# Grounded Architecture

## Redefining IT Architecture Practice in the Digital Enterprise

Željko Obrenović

# Grounded Architecture

## Redefining IT Architecture Practice in the Digital Enterprise

Željko Obrenović

This book is for sale at http://leanpub.com/groundedarchitecture

This version was published on 2023-06-04

# Table of Contents

# Introduction

image by fda54 from pixabay

**IN THIS SECTION, YOU WILL**: Understand what this book is about and how to use it.

***KEY POINTS**:*

- *This book aims to share my approach to running an IT architecture practice in larger organizations based on my experience as Chief Architect at AVIV Group, eBay Classifieds, and Adevinta.*
- *I called this approach "Grounded Architecture," highlighting the need for any IT architecture function to stay well-connected to all levels of an organization and led by data.*
- *I also explain my motivation to write this book.*

This book aims to share my approach to running an IT architecture practice in larger organizations based on my experience as Chief Architect at AVIV Group, eBay Classifieds, and Adevinta. I called this approach "grounded architecture," highlighting the need for any IT architecture function to stay well-connected to all levels of an organization and led by data.

Grounded Architecture has two main parts:

- **Structure**, where you will learn which elements you need to create in an organization to run a Grounded Architecture practice, and
- **Reflections**, where you will learn guiding principles that can help you put the ideas of Grounded Architecture into practice.

Figure 1 shows the structure of the Grounded Architecture consisting of three elements:

- Data Pillar,
- People Pillar,
- Architecture Activities Platform.

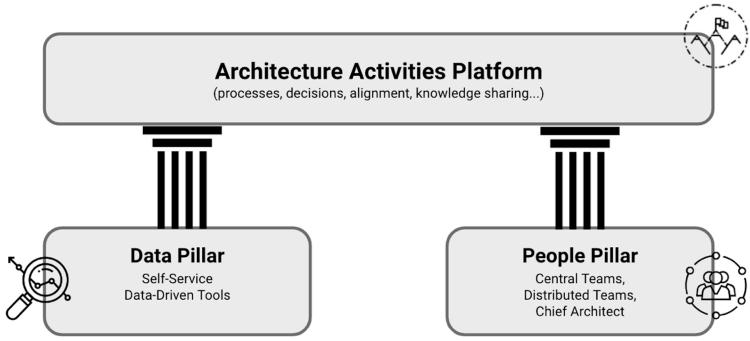



***Figure 1**: The structure of Grounded Architecture.*

The *Data Pillar* ensures that architects' decisions are data-informed, based on an up-to-date and complete overview of the organization's technology landscape.

The *People Pillar* represents a strong network of people doing architecture across the organization that is crucial to ensure that architecture function has any tangible impact.

Lastly, the *Architecture Activities Platform* defines a set of processes and agreements enabling architects to do everything architecture typically does, leveraging data and people pillars to create a data-informed, organization-wide impact.

As a part of my work on Grounded Architecture, I also provide key lessons I learned while developing ideas of Grounded Architecture in practice, grouped into two parts:

- Being an Architect:
  - Architects as Superglue
  - Skills
  - Impact
  - Leadership
  - Architects' Career Paths
- Doing Architecture:

- Culture Map: Architects' Culture Mindfield Compass
- Managing Organization Complexity: Six Simple Rules
- Product Development and The Build Trap
- Architecture Governance: Mandates, Taxation, Nudge
- Economic Modeling: ROI and Financial Options

The rest of this book will explain in detail the grounded architecture approach. In this section, I want to tell a few things about my motivation to write this book.

## What Will you Learn?

The three parts of the book (Structure, Being Architecture, and Doing Architecture) correspond to the aspects of work of Chief Architects or Heads of Architecture that need to set up and run modern IT architecture practices:

- Create organizational and technical structures to support architecture work,
- Define architecture roles and responsibilities, skills, and career paths,
- Operate effective architecture practice in complex multicultural organizations.

I invite you to read this book from beginning to end, following the progression from data and basic structures to management and organizational topics. However, you can also browse the text and start reading whatever interests you. I use many illustrations to create memorable pictures that you can associate with discussed topics, making the book usable across your organization as a coffee table book.

I have organized my lessons and insights in a form that, if you recognize the problems and are inspired by solutions, could use as a high-level "playbook" about how to work as an architect or run an architecture practice. I also provide more concrete tips on each discussed topic, finishing each section with questions you should consider when addressing these topics.

This book is not technical. We will not discuss the details of public cloud design patterns, security, reliability, how to optimize computer loads, or select the proper data storage. As a modern architect, you will need these skills, of course, but there are already many great resources for them. This book is about expanding your horizons to apply your technical skills in complex organizations.

Or to broaden your horizons as a head or manager of architects, to organize and support architects to use their technical skills more effectively as a team.

## Is This A Proven Method?

Like with many similar books, you may be disappointed if you are looking for a scientifically proven "method" of running a modern architecture practice. This book is personal and opinionated, building on my daily experiences as an architect and running an architecture practice. While subjective, I believe this book provides valuable insights for IT architects, their managers, and people working with architects. I invite others to share the lessons they have learned similarly. Even if opinionated and limited in scope, such practical reflections based on concrete examples have much more value for practitioners than abstract debates, formal methods, or academic analyses.

While not scientific, my Grounded Architecture approach builds and borrows many ideas successfully applied in practice. Gregor Hohpe's Architecture Elevator[1] view of architecture has heavily inspired my work. In many ways, my work reflects the lessons learned from implementing Gregor's ideas in practice. Gregor described modern architects' functions as aligning organization and technology, reducing friction, and charting transformation journeys. Such modern architects ride the Architect Elevator from the penthouse, where the business strategy is set, to the engine room, where engineers implement enabling technologies.

In my quest to define modern architectural roles, I used Staff+ Engineering jobs as an inspiration for the development of architects. Tanya Reilly's book The Staff Engineer's Path[2] and Will Larson's book Staff Engineer: Leadership beyond the management track[3] are a helpful guide in defining the responsibilities of architects. The Staff-plus engineering roles provide excellent examples for the development of modern architects.

[1] https://architectelevator.com/
[2] https://www.oreilly.com/library/view/the-staff-engineers/9781098118723/
[3] https://staffeng.com/guides/staff-archetypes/

Many other sources have influenced my work. Some of them are in the Bookshelf section.

## A Bit of Personal History

The work presented in this book builds on several years of my experience. Most of this work results from my current work as a Chief Architect at AVIV Group and previous works as a Principal Architect for eBay Classifieds and Adevinta.

Another vital part of my experience that shaped this book was my earlier experience as a consultant and analyst at the Software Improvement Group. I've learned the value and pragmatics of data-informed decision-making. As a spin-off of this work, I've also built a tool called Sokrates[4], which enable efficient and pragmatic extraction of data about technology and organization from source code. This work has directly influenced my view on the architecture data pillar.

My experience as a CTO of Incision[5], a startup, has helped me better understand the difficulties and pragmatics of creating and running an IT organization.

Lastly, my experience as a researcher at Dutch Center for Computer Science and Mathematics (CWI)[6] and Eindhoven Technical University (TU/e)[7] provided me with a valuable background to do rigorous data analyses and research. From this research period, I want to highlight a collection of essays Design Instability[8] I co-authored with Erik Stolterman, where we connected experiences of designing/architecting in three disciplines: classical design, UX design, and software engineering. This work has helped me better relate to and learn from non-technical fields necessary for architecture work.

[4] sokrates.dev
[5] https://incision.care
[6] https://www.cwi.nl/en/
[7] https://www.tue.nl/en/
[8] https://design-instability.com/

# Why This Book?

This book generalizes my experiences in a written form. I have written these articles for two reasons. Firstly, the act of writing helps me to clarify and improve my ideas (Figure 2). As Gregor Hohpe once noted, I free up some brain cells to learn new things with every sentence I write. As nicely described by Hohpe written word has distinct advantages over the spoken word:

- it scales: you can address a broad audience without gathering them all in one (virtual) room at the same time
- it's fast to process: people read 2-3 times faster than they can listen
- it can be easily searched or versioned.

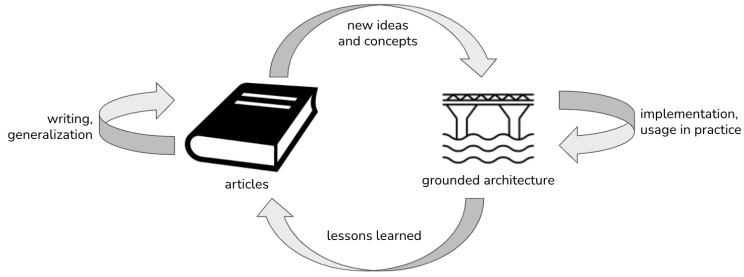



***Figure 2**: Writing a book helped me organize ideas, obtain new insights, improve principles and tools, and share the lessons learned.*

Second, by generalizing and putting my experiences on paper, I hope to create more usable materials to help others in similar situations. I also expect helpful feedback from a broader community.

# A Part of the Bigger Picture: A Trilogy in Four Parts

This book is a part of the collection of open-source tools and resources[9] I have built in the past ten years to help me in architectural work (Figure 3).

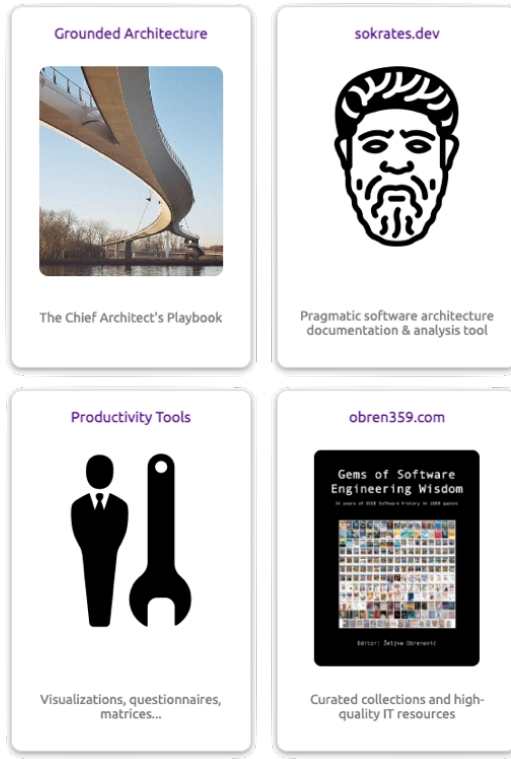



***Figure 3:*** *Grounded Architecture is a part of the collection of open-source tools and resources I have built in the past ten years to help me in architectural work.*

The other resources include:

[9] https://obren.io/

- **Sokrates**[10]: an open-source polyglot code analysis tool that implements my vision of documenting and analyzing software architectures of complex systems. Sokrates provides a pragmatic, inexpensive way to extract rich data from source code repositories. Sokrates can help you understand your code by making the size, complexity, and coupling of software and people interactions and team topologies visible.
- **Productivity Tools**[11]: for years I have been building simple personal productivity tools.
- **359° Overview of Tech Trends**[12] is my collection of knowledge resources with podcasts, and videos from over 20 authoritative, high-quality sources (IEEE, ACM, GOTO Conf, SE Radio, Martin Fowler's site, Ph.D. Theses). Architects need to learn fast, and finding good knowledge sources is difficult.

You can find more details about these tools at my homepage **obren.io**[13].

[10] https://sokrates.dev
[11] https://obren.io/tools
[12] https://www.obren359.com/
[13] https://obren.io/

## Who Should Read This Book?

When writing this book, I had a broad audience in my mind. The article should be helpful to both technical and non-technical people. The book can help IT architects to better understand their value and place in a broader organization. I also hope the articles show the less-technical audience the benefits of staying close to and well-connected with the technologists.

## The Structure of the Book

I have organized the book into several main parts. In the introductory part, I describe the context in which my ideas have developed.

In the second part, I discuss the Grounded Architecture structure describing its three elements: the data pillar, the people pillar, and the Architecture Activities Platform.

In the third part, I talk about the Reflections of Grounded Architecture, grouped int to sections: Being Architect and Doing Architecture.

I conclude with a summary and pointers to external resources for those who want to explore more.

The bookshelf section shows many other books and resources that have influenced my approach and ideas.

## Stay Connected

You can find additional resources and more resources online at:

- https://grounded-architecture.io[14]

Feel free to follow me on LinkedIn to see what I am up to:

- https://www.linkedin.com/in/zeljkoobrenovic

[14] https://grounded-architecture.io/

# Acknowledgments

Thank all eBay Classifieds Virtual Architecture Team (VAT) members and AVIV Group's Architecture Center of Excellence members who gave me invaluable feedback and discussions. Lastly, thank Peter Maas and Brent McLean for sponsoring and pushing for developing data-informed architecture in our organizations.

# Context: Fast Moving Global Organizations

image by paul brennan from pixabay

**IN THIS SECTION, YOU WILL:** Understand the context in which the ideas in this book developed.

**KEY POINTS**:

- To better understand any idea or solution, it is crucial to understand the context in which this idea developed.
- The grounded architecture approach has developed in the context of global, loosely coupled organizations that are diverse, with non-linear growth dynamics, and under transformation pressures.
- More specifically, the work I present here reflects my experiences as a Chief and Principal Architect at AVIV Group, eBay Classifieds, and Adevinta.

To better understand any idea or solution, it is crucial to understand the context in which these ideas developed. My approach to creating and running an architecture function is not an abstract idea. Instead, it is the generalization of many lessons learned while solving specific problems in a particular context.

I base my ideas on my experiences as a Chief and Principal Architect at AVIV Group, eBay Classifieds, and Adevinta. In this section, I discuss critical characteristics of the organizational context that have had an impact on my definition of the Grounded Architecture approach:

- **global scale**: operating across multiple countries and continents with millions of users,
- **multi-dimensional diversity**: the organizations I worked in were very diverse at all levels, including customer base, workforce, business models, team topologies, technology stacks,
- **nonlinear growth dynamics**: in addition to organic growth, big organizations frequently change the portfolio both by mergers and acquisitions of new businesses and as well as by divestments,

- **synergies and transformation pressures**: big organizations do not want just to be big, they want to exploit the benefits of the economies of scale and reduce duplication of efforts,
- **decentralized**, **loosely-coupled organizational units**: organizational units have significant autonomy.

In the following sub-sections, I discuss these characteristics in more detail from an architectural viewpoint.

