of action* [Online]. Cynefin. Available: https://cynefin.io/wiki/ASHEN.

Snowden, D. J. & Boone, M. E. 2007. A leader's framework for decision making. *Harvard business review,* 85(11)**,** pp 68.

TheOpenGroup. 2020. *TOGAF* [Online]. Available: http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap52.html.

Uludağ, Ö., Hauder, M., Kleehaus, M., Schimpfle, C. & Matthes, F. Supporting large-scale agile development with domain-driven design. International Conference on Agile Software Development, 2018. Springer, 232-247.

Uludag, Ö., Kleehaus, M., Reiter, N. & Matthes, F. 2019a. What to expect from enterprise architects in large-scale agile development. *A multiple-case study***,** pp.

Uludag, Ö., Nägele, S. & Hauder, M. 2019b. Establishing architecture guidelines in large-scale agile development through institutional pressures: A single-case study. pp.

Wardley, S. 2020. Wardley Mapping, The Knowledge: Part One - Topographical Intelligence in Business. *In:* Craddock, M. (ed.). Amazon Digital Services LLC - KDP Print US.

Waterman, M., Noble, J. & Allan, G. How much up-front? A grounded theory of agile architecture. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015. IEEE, 347-357.

Xheblati, E. (2022). Understanding the Data Landscape and Strategic Play Through Wardley Mapping. Retrieved from https://www.ergestx.com/data-landscape-wardley-mapping/