# Characteristic 1: Global Scale

I was lucky to develop my approach in genuinely global and multicultural organizations. Global context leads to a massive scale:

- Many geographies, cultures, and languages,
- Millions of daily user visits,
- Thousands of active software developers,
- Hundreds of product and development teams,
- Hundreds of millions of lines of source code.

image by pete linforth from pixabay

The global scale introduces several compelling opportunities. The global scale can increase organizational effectiveness due to reducing duplication of effort by centralizing shared activities. Second, the global scale enables leveraging economies of scale by achieving cost advantages, such as lowering unit prices of used technologies. Furthermore, the global scale can increase business resilience and flexibility, possibly compensating for negative local market changes with global resources. Global organizations also have a bigger talent pool to support local or international efforts. Lastly,

such organizations have significant resources to invest in to support nonlinear growth through mergers and acquisitions (M&As).

The global and massive scale has many challenges. It leads to high organizational complexity, with thousands of possible communication channels among the organization's units. Global scale means having a complex technology landscape with many services and interconnections. An immense talent pool also means continuously high costs for the workforce. And such organizations also have constantly high costs of computing resources due to the need to serve many customers. Operating in many places and with high customer demands also increases the complexity of running operations with increased customer demands. With many data centers, services, and applications, global organizations have a huge attack surface, with many points on the boundary of their systems where an attacker can try to enter, cause an effect on, or extract data. Lastly, any manual process, such as diagram drawing to create an overview of the organizational or technology landscape, is limited due to scale.

Balancing opportunities and challenges on a global scale is one of the most difficult and rewarding aspects of my architectural work.

# Characteristic 2: Multi-Dimensional Diversity

The organizations I worked on were very diverse across several dimensions:

- Cultures: both in terms of (local and remote) workforce and customers,
- Organization: different sizes, complexity, and styles of organizational units,
- Product: with diverse sets of features covering many markets and different customer segments,
- IT Architecture: with legacy and modern approaches mixed, and
- Technology: with dozens of programming languages and thousands of thirds party libraries, frameworks, and services.

image by simon from pixabay

Organization-wise, we had many units, such as local marketplaces (each with its sub-structures), global capabilities, enterprise IT teams, data capabilities, and generic IT infrastructure. These organizational units differed in multiple ways:

- Size: for instance, some units had hundreds of developers, and some units had only a dozen.
- Team topologies: some organizations had one team, while other teams were organized hierarchically, and
- Position of architecture: with some organizations having local architecture teams and local lead architects, smaller units have their team members conducting architecture activities in addition to other responsibilities.

IT architecture-wise, we had many architectural styles in active production systems, ranging from legacy monolith applications to complex modern microservice and serverless ecosystems. Each part of the organization has a different history and a different legacy.

And our technology covered almost any mainstream stacks. The technical infrastructure included several public cloud providers (AWS, GCP, Azure) and custom-built private data centers. The systems also employed diverse application technologies, such as:

- Database technologies (e.g., MySQL, Postgress, MongoDB, Cassandra, AWS RDS…).
- Backend programming languages (e.g., Java, C#, Go, Scala, PHP, Node.js, Kotlin…).
- Mobile app programming languages (e.g., Swift, Objective-C, Java, Kotlin, Flutter/Dart…).
- Frontend programming languages and frameworks (e.g., React, Android, AndroidJS, Vue, jQuery…).

From an architectural perspective, diversity offers several opportunities. It can increase technology innovation due to a diverse workforce and the possibility of creatively exploring more technologies and tools. And it can help address diverse implementation needs better by choosing from a more extensive and diverse pool of resources (selecting the best tool for the job).

Diversity also poses several challenges. High diversity increases overall system landscape complexity and the cognitive load of teams who must master many different topics simultaneously. Diversity can also reduce flexibility and reorganization possibilities as expertise is split among many domains and technologies. And the variety of technology stacks also may lead to higher technical debt due to many legacy components in many (outdated) technologies.

From an architectural perspective, diversity is an excellent source of new possibilities but comes with challenges in controlling complexity.

# Characteristic 3: Nonlinear Growth Dynamics

Complex organizations like the one I have worked in are frequently very dynamic. Such organizations grow (or shrink) and reorganize often and significantly. They change both organically and inorganically. Organic growth is internal growth the company sees from its operations. Inorganic change comes from buying other businesses, opening new locations, or divesting.

image by pexels from pixabay

Nonlinear growth in particular may be helpful in several scenarios. It can quickly increase the customers base or whole market segments. and such growth can also speed up innovation due to the acquisition of new technologies or services.

But nonlinear growth dynamics have significant influences on architectural activities. The sudden addition of new companies increased organizational complexity with many new units. Obtaining a new company also adds new technology and engineering units with processes and technology stacks. And nonlinear dynamics also require complex architecture to stay flexible if the organization

decides to divest a part of the organization. Due to its complexity, nonlinear growth dynamics open up many challenges as they can lead to many surprises. And as Clemens Rettich stated, in business there are no good surprises[1].

[1] https://www.linkedin.com/pulse/good-surprises-business-clemens-rettich/

## Characteristic 4: Synergy and Transformation Pressures

Complex organizations do not just grow. Instead, they want to be more efficient and leverage economies of scale, cost synergies, or increase capacity for innovation. Our investors expect us to transform to be more than a sum of our original parts.

image by mustangjoe from pixabay

Pressure for synergies and transformations can provide several opportunities. More synergies lead to cost reductions and less duplication. Such cost reductions can accelerate innovation due to more resources freed after synergies. By creating synergic components, we get more possibilities for reuse and sharing. And well-executed transformations can create more efficiencies and lower unit costs.

Pressure to be more synergic and efficient has its challenges. Upfront investment is needed to gain any benefit. Such investment typically brings high risks. The productivity of some teams may (temporarily) drop due to the need to balance efforts between transformation activities and other work. And after transformations, the complexity of the organization and technology landscape may increase due to more dependencies, e.g., reusing central services.

Synergies and transformation pressures can lead to high expectations and pressure that complicate regular architecture work. On the positive side, such forces can create many new opportunities.

## Characteristic 5: Decentralization and Loose Coupling

Researcher Karl Weick developed the concepts of tight and loose coupling to describe the organizational structure first in educational institutions and later applied to diverse businesses. According to Weick, a tightly coupled organization has mutually understood rules enforced by *inspection and feedback* systems. In tightly coupled organizations, management can more directly coordinate different departments' activities according to a central strategy.

In a loosely coupled organization, some of the elements of a tightly coupled organization are absent. Employees have more autonomy, and different departments may operate with little coordination.

image by shire777 from pixabay

Due to historical and strategic reasons, most organizational units I worked with were loosely coupled. Our companies frequently grow through acquisitions of companies in different marketplaces. The strategies also continually promoted a more independent evolution

of each marketplace to better and faster address local market needs. Marketplaces often have a high level of autonomy, frequently with their development teams and sometimes with local, CFOs, CMOs, or CEOs.

Loose coupling offers several advantages:

- Higher flexibility: units can keep developing independently, addressing specific needs without synchronizing with other marketplaces.
- High development speed / faster time-to-market: fewer dependencies make it much easier for marketplaces to change and evolve their products for local needs.
- Innovation: possibilities to quickly explore ideas in smaller contexts.

Loose coupling also has several challenges:

- Duplication of effort: while local market needs differ, there is frequently a significant overlap in product features and technology. This overlap leads to duplication of effort as each marketplace creates solutions for the same problems.
- Increased accidental diversity: limited synchronization offers flexibility but may lead to significantly different design choices, making it challenging to consolidate solutions, move people between teams, or benefit from the economy of scale.
- Limited possibilities for central control: due to fewer dependencies and different goals, it is more difficult to introduce changes across the board.

Loose coupling is architecture-wise an interesting challenge as it frequently leads to a conflict between global alignment and control and local autonomy.

## Questions to Consider

To better understand any idea or solution, it is crucial to understand the context in which these ideas developed. The same applies to applying concepts: contextualizing them in your organizations. When using ideas from this book, ask yourself how your organizational context differs from mine:

- *What are the unique characteristics of your organizational context?*
- *What is the scale of your organization? How it affects architecture function?*
- *How diverse is your organization?*
- *What are grow dynamics of your organization?*
- *Are you experiencing synergy and transformation pressures?*
- *How (de)centralized is your organization?*

# Goals

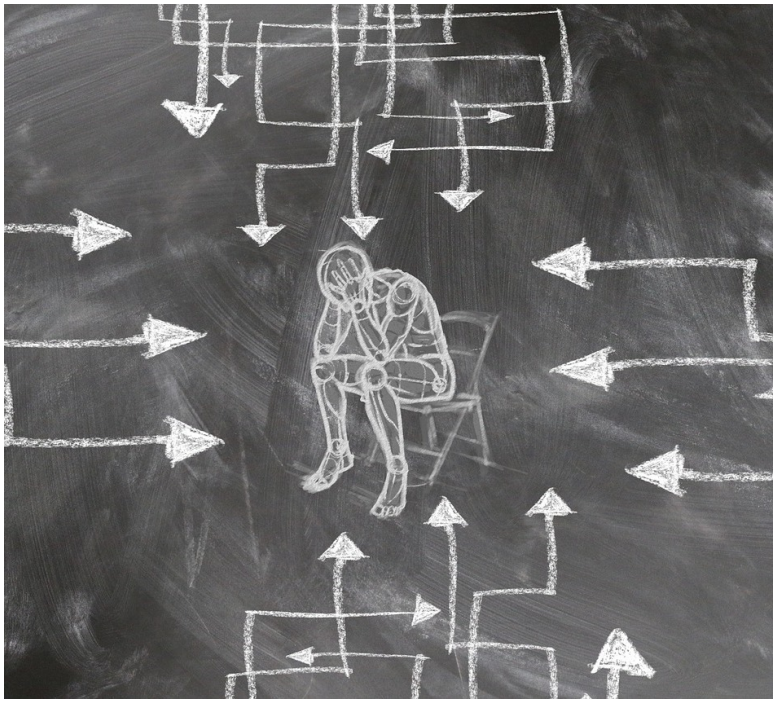


image by gerd altmann from pixabay

**IN THIS SECTION, YOU WILL**: Understand the requirements I identified for an architecture function in complex organizations.

**KEY POINTS:**

- I identified the following needs that an architecture function should support: Executing At Scale, Increasing the Quality of Decision-Making with Data, Maximizing Organizational Alignment & Learning, Higher Adaptivity.

Considering the scale and complexity of the organizational context I operated in, it was apparent that conventional approaches of doing architecture that rely on manual processes need to be revised.

More specifically, I identified the following needs that an architecture function should support.

## Goal 1: Executing At Scale

We needed to find a way to support hundreds of teams, and thousands of projects, with significant complexity and diversity.

## Goal 2: Increasing Quality of Decisions with Data

Intuition does not work at scale. We need tools and mechanisms to make a decision process more data-informed and less dependent on opinions.

## Goal 3: Maximizing Organizational Alignment

In a global, diverse, fast-moving organization, misalignment is a natural state. The architecture function should be a cohesive factor

in minimizing such misalignments.

## Goal 4: Maximizing Organizational Learning

In complex organizations with lots of effort needed to maintain legacy systems, learning and following new technology developments takes work. Architecture should help organizations to learn fast, stay up-to-date with emerging technologies and industry trends and recommend technology upgrades.

## Goal 5: Adaptivity

Significant organic and inorganic changes are frequent and expected. The architecture function must adapt quickly to stay relevant in a new context.

## Questions to Consider

Knowing what goals architecture practice needs to support in your organization is crucial to define structures and measure your impact. Some of the plans may be universally applicable. Others may be unique to your context. Ask yourself the following questions:

- *What is the scale of your architecture operations? Does your scale require special measures to ensure your architecture practice efficient operations?*
- *What are the key decisions you need to make? Do you have the data to base your decisions?*
- *How aligned are units in your organizations? How much friction is there? How can architecture function help?*
- *How much is your organization learning? How is the learning supported?*
- *How stable is your organization? How likely is it that significant changes will occur in your organization?*

# Grounded Architecture

# Grounded Architecture: Introduction

image by ichigo121212 from pixabay

**IN THIS SECTION, YOU WILL:** Get an overview of the Grounded Architecture structure: Data Pillar, People Pillar, and Architecture Activities Platform.

**KEY POINTS:**

- I introduce three elements of Grounded Architecture: The Data Pillar, The People Pillar, The Architecture Activities Platform.

In this part of my playbook, I will introduce the structure of Grounded Architecture. I chose the term "Grounded Architecture" to highlight that the primary goal of my approach is to avoid having an "ivory tower" architecture function disconnected from the organization, which in a fast-moving, global, and diverse setting is a real danger. In other words, I wanted to create an architectural function that is well grounded in the organization.

Grounded Architecture has three elements:

- The Data Pillar,
- The People Pillar,
- The Architecture Activities Platform.

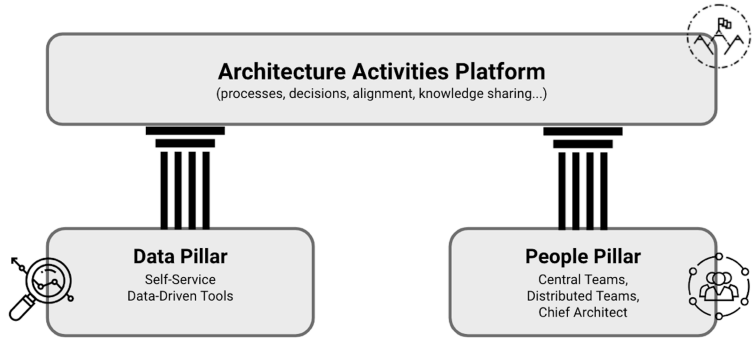



***Figure 1**: The structure of Grounded Architecture.*

The *Data Pillar* is a collection of tools and resources that enables architects to make data-informed decisions based on a real-time

and complete overview of the organization's technology landscape. The Data Pillar section provides more details.

The *People Pillar* is a network of people doing architecture across the organization. This Pillar is crucial to ensure that architecture function has any tangible impact. As noted by Gregor Hohpe, to transform an organization, you do not need to solve mathematical equations; you need to move people. The People Pillar section provides more details.

Lastly, the *Architecture Activities Platform* defines a set of processes and agreements enabling architects to do everything that architecture typically does, leveraging data and people pillars to create a data-informed, organization-wide impact. The Architecture Activites Platform section provides more details on Architecture Activities Platform. While the Architecture Activities Platform looks most like a typical architecture function, I consider such a platform only valid with the healthy Data and People pillars. Without data and people connections, an Architecture Activities Platform becomes an ivory tower institution, generating opinion-based decisions disconnected from reality.

# Data Pillar

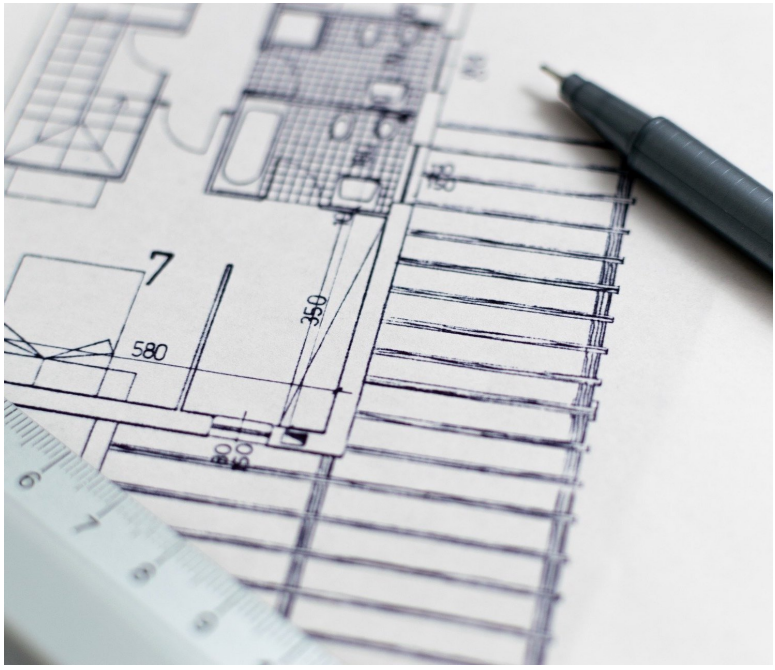



image by lorenzo cafaro from pixabay

**IN THIS SECTION, YOU WILL**: Understand how to use diverse data sources to support architecture decision-making processes and get concrete tips on creating architecture-centric data tools.

**KEY POINTS:**

- The architecture data pillar serves as a medium to create a complete, up-to-date picture of critical elements of technology landscapes of big organizations.
- The data pillar provides an architecture-centric view on data about a technology landscape based on source code analyses, public cloud billing reports, vibrancy reports, or incident tickets.
- To facilitate creation of data pillars, I have been working on creating open-source tools, such as Sokrates[a], that can help obtain valuable architectural insights from data sources, such as source code repositories.

[a] https://sokrates.dev

In every place I worked on creating architectural functions, I strongly emphasized data. In the past several years, I have been working on creating open-source tools, such as Sokrates[1], that can help obtain valuable architectural insights from data sources, such as source code repositories or public cloud billing reports. Consequently, one of the first steps I make in any architecture role is to create an architecture data pillar to get a complete, up-to-date picture of critical elements of the technology landscapes of an organization (Figure 1). Manual documentation does not scale in our context, and relying on data ensures reliability and scalability.

[1] https://sokrates.dev

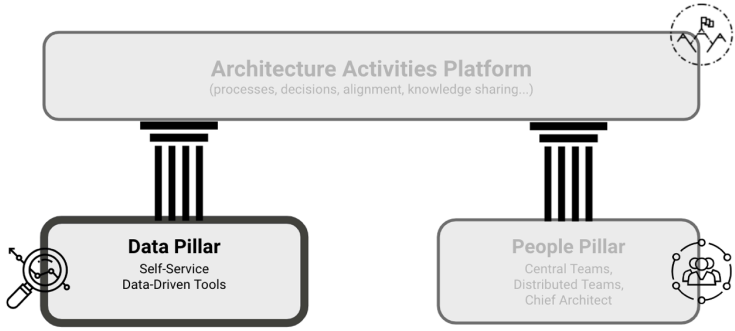



***Figure 1**: The structure of Grounded Architecture: The Data Pillar.*

The good news is that big organizations have lots of data that, if used wisely, can provide an excellent basis for an architectural data pillar. With some automation and curation, getting a good overview of the technology landscape may be closer than it initially appears.

My vision for building the Data Pillar follows the map-making metaphor. Maps are one of the most critical documents in human history. They give us tools to store and exchange knowledge about space and place. While there are differences between maps and the layers they show, the one thing that all maps do is provide readers with **orientation**. A sense of place is central to meaning-making. Maps are also composed of multiple layers. Similarly, the architecture data pillar should give readers a sense of orientation, offering data layers about systems that describe their sizes, connections, quality, security, or human activity.

## Examples of Data Sources and Tools

I've always aimed to get reliable data about technology with as much as possible automation. Some examples of data I used include (Figure 2):

- **Source code**, which contains an incredible amount of information about technology, people's activity, team dependencies, and the quality of software systems.
- **Public cloud billing reports**, which provide an overview and trends used cloud services, regions, and budgets.
- **Incident reports**, which can reveal trends and dependencies among incidents.
- **Key business metrics**, like vibrancy, which can show user activity on our systems.
- **Messaging and collaboration platform (such as Slack) activity reports**, which can help understand discussion topics and team interactions.

In the following sections, I detail several of these architectural data-driven tools.

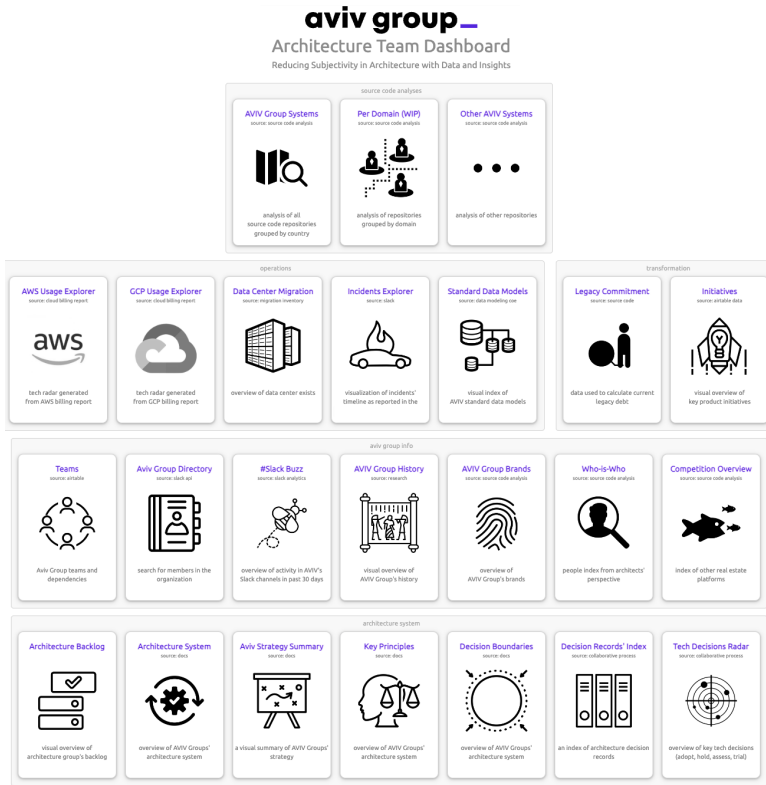



**Figure 2:** *A screenshot of the start page of the architecture data dashboard we've built and used at AVIV Group.*

## Example 1: Source Code and Commit History

I have repeatably found the source code to be an incredible source for creating data-driven architecture documentation. Source code and its commit history include an astonishing amount of information about technology, people activity, team dependencies, and the quality of software systems. I've started and still actively maintain the project Sokrates[2], with the idea to help further extract data from

[2] https://sokrates.dev

source code that can help my work as an architect. I use Sokrates daily, improving it on the way.

I have designed Sokrates from an architect's point of view, enabling quick zooming in and out into source code landscapes. On the one hand, Sokrates provides a high-level view of the landscape, summarizing data from all teams and groups. At the same time, you can zoom in on the details of particular systems to the code level. That means you can use the same tools to have CTO-level discussions looking at overall trends in our technology usage and costs. At the same time, I could engage with developers and discuss concrete code fragments and potential improvements in the code level (e.g., duplicated blocks, complex units, dependencies).

The Appendix at the end of this section shows some insights from source code analyses with Sokrates. For more complex examples of insights that Sokrates generates from source code, take a look at Sokrates examples[3], with analysis of complex open-source landscapes, such as:

- **Apache Software Foundation Repositories**[4], with aggregated multi-level analysis of more than 1,000 repositories with more than 180 million lines of code, more than 22,000 historical contributors, and 2.4 million commits.
- **Facebook/Meta OSS Repositories**[5], with aggregated multi-level analysis of around 800 repositories with 120 million lines of code, more than 20,000 historical contributors, and more than 2 million commits.
- **Microsoft OSS Repositories**[6], with aggregated multi-level analysis of more than 2,400 repositories with more than 100 million lines of code, more than 18,000 historical contributors, and more than 1.2 million commits.

[3] https://www.sokrates.dev/
[4] https://d3axxy9bcycpv7.cloudfront.net/asf/_sokrates_landscape/index.html
[5] https://d3axxy9bcycpv7.cloudfront.net/meta/_sokrates_landscape/index.html
[6] https://d3axxy9bcycpv7.cloudfront.net/microsoft/_sokrates_landscape/index.html

- **Google OSS Repositories**[7], with aggregated multi-level analysis of more than 1,600 repositories with more than 200 million lines of code, more than 27,000 historical contributors, and more than 2.4 million commits.
- **Linux Source Code**[8], with aggregated multi-level analysis of 178 Linux repository sub-folders with more than 23 million lines of code, more than 17,000 historical contributors, and more than 1.7 million commits.
- **Amazon OSS Repositories**[9], with aggregated multi-level analysis of more than 2,700 repositories with more than 130 million lines of code, more than 13,000 historical contributors, and more than 600,000 commits.

In addition to standard source code and commit history analyses, I also have built several special source code analyses to get further details:

- Travis and Jenkins files analyzers to understand how teams build CI/CD pipelines.
- Dockerfile scan to create a tech radar of runtime technologies.
- GitHub API pull requests analyses to identify deployment frequency.

And I encourage you to experiment with your source-code analyses.

## Example 2: Public Cloud Usage

Migrating to the public cloud can dramatically increase transparency thanks to uniform automation and monitoring. The public

[7] https://d3axxy9bcycpv7.cloudfront.net/google/_sokrates_landscape/index.html
[8] https://d3axxy9bcycpv7.cloudfront.net/asf/_sokrates_landscape/index.html
[9] https://d3axxy9bcycpv7.cloudfront.net/amzn/_sokrates_landscape/index.html

cloud transparency offers an incredible amount of valuable data out-of-box.

Figure 3 shows the anonymous screenshot of the Cloud usage explorer, a tool I built to visualize automatically-collected data from standard Google Cloud Platform (GCP) usage reports.

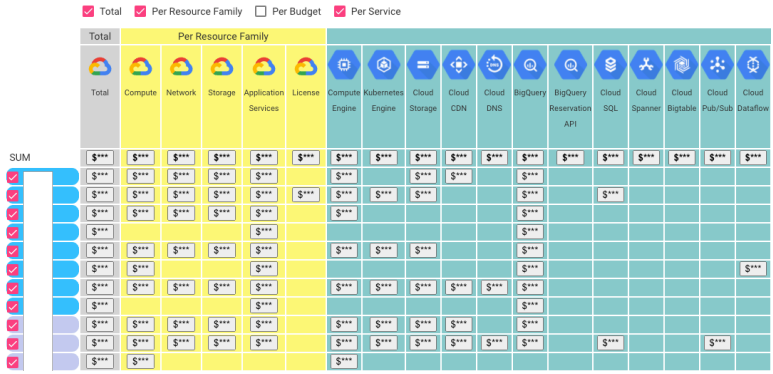



***Figure 3**: An example of a cloud usage explorer.*

Amazon Web Services (AWS)[10], Google Cloud Platform (GCP)[11], Microsoft Azure[12], and other Public Cloud Providers give detailed data about which platform uses which services, resource family, and budget. You can also understand which people and teams have access to each service. It is possible to get real-time information about cloud usage and understand the trends fully automatically.

## Example 3: Financial and Vibrancy Data

Finance departments are very data-driven and have high-quality data that could be relevant for architects. In addition to standard costs, budgets, and other pure financial data types, I frequently found that finance teams also have different data sources, such as vibrancy or usage levels. These teams need such data to, for

[10] https://aws.amazon.com
[11] https://cloud.google.com/
[12] https://azure.microsoft.com/

instance, correlate finance performance with usage levels. Such usage data are beneficial for architecture discussions. For example, linking usage levels and vibrancy of systems with their public cloud usage can identify areas of improvement and inefficiencies.

## Using Architecture Data Pillar

The architecture data pillar can provide lots of data. Sometimes, as in an ordinary map or atlas, such data could be helpful for those who want to orient themselves and understand the context. You can obtain more insights from such data. However, finding the right ways to interpret and use data requires active effort. In other words, the data can give you the answers, but we may not know the questions[13]. Here are some of the questions you can ask when you have data:

- Are we aligned and going in the same direction? Source code overviews, public cloud usage explorers, or tech radars can highlight differences among systems and teams and trigger discussions.
- Are we using technology optimally? Comparing usage trends between teams can show interesting outliners (both positive and negative).
- Are there indicators of poor code quality? Too large systems, duplication, long units, or long files.
- Productivity: is more really more or is more actually less. For instance, comparing the number of git merges with the number of developers can indicate if our dev processes are scalable. When we scale up teams, we want to speed up our delivery (but if team structure is not proper, it can easily be the opposite as people "step on each other toes").
- Do we collaborate in the way we want? Repository analysis can point out team topologies and (un)desired dependencies.
- Do we work on the things we want? We may want to focus more on innovations, but in reality, we may spend too much time on legacy maintenance.

[13] https://en.wikipedia.org/wiki/42_(number)#The_Hitchhiker's_Guide_to_the_Galaxy

## Building Data Pillar

While each organization will have its unique sets of data, here are some tips I found helpful in my approach to forming the architecture data pillar:

- **Start with the source code**. My motto is "Talk is expensive. Show me the code." I scan as soon as possible all source code using tools such as Sokrates[14]. I highly recommend Sokrates[15] as the basis for the data pillar, but other simple analyses could also provide a good starting point. Modern IT enterprises store almost everything as a code. It is the richest and most up-to-date documentation on most things happening in an IT enterprise.
- **Connect with finance and governance teams** to get exports of their data (without sensitive parts, such as revenue projections). Cloud billing reports and data about vibrancy or revenue streams are collected anyway. By extracting more technology-oriented data (e.g., public cloud technology usage trends) and connecting them to other data, many new insights may be obtained without starting new processes or asking people to provide more details. First, leverage what you have, squeeze all the value from it, then ask people for any missing elements.
- **Use simple and easy-to-maintain infrastructures**. For example, I publish the results of Sokrates analyses and other simple data Web apps as static resources in our enterprise GithHub pages. Configuring more complex infrastructure with complex databases and backend software requires more maintenance.
- **Maintain a culture of transparency**. It is much simpler and more effective to share fewer data with everyone than to have more data, but complex authorization is needed.

[14] https://sokrates.dev
[15] https://sokrates.dev

- **Own the curation**. People need to be able to trust your data. Spend enough time to understand data sets, curate them, and ensure presentation consistency. I consider myself a master curator and chief UX designer of a data pillar.

While I do not want to prescribe the best technology, I can tell what I use in daily work. I build most architecture data tools as simple web applications, taking data from JSON files hosted on a static web server. See some of my public tools[16] to illustrate how I build such simple data-driven web apps.

[16] https://obren.io/tools

# Appendix: Examples of Insights From Source Code Analyses with Sokrates

Figures 3 to 7 show some insights from source code analyses with Sokrates.

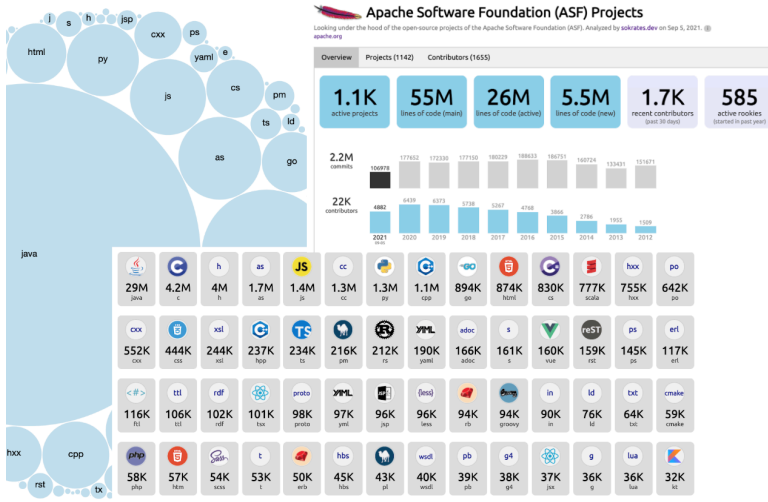



**Figure 4**: *Sokrates can instantly create a helicopter view of the technology landscape, programming languages, active contributors, and commit trends.*

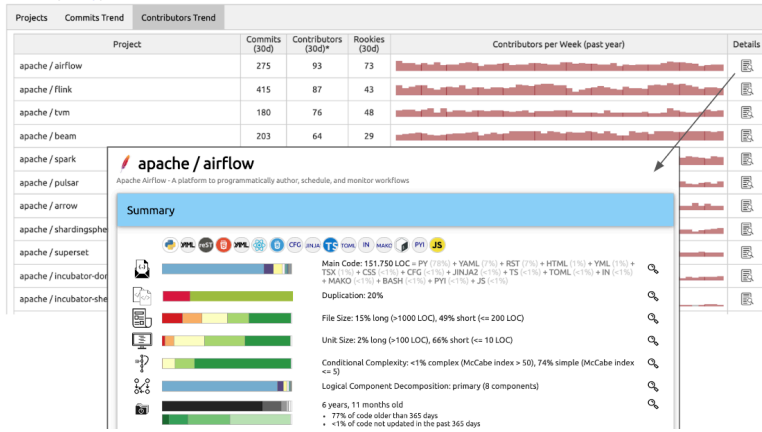



**Figure 5**: *Sokrates can show detailed code and contributors' trends per repository, enabling zooming in each repository up to the code level.*

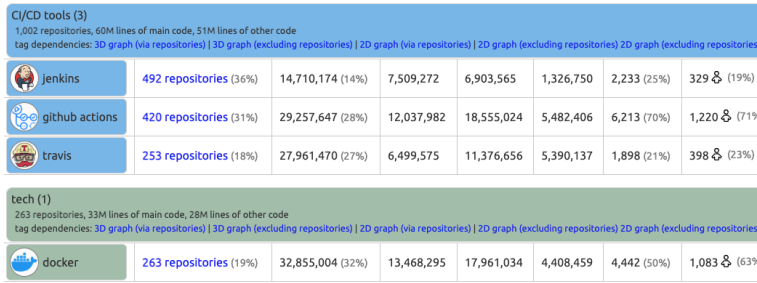



**Figure 6**: *Sokrates can create a tech radar by tagging projects with identified technologies.*

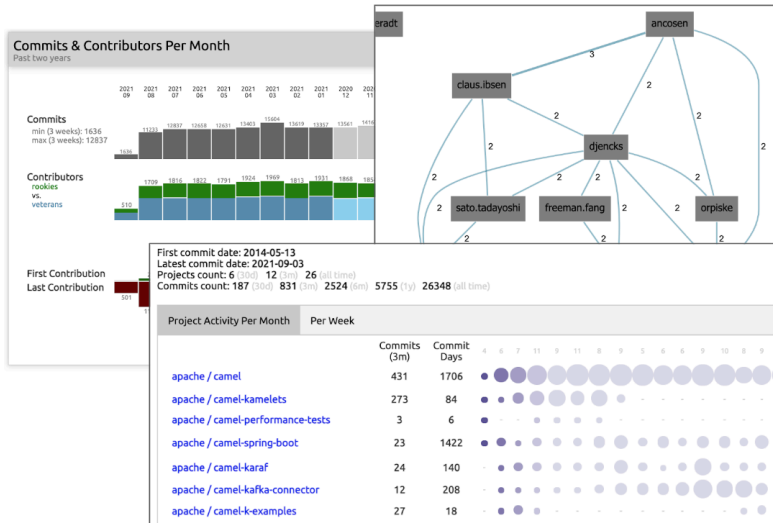



**Figure 7**: *Sokrates can show contributor trends, distribution of "veterans" and "rookies," and dependencies between people and repositories, enabling zooming in into patterns of the contribution of individual contributors.*

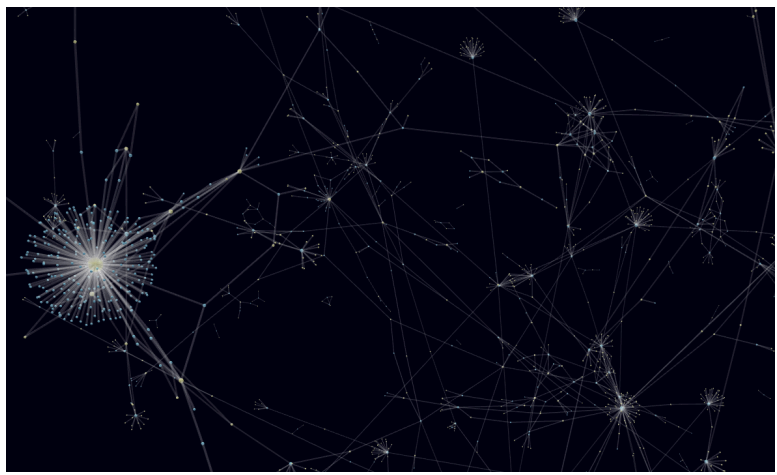


**Figure 8**: *Sokrates can reveal the team topologies by plotting 2D and 3D graphs of dependencies that people create through working on the same repositories in the same period.*

## Questions to Consider

Using data can significantly improve the efficiency and impact of architectural practice. But there are no simple tools that can instantly provide you insights. Ask yourself the following questions:

- *Have you considered using open-source tools like Sokrates to gain architectural insights from data sources? Why or why not?*
- *What are your views on the reliability and scalability of manual documentation as opposed to data reliance?*
- *What steps would you take to create an architecture data pillar in your organization?*
- *Do you think there are untapped data sources within your organization that could inform your architectural data pillar?*
- *How could you automate the process of gathering data for architectural insights in your organization?*
- *What examples can you provide of the data you've used to gain reliable information about technology in your organization?*
- *How would you examine public cloud billing reports, incident reports, or key business metrics for architectural insights?*
- *How can you ensure your data is reliable and up-to-date?*
- *Do you collaborate with finance and governance teams to incorporate financial and vibrancy data into your data analysis?*
- *Is there a culture of transparency in your organization?*

# People Pillar

image by mostafa meraji from pixabay

**IN THIS SECTION, YOU WILL:** Understand that architecture practice is all about people and get tips on creating organizational structures that support practical architecture functions.

**KEY POINTS:**

- Developing the architecture function requires having competent, empowered, and motivated architects.
- Any architecture function must carefully organize, empower, and leverage scarce architecture talent.
- We should not take architectural talent for granted. Architects are difficult to hire talent as they need not only in-depth technical knowledge but also domain-specific and organizational knowledge.
- In my work in the past few years, I combined two teams of architects: a small central architecture team and a cross-organizational distributed virtual team.

The **People Pillar** is an essential element of Grounded Architecture. As noted by Gregor Hohpe, to transform an organization, you do not need to solve mathematical equations. You need to move people. Consequently, having a strong network of people doing architecture across the organization is crucial to ensure that the architecture function has any tangible impact. In other words, Strong Architecture = Strong Architects.

Developing the architecture function requires having competent, empowered, and motivated architects. We should not take architectural talent for granted. Architects are bridging local business, product, organizational, and technology issues. Architects are difficult to hire talent as they need not only in-depth technical knowledge but also domain-specific and organizational knowledge. Consequently, any architecture function must carefully organize, empower, and leverage scarce architecture talent (Figure 1).

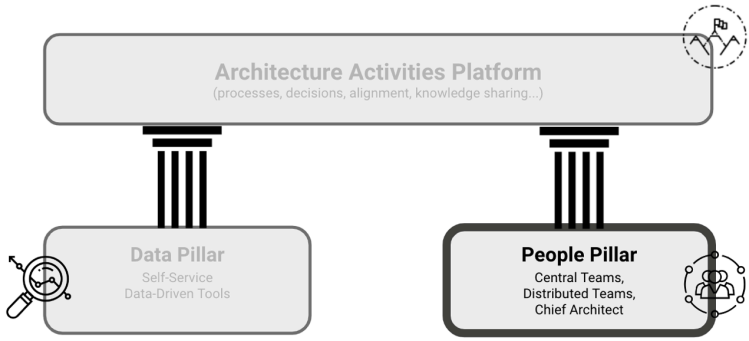



***Figure 1**: The structure of Grounded Architecture: The People Pillar.*

In my work in the past few years, I was working by combining, in different forms, two teams of architects: a small central architecture team and a cross-organizational distributed virtual team. A central architecture team is an enabler for the rest of the organization, supporting teams and addressing global strategic topics. A distributed virtual architecture team consists of architects (or other people making architecture decisions in their teams) working in local organizational units but spending some time in a virtual team with peers from other teams. Such a distributed virtual architecture team is a crucial element of an architecture function. It provides the connection (grounding) across all parts and levels of the organization, increasing transparency, building people networks, and making it easier to implement change.

# Background: Central and/or Federated Architecture Function

The architecture function generally follows one of two fundamental models: central or federated (Figure 2). (McKinsey 2022[1]).

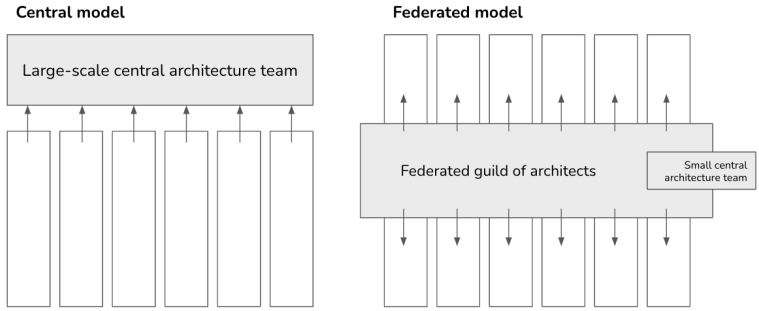



***Figure 2:*** *Central and/or Federated Architecture Function.*

The central model involves a large-scale central architecture team. The central team typically defines the process for approval of new work and assures adherence by development teams. In this model, development teams have few or no qualified solution architects that are typically a part of the central architecture team. This model also holds centralized infrastructure, operations, and security teams apart from the development function. Control and governance are typically the primary concerns of the central architecture team.

The federated model generally relies upon a guild or "community of practice" of solution architects embedded into individual development teams. A small central architecture team or an architecture center of excellence (CoE) may complement such a guild. The federated model's architects facilitate high-level planning and act as on-demand service providers for distributed teams.

The federated model is more commonly associated with cross-

[1] https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/tech-forward/crafting-the-optimal-model-for-the-it-architecture-organization

functional DevOps culture. The roles of solution and enterprise architects are generally broader in scope to integrate infrastructure, operations, and security concerns in product-oriented teams. The architect's role focuses on facilitation and enablement rather than control.

Today, modern agile organizations mainly adopt the federated model. This approach increases the likelihood that the central architecture team will spend time closely involved with the challenges identified in the teams. The model ensures that the architects will be evaluated against the goals of the individual products they support, thereby focusing on improving performance and reducing complexity.

# The Hybrid Model

To operate in a complex context, you need to invested effort to ensure you have the right people at the right places. In the end, I usually found it best to adopted model of a hybrid organization combining elements of central and federated orientation structures:

- A Central Architectural Teams, and
- Architecture Guilds & Virtual Architectural Teams.

The hybrid team structure supports well our goals of executing at scale. Guilds and virtual architecture teams support execution by increasing the number of people involved in architectural activities and increasing work efficiency through better alignment. By having members representing various organizational units, we are able to have much more impact across the board. And having some capacity on the central level serves as a catalyst helping people at local levels to do their job while being aligned and better connected with overall strategic goals and other teams working on similar topics.

## Central Architecture Team

The roles of people in central teams may differ depending on the organization. In addition to doing typical architecture work, I found it helpful to be able to have people that can cover the following types of responsibilities:

- **Build and maintain the architecture Data Pillar**. Building a data pillar will not happen by accident. It requires clear ownership, curation, and technical support.
- **Promote data-informed decision-making**. identify, collect, and use relevant data. Only some people are used to applying data in their decision-making. Architects should provide support and be a role-model for data-informed organizations.

- **Proactively identify, connect, and maintain relationships with all relevant stakeholders**. Architects are frequently uniquely positioned to bridge different organizational units and stakeholders.
- **Build internal architecture communities and guilds**. Organizing rituals and people requires active effort.

While guilds and virtual teams could do many of the listed activities, the voluntary nature of guilds and virtual groups makes such support more fragile. The central architecture team can take full long-term ownership of some topics and be a backup if community support weakens, ensuring long-term continuity.

## Architecture Guilds & Virtual Architecture Teams

> *"A lot of cheap seats in the arena are filled with people who never venture onto the floor. They just hurl mean-spirited criticisms and put-downs from a safe distance. … we need to be selective about the feedback we let into our lives. For me, if you're not in the arena getting your ass kicked, I'm not interested in your feedback."* — Brené Brown, Rising Strong

I always found it essential to connect organization members passionate about architecture in some form, a guild, a community of interest, or a virtual team.

Guild or virtual teams are composed of people that work full-time as architects or tech leads in specific organizational units but spend part-time collaborating with architects from other departments to reach more alignment, share knowledge, and leverage each other's work. In this peer-to-peer community, architects are collectively

responsible for identifying and growing architectural talent, mentoring, and helping each other.

When having many guilds and teams, we have organized architects in several sub-areas:

- General teams for a broader set of architectural topics.
- Specialist teams focus on a particular part of the technology stack. Examples include native mobile apps, web frontends, public cloud infrastructure, etc.
- Strategic initiatives teams. For instance, data strategy, public cloud strategy, transactions, or verticalization.

Having places and events to connect central and distributed teams is essential. Such events can transform individual experiences into collective knowledge that can benefit the whole organization. In most organizations I worked in, distributed teams followed a similar pattern of rituals:

- Regular (e.g., bi-weekly) forums, with updates, announcements of architectural spikes, and sharing or architectural decisions (similar to Andrew Harmel-Law's Advice Process[2])
- Annual or bi-annual summit, with several days of intensive knowledge sharing and workshops
- Ad-hoc workshops focusing on some explicit topic

While the central team can provide some essential support, all communities must take the initiative and engage as many people as possible during these events. People should be active participants rather than passive receivers of information to ensure more involvement and commitment.

[2] https://martinfowler.com/articles/scaling-architecture-conversationally.html

## Embracing Diverse Team Structures

When building architecture guilds and virtual architecture teams, it is essential to acknowledge that organizational units have diverse structures and sizes. There is no one-fit-all solution about how departments should assign architecture responsibilities. I generally worked on three types of team-architect systems per Gregor Hohpe's view of architects and their teams' relationships[3]:

1. **Benevolent "dictator"**: An architect or architect team tells developers what to do or how to do things. An important nuance is to what extent the line is unidirectional or bi-directional.
2. **Primus inter pares (first among equals)**: Architects are embedded into teams where each is just another team member focusing on the system structure and trade-offs, perhaps taking a longer-term view than other team members.
3. **Architecture without architects**: Architecture is done within teams. However, the task is a shared responsibility across multiple (or all) team members. This approach can work quite well and is often the preferred model.

In big organization, embracing diversity is a pre-requirements to have any broad impact.

[3] https://architectelevator.com/architecture/organizing-architecture/

## Building People Pillar

While each organization will need its unique approach, here are some tips I found helpful in forming architecture teams and "People Pillar":

- Before making grandiose plans for reorganizations, connect with the people already doing architectural work in an organization, creating a community of practices or a virtual team. Being inclusive and connecting all key tech leaders, regardless of their actual position and title, is vital. Being well-connected to these people will be crucial in any architecture organization, so you will always benefit from this effort.
- If creating a virtual team is a part of your architecture strategy, move away from making an informal community of practice towards building a team with more accountability and responsibility. It was helpful to get buy-in from key stakeholders, e.g., Engineering Leaders who should want to get their people to work together with other architects.
- Connect with non-architecture stakeholders in the early stages. Again, being well-connected to these stakeholders will be crucial in any architecture organization.
- Avoid hiring digital hitman[4]. Invest in growing internal talent. Architects require technology, domain, and organization knowledge. Finding such a person outside the organization is challenging.
- Externalize. Reach out and connect. Participate in external events. Publish. Being strong externally can help you to both grow and attract architectural talent.

[4] https://architectelevator.com/transformation/dont-hire-hitman/

image by chantellev from pixabay

## Questions to Consider

It is difficult to overestimate the importance of people for architecture practice, yet many organizations take architectural talent for granted. To reflect on the importance of carefully organizing, empowering, and leveraging scarce architecture talent, ask yourself the following questions:

- *Reflect on your organization's current architecture function. Do you have a strong network of architects across the organization?*
- *How do you ensure architects' competency, empowerment, and motivation in your organization? What systems do you have in place to develop architectural talent?*
- *Which central, federated, or hybrid model best represents your current architectural function? Why was this model chosen, and how effective has it been for your organization?*
- *If you are a part of a central architecture team, how would you support the rest of the organization? How would you contribute to the global architecture function if you were part of a distributed virtual team?*
- *Consider having the roles of central architecture teams and federated architecture teams in your organization. How would they complement each other?*
- *How effective is the current division of responsibilities among architects in your organization? Are there areas of overlap or gaps in coverage?*
- *What steps has your organization taken to ensure architects are well-connected across all parts and levels? What impact has this had on transparency and the implementation of changes?*
- *Reflect on the diversity of team structures within your organization. How does this diversity impact the roles and responsibilities of architects?*

# Architecture Activities Platform

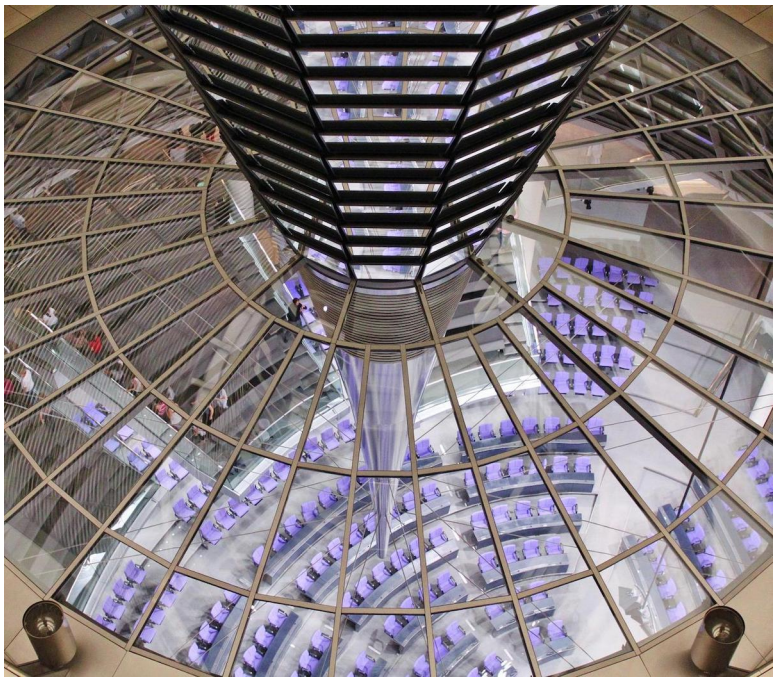


image by anja from pixabay

**IN THIS SECTION, YOU WILL:** Understand what activities you can do as a part of architecture practice and get tips on creating pragmatic operating models.

**KEY POINTS:**

- The Architecture Activities Platform defines a set of processes and agreements enabling architects to do everything architecture typically does, leveraging data and people pillars to create a data-informed, organization-wide impact.
- Examples of activities include: supporting teams in their daily work; tracking tech debt, defining tech debt reduction programs; performing technical due diligence; standardization of processes and documentation; defining cloud, data, and platform strategies.

Each organization will have different architectural needs and contexts. When forming architecture functions, I use as a starting point these two pieces of advice from Gregor Hohpe[1]:

- *"Your architecture team's job is to solve your biggest problems. The best setup is the one that allows it to accomplish that."*
- *"Your organization has to earn its way to an effective architecture function. You can't just plug some architects into the current mess and expect it to solve all your problems."*

[1] https://architectelevator.com/architecture/organizing-architecture/

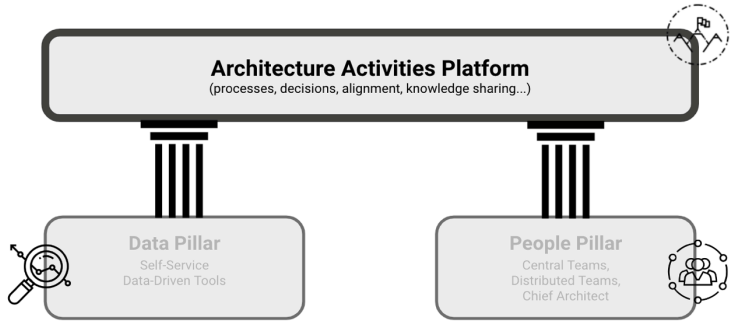



***Figure 1**: The structure of Grounded Architecture: Architecture Activities Platform.*

Considering the previous two points from Gregor Hohpe, there is no one-size-fits-all approach: each organization must find activities and operating models to enable architecture to work on the most critical problems.

The Architecture Activities Platform (Figure 1) is where we have to perform activities that help an organization reach its goals. The Architecture Activities Platform defines a set of processes and agreements enabling architects to do everything architecture typically does, leveraging data and people pillars to create a data-informed, organization-wide impact. In all these activities, data and people posts provide foundations for data-informed decision-making well-embedded in the organization.

## Examples of Architecture Activities

Here are some examples of activities I have been performing with architects:

- **Design mechanisms for teams to make better decisions**. This includes creating global decision-support mechanisms, such as advisory forums[2], formal design authority (for compliance-sensitive projects), and team-specific mechanisms, such as escalation paths in case of decision conflicts (e.g., teams cannot align on a common messaging middleware).
- **Supporting teams in their daily work**. Being part of key team activities, aligning architectural work with team rituals to provide timely support, and supporting the team in all crucial phases of their work (e.g., reviewing architecture proposals early before the project or sprint start).
- **Supporting planned new initiatives and projects**. Ensuring alignment between projects that require multi-team collaboration.
- **Supporting teams in dealing with the legacy landscape**. Providing data and knowledge regarding legacy landscape, identifying hotspots (e.g., frequently changed, low-quality untested pieces of legacy code), defining scenarios and roadmap for legacy modernization.
- **Tracking tech debt**, **defining tech debt reduction programs**. Defining a centrally aligned backlog of technology depth, defining programs for its reduction and integration in planning processes.
- **Performing SWOT and other analyses of platforms and systems**. Doing deep dives to better understand some areas of the technology landscape and create plans and roadmaps for improvement.

[2] https://martinfowler.com/articles/scaling-architecture-conversationally.html

- **Standardization of processes and documentation**. Defining common templates for documents such as Architectural Decision Records (ADRs), Technical Design Reviews (TDRs), or common diagrams.
- **Supporting merger and acquisitions (M&A) activities with expertise and analyses**. Support analyses, recommendations, and integration planning regarding mergers and acquisitions.
- **Defining key technology strategies**. Examples include Cloud, Data, and Platform strategies.
- **Defining vision and direction of technology, frequently collaborating with Engineering Leaders**. Working with managers to create a sustainable organizational setting aligned with technology strategies.

## Operating Model

While exact activities and their scope will depend on an organization setting and will change over time, I aimed to implement the common operational model in daily work.

Inspired by Gregor Hohpe's strategy-principles-decisions model, I typically used the process illustrated in Figure 2.

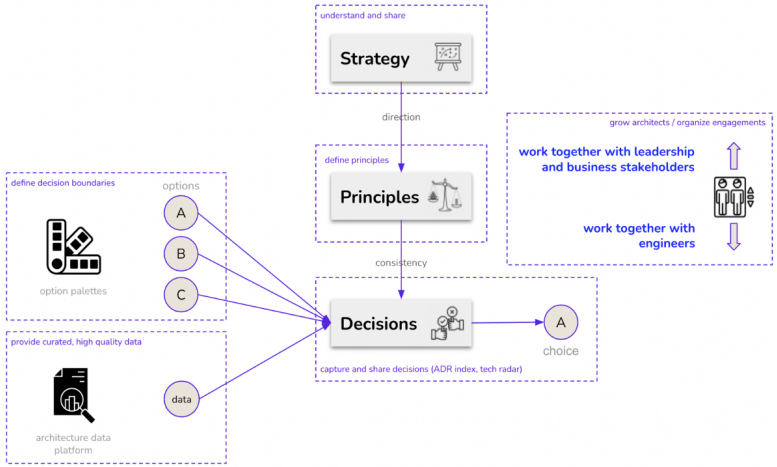



**Figure 2:** *A common operating model I typically use for Grounded Architecture activities.*

A common operating model I typically use for Grounded Architecture activities is as follows:

- Architects engage with stakeholders and product and project teams in a **collaborative and supportive manner**.
- Architects are **empowering the teams** so that they make most of the decisions.
- In all activities, architects do:
  - **Bring relevant data** to inform decisions leveraging the Data Pillar.

  - **Define decision boundaries** to enable the minimal level of compatibility and strategic alignment (e.g., public cloud provider, tech stack constraints).
  - **Define key principles** to facilitate consistency in decision-making.
  - **Share and generalize** lessons learned.

- Architects then spend their time in **constant motion** between supporting teams' **daily work** and working on **strategic topics**, helping the organization achieve alignment between strategy and implementation.

Another characteristic of this operating model is **shifting left** the architecture work. My goal was to avoid formal bureaucratic approval processes, where architects appear too late and are frequently busy approving trivial decisions. Instead, my goal was to have architects involved early in any of the processes, such as during the planning and preparation stages, where it is possible to make more significant changes.

## Distributing Decisions

With proposed operating model, I aimed to keep architectural decision-making distributed across the organization and embedded in the development teams. Development teams traditionally have the best insights and most information relevant for making a decision.

As noted by Gregor Hohpe, the worst case of organizational decision-making happens when people with relevant information are not allowed to make decisions, while people who lack sufficient information make all decisions. Grounded architecture aims to make relevant information more readily available to a broader audience and better connect people when making decisions.

## Autonomy and Alignment

While I aim to create a mechanism to give teams autonomy, autonomy does not mean that teams are alone and do not align with anyone, do not get feedback from anyone, and do whatever they want. Autonomy must be complemented with high transparency and proactivity in alignment with other teams.

I have sometimes implemented the concept of a **decision pyramid** to give maximal autonomy to the teams while maintaining a **minimal** level of global **alignment and compatibility** (Figure 2).

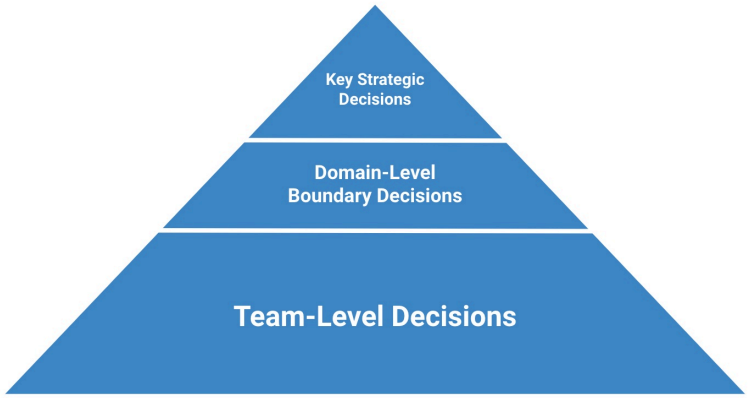



***Figure 2:*** *A decision pyramid. The development teams should make most decisions. However, several strategic and area-level decisions may provide decision boundaries for teams (e.g., a public cloud provider).*

Development teams should make most of the decisions. However, several strategic and area-level choices may provide team decision boundaries. For example, selecting the public cloud provider is typically a CTO-level strategic decision. Similarly, engineering leaders may want to limit some choices, such as the number of programming languages, to more easily train new people, maintain code, and support moves between teams.

## Questions to Consider

Your architecture practice job is to solve the biggest problems in your organization. Ask yourself the following questions:

- *How can you identify the most critical problems that your architects needs to solve in your organization?*
- *What activities and operating models can you think of that will best enable architecture in your organization to work on these critical problems?*
- *In your organization, what does the Architecture Activities Platform look like, and how could it be improved?*
- *Which of the provided examples of architecture activities are you currently performing in your organization?*
- *How does the proposed common operating model align with your current operational practices in your organization? What changes might be necessary to adopt this model?*
- *In your experience, how early are architects involved in projects and activities? Do you agree with the goal of 'shifting left' the architecture work?*
- *How are architectural decisions distributed across your organization currently? How could this process be improved to ensure the people with the most relevant information make the decisions?*
- *Reflect on the balance of autonomy and alignment in your organization. How could you better implement a mechanism to give teams autonomy while maintaining alignment and compatibility with global strategy?*
- *How does the concept of a decision pyramid resonate with you? How is it reflected in your current organization, and how could it be better implemented?*
- *Which strategic and area-level decisions provide team decision boundaries in your organization? Are there areas where you need more or less limitations to optimize performance?*

# Value of Grounded Architecture

image by matthias wewering from pixabay

**IN THIS SECTION, YOU WILL:** Understand the value that architecture practice based on the ideas of Grounded Architecture can create for an organization.

**KEY POINTS:**

- When Grounded Architecture is in place, it can have a significant positive impact on the functioning of an organization.
- These categories of impact are: Executing At Scale, Increasing the Quality of Decision-Making with Data, Maximizing Organizational Alignment & Learning, Higher Adaptivity.

When Grounded Architecture is in place, it can have a significant positive impact on the functioning of an organization:

- Enable Execution of Architecture Function At Scale,
- Increase the Quality of Decision-Making with Data,
- Maximize Organizational Alignment,
- Maximize Organizational Learning, and
- Increase Architecture Function Adaptivity,

The following sections elaborate on these categories in line with the goals.

## Executing at Scale

Grounded Architecture aims to enable architecture functions to operate at scale efficiently. We must rely on something other than intuition and manual processes when supporting thousands of developers and dealing with hundreds of millions of lines of code.

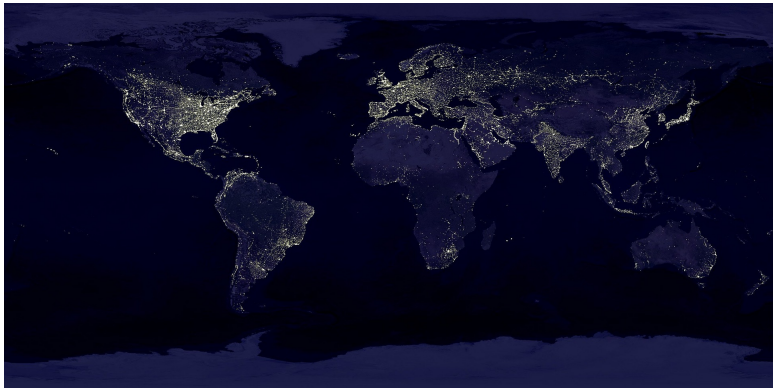


image by wikiimages from pixabay

The architecture data pillar aims to support working at scale with automation and self-service. Our People Pillar aims to help execution at scale by developing connections at all levels of the organization, increasing transparency, and through a strong network speeding up alignment and the execution of shared decisions.

# Increasing the Quality of Decision-Making with Data

> *"If we have data, let's look at data. If all we have are opinions, let's go with mine."* -— Jim Barksdale

Architectural discussions can be very heated and opinionated. There are significant benefits to making our decision process as much as possible data-driven. It is one of the critical tasks for any architect to maintain high-quality data on relevant internal and external technology developments, providing fuel for data-informed discussions and decision-making.

image by arek socha from pixabay

The Data Pillar enables architects to move away from opinion-based decisions to data-driven economic risk modeling to become more data-driven. Such frameworks can help architects to achieve the following (credit Gregor Hohpe):

- dismantle the buzzwords, present the problem in clear terms,

understandable to a broader audience
- identify the real drivers behind buzzwords based on internal and external research
- bring data into the discussion
- translate drivers into an economic risk model, and use the model and data to find the best spot for the given business context

## Maximizing Organizational Alignment

Misalignment frequently happens in big organizations. The Data Pillar can increase organizational alignment by creating transparency. The People Pillar develops global structures that can help people before and after they make an architectural decision:

- Before people decide, people starting to work simultaneously on the same topics can decide to work together, minimize effort duplication, and in that way, save time and resources.
- After making a decision, Grounded Architecture can make all organizations aware of it and distribute it so everyone can profit from lessons learned in one unit.

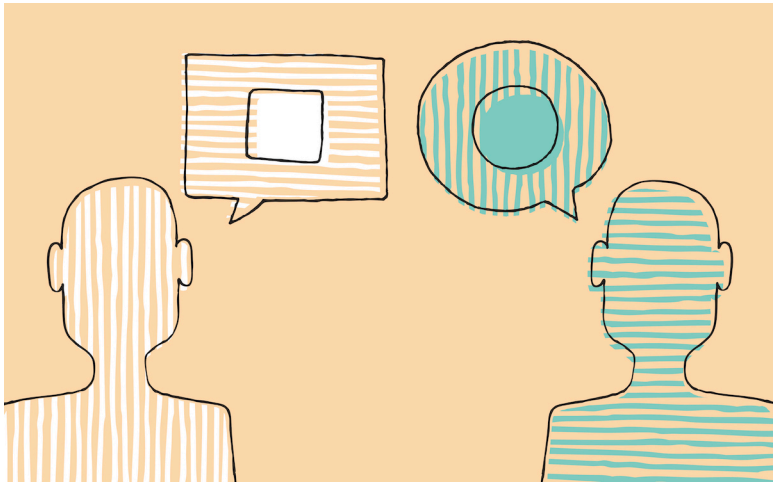


image by istock

## Maximizing Organizational Learning

> *"Good judgment comes from experience, and experience comes from bad judgment."*–Fred Brooks *"I expect you to learn to be better each day. I challenge you to look at each working day as an opportunity to learn more, and by doing so, to grow as a person."*–L. David Marquet

When we have teams of well-connected architects and other stakeholders, such groups can transform individual experiences into collective knowledge to benefit the whole organization.

One of architects' primary daily tasks is learning. We must discover new things about our domain, teams, tools, and technologies. As individual architects, we need to use each day to learn something new. We must maximize personal learning, transforming individual lessons learned into shared guidelines.

One of the problems I frequently see in organizations, particularly complex and international ones, is that they may need more natural spaces for group knowledge sharing. Consequently, the People Pillar aims at creating spaces for sharing knowledge about architecture and technology. These spaces include but are not limited to regular update calls, knowledge-sharing sessions, or conferences.

In addition to creating spaces, as a community, we can further increase our learning value by deriving generalized insights from cross-group cases.

Lastly, one of the problems many organizations face is that due to their complexity and size, they have more challenges in introducing new technologies than their disruptors (Figure 3).

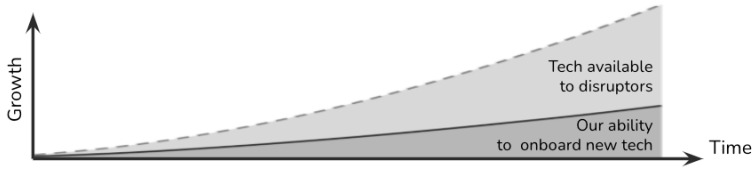



***Figure 3:*** *One crucial aspect of Architects' work is following external trends and finding pragmatic ways to introduce these trends in the organization. Inspired by thoughtworks.com/insights/blog/whats-hurry-building-digital-enterprise*[1]

We need more time to understand and utilize new technology developments while the number of new technologies is increasing due to, e.g., the continuous and accelerating evolution of cloud and mobile products. As architects, we must proactively identify relevant new technology developments. Based on our understanding of these developments, we must create pragmatic technology recommendations for concrete platforms across the organization.

The grounded architecture aims to accelerate the adoption of new technology and quick learning by providing more data to facilitate reflection and better connecting people to leverage each other's explorations.

[1] https://www.thoughtworks.com/insights/blog/whats-hurry-building-digital-enterprise

## Adaptivity

The three elements of the grounded Architecture model, data, people, and strategy/governance platforms, provide a highly flexible and adaptive setting.

This adaptivity is driven by the independence of elements and the possibility of using the elements in different combinations. Here are some of the critical drivers of flexibility:

- The data pillar, if implemented with a high level of automation, allows for quick extensions and reconfigurations to provide data for any change of direction. Extending the platform with new data should be easy after acquisitions or mergers. And the ability to group data at multiple levels supported the what-if scenario analysis (e.g., the impact of different reorganization scenarios). It defined a baseline to track changes in the organization.
- A strong data pillar provides crucial connections and feedback on the daily reality of each part of the organization.
- Grounding the architecture with data and people connections also makes the work of a Chief Architect much more flexible. As most architectural decisions can be delegated to well-aligned teams, a Chief Architect, typically the most experienced technologist, can spend more time on crucial strategic initiatives, such as defining cloud, data, or platform strategies or supporting decisions on mergers and acquisitions.
- Lastly, the structure of our architectural model enables a more sustainable architectural function. With an established Data Pillar and a well-connected architect, the architecture can still benefit the organization even without a strong central team.

image by francis ray from pixabay

## Questions to Consider

It is always essential to be thoughtful about the value and impact of your work. Ask yourself the following questions:

- *How effective is your organization's current architectural function at scale? How valuable are principles of Grounded Architecture to enhance its efficiency?*
- *To what extent does your organization use data to inform architectural decisions? What steps could you take to move your organization from opinion-based to more data-driven decision-making?*
- *How well-aligned are the different areas within your organization, and how does this affect your architectural function? Could the Data and People Pillars principles be utilized to improve alignment?*
- *What strategies does your organization currently have to foster organizational learning? How could the methods described in the Grounded Architecture model enhance this?*
- *How quickly can your organization adopt and utilize new technologies? How could your architecture practice accelerate this process?*
- *Consider the adaptivity of your organization's architectural function. How could your architecture practice improve it?*
- *Reflecting on the value of the "data pillar" concept, how effectively is your organization tracking changes, supporting what-if scenario analysis, and defining baselines?*
- *What role does the Chief Architect play in your organization? Could their time be better utilized on strategic initiatives?*
- *How sustainable is the architectural function in your organization in the absence of a strong central team? Could implementing a Data Pillar and well-connected architects help mitigate this?*

# Being Architect

# Being Architect: Introduction

image by borko manigoda from pixabay

**IN THIS SECTION, YOU WILL:** Get an overview of topics related to what it means to be an architect, covered in this chapter.

**KEY POINTS:**

- I introduce ideas and key lessons I learned about what it means to be an architect in practice.

The following reflections provide some resources, ideas and key lessons I learned about what it means to be an architect in practice:

- **Architects as Superglue**: Architects in IT organizations should develop as "superglue," people who hold architecture, technical details, business needs, and people together across a large organization or complex projects.
- **Skills**: A typical skillset of an architects includes: hard (technical) skills, soft (people & social) skills, and business skills.
- **Impact**: Architects' work is evaluated based the impact they've had on the organization. They must demonstrate that they identify, tackle, and deliver on strategic problems, have a deep and/or broad influence, and deliver solutions that few others can.
- **Leadership**[1]: My view of architecture leadership is inspired by David Marquet's work and the Netflix's valued behaviors.
- **Architects' Career Paths: Raising the Bar**: Architects' career paths ideally stem from a strong engineering background. Hiring architects requires constantly raising the bar to ensure a strong and diverse team structure.

[1] behaviors

# Architects as Superglue


**IN THIS SECTION, YOU WILL:** Understand the view on architects as superglue (people who hold architecture, technical details, business needs, and people together across a large organization or complex projects) and get valuable tips on developing "superglue" abilities.

**KEY POINTS:**

- Architects in IT organizations should develop as "superglue," people who hold architecture, technical details, business needs, and people together across a large organization or complex projects.
- Architects need to be technically strong. But their unique strengths should stem from being able to relate, or glue, technical issues with business and broader issues.
- Architects should stand on three legs: Skills, Impact, Leadership.

In my view, architects in IT organizations should develop as "superglue." I borrow the "superglue" view from Adam Bar-Niv and Amir Shenhav from Intel[1]. They pointed out that instead of the superhero, we need "superglue" architects - the people who hold architecture, technical details, business needs, and people together across large organizations or complex projects. More recently, Tanya Reilly presented a similar view[2] concerning software engineering positions.

The superglue characteristics mean serving as the organizational connective tissue, linking the "business wheelhouse" and the "engine room." Architects, of course, need to be technically strong. But their unique strengths should stem from being able to relate, or glue, technical issues with business and broader issues.

From discussions I've had with our technology leaders, engineers, and architects, the picture below has crystallized as a representation of the "superglue" metaphor for architects (Figure 1).

[1] https://saturn2016.sched.com/event/63m9/cant-find-superheroes-to-help-you-out-of-a-crisis-how-about-some-architecture-and-lots-of-superglue

[2] https://noidea.dog/glue

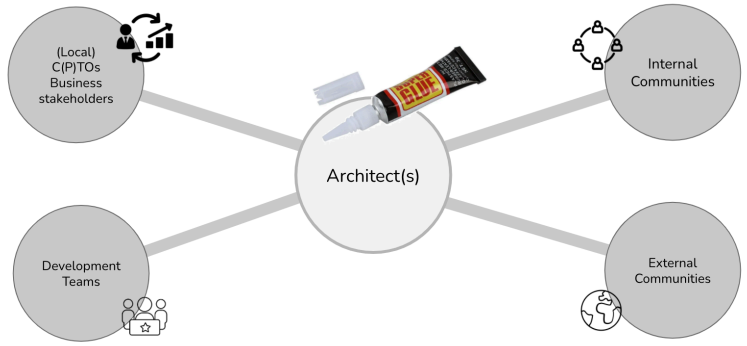



***Figure 1**: Architects serve as a superglue, connecting development teams with business stakeholders while linking teams with the internal communities and the external world.*

Architects must have good relationships with developer teams, local business stakeholders, and functions. Simultaneously, such a person needs to be well-connected with broader internal communities. External visibility is essential for architects, who can bring ideas from outside into the organization and promote the organization to the outside world.

## Supergluing in Action: Reducing Tension among Business Function, Product, Technology, Organization

The primary value of superglue architects in an organization is aligning business, product, technology, and organizational functions. While technology, product, organization, and business function face challenges, additional problems occur when there is tension among them (Figure 2). For example, we may organize teams using a well-defined domain model (organizational design). Still, if our system is a monolith (technical design), our teams will collaborate in a different pattern than domain splits suggest. On the other hand, if our teams are well aligned with the technology implementation (e.g., clear ownership of microservices), but the product architecture differs from the microservice domain split, we may need to change dozens of microservices when introducing relatively simple product features. Similarly, business benefits if they align their objectives with product or technology; otherwise, tense interactions will happen (e.g., try reducing short-term costs while adding new features and migrating to the public cloud).

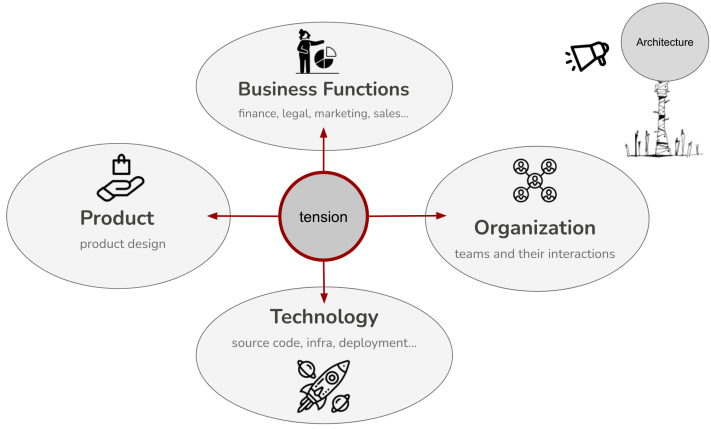



***Figure 2**: The tensions between technology, product, organization,*

*and business functions.*

The main problem of these tensions is that they slow things down due to miscommunications and misalignment, lead to bad decisions due to lack of information, introduce unnecessary complexity, and lead to many missed opportunities. Too frequently, architecture sits on the side, shouting principles and abstract ideals that everyone ignores. By acting as a superglue, the architecture function can help reduce tension between technology, product, organization, and business functions (Figure 3). Architecture should ensure that conversations happen between the technical, product, organizational, and business functions.

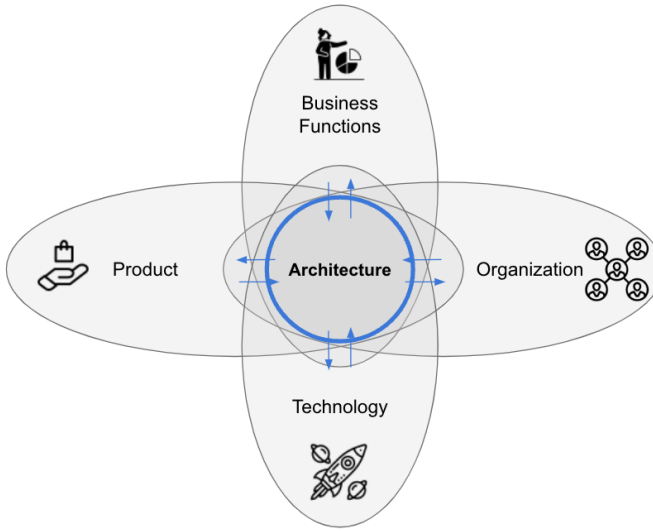



***Figure 3**: Architects should be in the middle of reducing tensions between technology, product, organization, and business functions.*

Getting the product/technology/organizational/business alignment right takes a lot of work, which is one of the main areas of architecture work. There will always be essential tension between system architecture, team organization, and product organization. Ideally, these structures all change simultaneously and stay in

perfect sync. But in practice, these structures change and move at different speeds. All in all, this situation guarantees the job security of superglue architects.

## Superglue Abilities

Setting the architects' goals to be "superglue" also requires some thought on developing architects as a superglue. Borrowing from Gregor Hohpe's view on architect development from his book Software Architecture Elevator, I share the view that our architects should stand on three legs (Figure 4):

- Skills
- Impact
- Leadership


***Figure 4**: Architect Profile: Skills + Impact + Leadership.*

## Skills

Architects have to have proper skill sets. By skills, I mean possessing knowledge and the ability to apply relevant knowledge in practice. These skills should include both technical (e.g., cloud architecture or Kubernetes technology) as well as communication and influence skills.

A typical skillset of an architects includes:

- **Hard (technical) skills**, including extensive knowledge of for both new technology and legacy technology stacks,

- **Soft skills**, and
- **Business Domain knowledge**.

The section Skills provides more details.

## Impact

Impact should be measured as a benefit for the business. Architects need to ensure that what they are doing profits the business:

- They identify, tackle and deliver on **strategic problems** at the organization and area level.
- They have a **track record of deep and/or broad impact** on a product or technology area.
- They **deliver solutions that few others can**, either by your **heavy lifting** or the **ability to orchestrate large group efforts**.

Architects need to get out into the world and make an impact. Architects that do not make an impact **do not have a place in a for-profit business**.

Examples of such impact may include:

- **Aligning** business, product, technology and organizatinal strategies (see this section for more details),
- **Process** optimizations and improvements, with real measurable impact on work of an organization,
- **Cost** optimizations of systems, based on data informed decisions,
- Developing pragmatic **technology strategies**, helping business reach goals in a sustainable way,
- Driving **delivery of products**, supporting teams to increase quality and speed of delivery,

- Supporting **business innovation**, bringing new ideas in a pragmatic way aligned with business strategy and goals.

The section Impact provides more details. In my view, architects in IT organizations should develop as "superglue."

## Leadership

Lleadership acknowledges that experienced architects should do more than make architecture:

- They are a **role model for others** in the company on both the **technical** and **cultural** front.
- Their **technical influence** may extend **beyond you organization and reach the industry at large**.
- They **lead efforts** that **solve important problems** at the engineering area level.
- They may **contribute to the broader technical community** through **tech talks**, **education**, **publications**, **open source projects**, etc.
- They **raise the bar of the engineering culture** across the company.

Mentoring junior architects is the most crucial aspect of senior architects' leadership. Feedback cycles in (software) architecture are inherently slow. Mentoring can save new architects many years of learning by doing and making mistakes. The People Pillar should create spaces for such coaching and collaborations.

The section Leadership provides more details.

## Balanced Development

Architects need to have a minimal "length" of all of these "legs" to be successful (Figure 5). For instance, having skills and impact

without leadership frequently leads to hitting a glass ceiling. Such architects plateau at an intermediate level and cannot lead the company to innovative or transformative solutions. Leadership without impact lacks foundation and may signal that you have become an ivory tower architect with a weak relation to reality. And having impact and leadership qualities but no skills leads to impractical decisions not informed by in-depth knowledge.

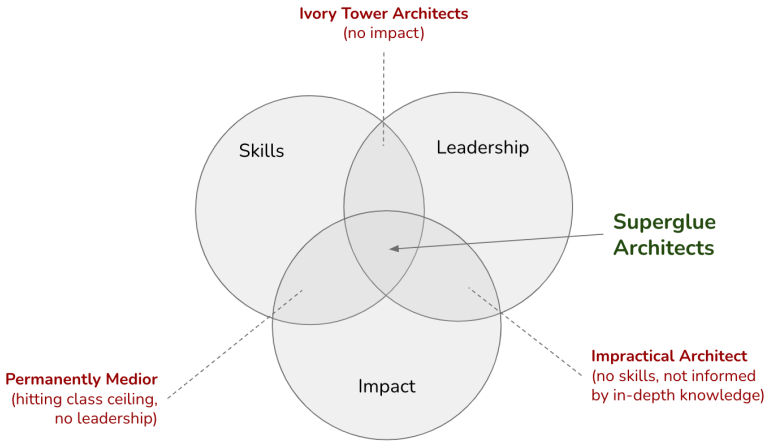



***Figure 5:*** *Architects need to have a minimal "length" of all of these "legs" to be successful.*

## Questions to Consider

Being a superglue architect means constantly developing and redefining your role to benefit a changing organization. Ask yourself the following questions:

- *How well do you think you currently embody the characteristics of a "superglue" architect? Which areas could you improve on to become more effective in this role?*
- *Reflect on your ability to connect the "business wheelhouse" and the "engine room" within your organization. How effectively do you bridge the gap between technical issues and business needs?*
- *How strong are your relationships with developer teams, local business stakeholders, and broader internal communities? What strategies could you employ to strengthen these connections?*
- *How much external visibility do you currently have? How could this be enhanced to promote the flow of ideas into and out of the organization?*
- *Can you identify specific instances where tension occurred between your organization's technology, product, organization, and business functions? What caused this tension, and how was it addressed?*
- *How does your current architecture aid in reducing tension between these functions? If it doesn't, what changes can you make to align these functions better?*
- *Reflecting on your organization, have you witnessed the architecture sitting on the side, being ignored? If so, what steps can you take to involve architecture in decision-making processes actively?*
- *Are conversations between the technical, product, organizational, and business functions encouraged and facilitated within your organization? If not, how might they be initiated and supported?*

- *Considering the three legs of a successful architect (skills, impact, leadership), which do you consider your strongest? Which might need more development?*
- *Assess your technical skills and ability to apply this knowledge in practice. How well do you balance hard (technical) skills, soft skills, and business domain knowledge?*
- *How do you currently measure your impact within your organization? Could you identify and address more strategic problems to benefit the business?*
- *Reflect on your leadership abilities. How are you acting as a role model within your organization and contributing to the broader technical community?*
- *How do you balance skill development, impact, and leadership? Where should you focus more on ensuring a balanced development as an architect?*
- *In what ways could you be more of a mentor for junior architects? What spaces could you create or utilize to facilitate more effective coaching and collaboration?*

# Skills

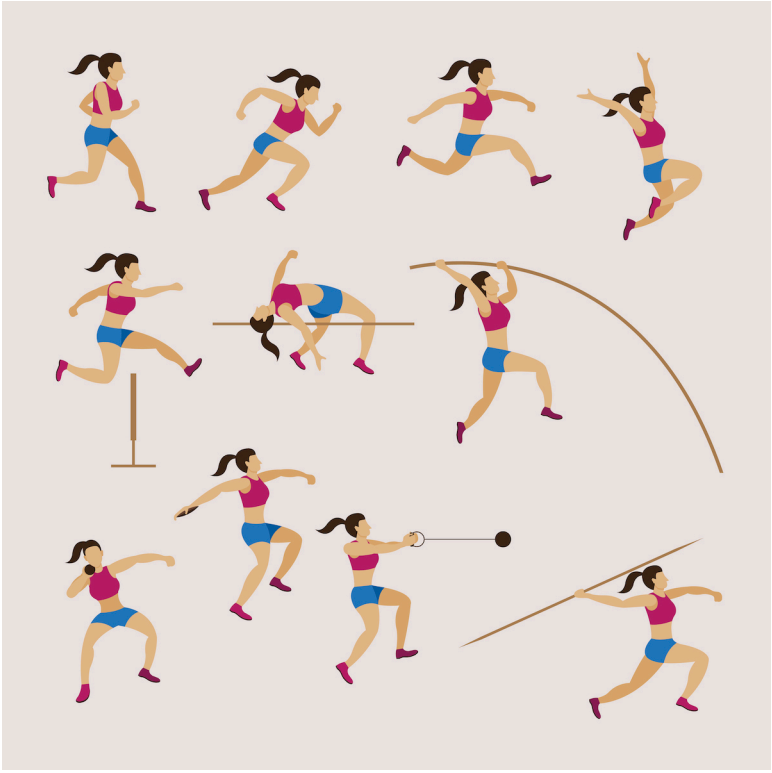


image by istock / muchmania.

**IN THIS SECTION, YOU WILL:** Understand that architects' skills should include a mix of technical, communication, and business skills, and get valuable pointers to resources for developing these skills.

**KEY POINTS:**

- A typical skillset of an architects includes: hard (technical) skills, soft (people & social) skills, and business skills.

Architects have to have proper skill sets. By skills, I mean possessing knowledge and the ability to apply relevant knowledge in practice. These skills should include both technical (e.g., cloud architecture or Kubernetes technology) as well as communication and influence skills.

A typical skillset of an architects includes (I provide links to some of my tools[1] I found useful for obtaining these skills):

- **Hard (technical) skills**, including extensive knowledge of for both new technology and legacy technology stacks,
- **Soft skills**, and
- **Business Domain knowledge**.

[1] https://obren.io/tools

## Hard Skills (Technical Architecture)

Hard skills, also known as technical skills, are the abilities and knowledge needed to perform specific tasks in a professional setting. In the context of technical architecture, these skills are essential for designing, implementing, and maintaining various aspects of an organization's technology infrastructure. Some typical hard skills that architects need in their work include:

- **System design**[2]: System design refers to the process of defining and developing the architecture of a complex system. An architect with this skill set is capable of creating comprehensive system designs that incorporate various components and sub-systems to achieve the desired functionality.
- **Engineering processes**[3]: An architect needs to have an in-depth understanding of engineering processes, including software development life cycle, Agile development, DevOps, and continuous delivery.
- **Design patterns**[4] **and tactics**[5]: A good architect should be familiar with design patterns and tactics such as Cloud Design Patterns, Model-View-Controller (MVC), Service-Oriented Architecture (SOA), and Microservices. These patterns help architects to design systems that are modular, scalable, and maintainable.
- **Security and privacy by design**[6]: In today's world of cybersecurity threats, architects need to have a deep understanding of security and privacy best practices. They must ensure that the systems they design are secure and comply with data protection regulations.

[2] https://blog.pragmaticengineer.com/system-design-interview-an-insiders-guide-review/
[3] https://obren.io/tools/catalogs/?id=design-tactics-high-performing-technology-organizations
[4] https://obren.io/tools?tag=design_patterns
[5] https://obren.io/tools?tag=design_tactics
[6] https://obren.io/tools?tag=security

- **System optimizations**[7]: An architect should know how to optimize systems for performance and scalability. They should be familiar with tools and techniques for profiling and tuning systems to achieve optimal performance.
- **Source code structures and maintainability**[8]: Architects should have a good understanding of software engineering principles such as clean code, code maintainability, and refactoring. They should be able to design systems that are easy to maintain and modify.
- **Reliability and stability (anti)patterns**[9] **and tactics**[10]: An architect should be aware of the common reliability and stability issues that can arise in complex systems. They should be able to identify and address potential problems by using patterns and tactics such as redundancy, failover, and graceful degradation.
- **Usability**[11]: An architect should have a good understanding of usability principles. They should design systems that are easy to use and provide a good user experience.

[7] https://obren.io/tools/catalogs/?id=design-tactics-sig-performance
[8] https://obren.io/tools/catalogs/?id=design-tactics-sig-maintainability
[9] https://obren.io/tools/catalogs/?id=releaseit-stability-awareness
[10] https://obren.io/tools/catalogs/?id=releaseit-stability-tactics
[11] https://obren.io/tools?q=usability

# Soft Skills (Social Architecture)

> To change the architecture of a software-intensive system ensconced in a large organization, you often have to change the architecture of the organization. And ultimately, that is a political problem, not just a technical one. —Grady Booch

Social Architecture refers to the design and management of social systems, interactions, and relationships within an organization or community. Soft skills, often described as non-technical or interpersonal skills, are an integral part of social architecture, as they enable individuals to effectively navigate and contribute to these systems. By developing and refining soft skills, individuals can more easily adapt to changes, collaborate with others, and foster a positive work environment. Key soft skills related to social architecture include:

- **Communication skills**[12], written[13], visual[14], verbal (presentation), and listening skills: Effective communication involves not only expressing oneself clearly, but also being able to understand and empathize with others. This includes written, visual, verbal (presentation), and listening skills, which are crucial for building and maintaining relationships, as well as for conveying ideas and facilitating discussions.
- **Networking and collaboration skills**[15]: Networking involves building and maintaining a diverse range of professional connections, which can be useful for personal growth and career development. Collaboration skills encompass the ability to work effectively with others,

[12] https://obren.io/tools?tag=consultancy
[13] https://obren.io/tools/sowhat/
[14] https://obren.io/tools?tag=visuals
[15] https://obren.io/tools?tag=leadership

regardless of their role or seniority. This includes partnering with peers, junior and senior colleagues, managers, and executives to achieve common goals.

- **Organizational and time management skills**[16]: These skills involve the ability to efficiently plan, prioritize, and manage tasks, resources, and time. Effective organization and time management are crucial for meeting deadlines, achieving goals, and maintaining a healthy work-life balance. Key aspects of these skills include prioritization, goal-setting, task management, and delegation.
- **Analytical, strategic thinking, and problem-solving skills**[17]: Analytical skills involve the ability to assess and interpret complex information to make informed decisions. Strategic thinking is the capacity to envision and plan for long-term success, while problem-solving skills involve identifying and addressing challenges creatively and effectively. These skills are essential for recognizing and capitalizing on unique opportunities and creating value within an organization..

[16] https://obren.io/tools?tag=reflect
[17] https://obren.io/tools?tag=it

## Business Skills (Business Architecture)

Business Architecture refers to the strategic design and alignment of an organization's business processes, structures, and systems to achieve its objectives and create value. Architects, regardless of their technical or design expertise, must possess a solid understanding of business skills to effectively contribute to an organization's success. Key business skills for architects include:

- **General business concepts knowledge** (The Personal MBA[18] book being my favorite resource to get familiar with such concepts): A fundamental understanding of general business concepts is essential for architects to make informed decisions and effectively communicate with stakeholders. Familiarity with concepts such as finance, marketing, sales, operations, and strategy can provide a strong foundation for architects to engage with various aspects of an organization.
- **Specific business domains**[19] of the organization: In addition to general business concepts, architects should also develop a deep understanding of the specific business domain in which their organization operates. This may include knowledge of industry-specific regulations, market trends, customer preferences, competitive landscape, and more. Gaining insights into the specific business domain enables architects to better align their work with the organization's goals, strategies, and priorities.
- **Business analysis and requirements gathering**: Architects should be adept at analyzing business needs and gathering requirements from various stakeholders. This skill involves understanding the organization's objectives and translating them into functional and technical specifications that can guide the design and development of solutions.

[18] https://personalmba.com/
[19] https://obren.io/tools?tag=domain_models

## Questions to Consider

Architects must possess the knowledge and the ability to apply relevant knowledge in practice. Ask yourself the following questions:

- *On a scale from 1 to 10, how would you rate your current architectural skill sets, considering technical, communication, and business skills?*
- *Reflect on your technical skills. How proficient are you in system design, understanding engineering processes, recognizing design patterns and tactics, ensuring security and privacy, optimizing systems, and maintaining code structures?*
- *Are there any specific hard skills you need to develop to enhance your performance as an architect?*
- *Now, think about your soft skills. How effectively do you communicate (in writing, visually, verbally, and through listening)? How strong are your networking and collaboration skills, and how well do you manage your time and organizational tasks?*
- *Can you identify an instance where your problem-solving skills and strategic thinking have significantly influenced your work as an architect?*
- *Looking at business skills, how well do you understand general business concepts, and how familiar are you with the specific business domain of your organization?*
- *How competent are you in business analysis and requirements gathering? Can you share an example where you effectively translated business objectives into functional and technical specifications?*
- *Are there any soft or business skills that you believe you need to develop or improve to be more successful in your role as an architect?*
- *Reflect on how you have used your soft skills to effect organizational change. Are there areas or situations where you could have applied these skills more effectively?*

- *How do you balance developing and maintaining your hard, soft, and business skills? Is there a particular area you tend to focus on more, and if so, why?*

# Impact

image by wikipedia / sandi morris.

**IN THIS SECTION, YOU WILL:** Understand that architects' work is evaluated based on their impact on the organization and get guidelines for making an impact.

**KEY POINTS:**

- Architects' work is evaluated based the impact they've had on the organization.
- Architects can make an impact via three pillars: Big-Picture Thinking, Execution, and Leveling-Up.

Architects' work is **evaluated based the impact** they've had on the organization. Architect typically make impact by:

- **Identifying, tackling, and delivering on strategic problems** at the organizationd and area levels (domain or technical areas). Architects' work needs to be prioritized based on global strategic objectives.
- **Having a deep and/or broad influence** on a domain, product, or technology area. Architects sometimes need to go deep, addressing specific critical issues in one area. And frequently that need to look broad, creating impact by leveraging the results across multiple teams.
- **Delivering solutions that few others can**, sometimes by their heavy lifting but more often by their ability to orchestrate large group efforts. Architects can help move the organization forward by leveraging their "hard" technical skills and "soft" strategic, execution, and people skills.

To make this impact, architect need a few key competencies.

## Pillars of Impact

Architects must have strong technical, people, and business skills, ideally obtained through years of practice. On top of this strong foundation, architects need to develop competencies that enable them to use their experiences and abilities to impact organizational performance positively. The more senior architects become, the more their competency development should be driven by the impact they need to have rather than mere skills development. I typically coach architects in the context of concrete activities, guiding their development via involvement in the right set of actions and crafting skills developments based on challenges in making an expected impact in practice.

I use Staff Engineering roles as an inspiration for the development of architects. Tanya Reilly's book The Staff Engineer's Path[1] and Will Larson's book Staff Engineer: Leadership beyond the management track[2] were helpful guides in defining the responsibilities of architects.

[1] https://www.oreilly.com/library/view/the-staff-engineers/9781098118723/
[2] https://staffeng.com/guides/staff-archetypes/

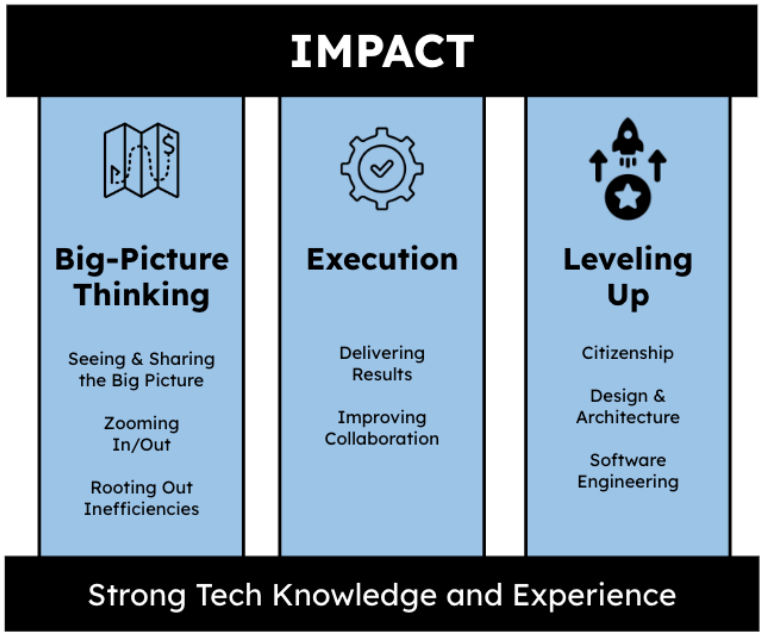



Figure 1: Key competencies of architects. Inspired by The Staff Engineer's Path by Tanya Reilly.

Inspired by The Staff Engineer's Path by Tanya Reilly, I group architects' competencies in three groups (Figure 1):

- Big-picture thinking,
- Execution, and
- Leveling up.

## Big-Picture Thinking

Architects are frequently the only people in the organization with a "helicopter view," overseeing vast domains and being able to foresee the consequences of decisions in a broader context. As big-picture thinkers, architects can help organizations in multiple ways:

- Seeing the **big picture** can identify high leverage points for maximum impact.
- **Helping others to see the big picture** and create tools (e.g., Data Pillar) that facilitate big-picture thinking.
- Being able to **zoom in and out**, having a strategic overview, but being able to go deep and engage with implementaiton details.
- Using big-picture thinking to **consistently root out inefficiencies** and lead the adoption of technologies and processes that **make multiple teams more efficient**.

## Execution

As execution-focused practitioners, architects need to be able to help deliver results, and improve collaboration.

Architects can help delivering results by combining their skills with high dose of pragmatism:

- **Create meaningful solutions**, rather than theorietical ideals and models.
- **Break down complex problems** to enable delivery of impactful results.
- **Craft pragmatic plans** by considering technical, logistical, and organizational constraints.

Architects also can help execution by finding ways to enable others to collaborate and work better:

- **Creating alignment** and **improving collaboration** within their areas or the wider engineering organization.
- **Collaborate meaningfully across groups** in a way that builds trust and incresses execution speed.

## Leveling Up

Architects are frequently seen and leaders and role models that should help organizations to raise the bar on the technical and cultural fronts. I group such role of architect in three categories: citizenship, design and architecture, and software engineering.

### Citizenship

Architects should look broader and raise the bar of practices and behaviors in their organizations:

- **Contribute to the broader technical community** through tech talks, education, publications, open source projects, etc.
- **Have influence that extends** beyond their organization and reach the industry at large.
- **Lead efforts** that solve **important problems** in their areas.
- **Raise the bar** of the engineering culture across the company.

### Design and Architecture

Architects are leading authority for systematic and strategic design and architecture topics:

- **Identify and solve systemic architectural problems**. Architects quickly recognize systemic problems and can **articulate possible solutions** to them.
- **Improve the definition of best practices** and architecture with deep domain knowledge.

### Software Engineering

Lastly, architects can help by staying well-connected to software engineering practice, leveraging their experience to:

- **Promote and/or demonstrate best-in-class** of code, documentation, testing, and monitoring practices.
- **Solve challenging technical and/or execution problems** that few others can.

## Questions to Consider

- *Reflect on the impact you have had on your organization. Have you prioritized your work based on global strategic objectives, and what has been the outcome?*
- *Can you identify instances where you had to go deep into a specific issue and others where you needed a broad perspective across multiple teams? How did you manage both scenarios?*
- *How have you used your technical, strategic, execution, and people skills to deliver solutions? Can you share an example?*
- *How can you build on your technical, people, and business skills to positively impact your organization's performance? How do you measure this impact?*
- *As an architect, how can you develop your big-picture thinking ability? Can you give an example of how your big-picture thinking helped to identify a high leverage point for maximum impact?*
- *Reflect on your role in execution. How can you help in delivering results and improving collaboration? Can you share an example where your pragmatism resulted in a meaningful solution?*
- *What initiatives have you taken to improve collaboration and build trust within your organization?*
- *How have you contributed to the broader technical community through tech talks, education, publications, open source projects, etc.?*
- *In what ways have you helped solve important problems in your area and raise the bar of the engineering culture across the company?*
- *Can you provide an example of a systemic architectural problem you identified and the solution you proposed?*
- *How have you promoted and demonstrated best-in-class practices in coding, documentation, testing, and monitoring?*

- *Reflect on a challenging technical or execution problem you solved that few others could have. What was your approach, and what was the outcome?*

# Leadership

image by david mark from pixabay

**IN THIS SECTION, YOU WILL:** Understand how to apply ideas from David Marquet's work and Netflix's valued behaviors to develop architects' leadership traits.

**KEY POINTS:**

- My view of architecture leadership is inspired by David Marquet's work and the Netflix's valued behaviors.
- Marquet focused on leadership and organizational management, with a particular emphasis on the principles of Intent-Based Leadership.
- Borrowing from Netflix's original values the following behavioral traits are crucial for architects are: communication, judgment, impact, inclusion, selflessness, courage, integrity, curiosity, innovation, and passion.

My approach to architecture leadership draws inspiration from two sources: David Marquet's[1] leadership principles, as articulated in his book "Turn the Ship Around!" and Netflix's valued behaviors. Marquet's ideas emphasize empowering team members, providing clarity, decentralizing decision-making, striving for continuous improvement, and practicing servant leadership. Meanwhile, Netflix valued behaviors[2] offer useful guidance for coaching and developing architects aligned with the "superglue" version.

[1] https://davidmarquet.com/
[2] https://jobs.netflix.com/culture

## David Marquet's Work: The Leader-Leader Model

Marquet's work is closely tied to the Leader-Leader model of leadership, a style where authority is shared across a team or organization instead of being concentrated at the top. In this model, every team member has something valuable to contribute and can work together toward the group's success.

This leadership approach empowers individuals to take ownership of their work and collaborate with others to achieve common goals. Instead of relying on a single leader to make all decisions, authority, and responsibility are distributed across the team.

The leader-leader model is an excellent standard for architects' leadership vision. Like managers in a leader-leader model, architects should act more as facilitators, coaches, and mentors than traditional top-down decision-makers. They provide team members guidance, support, or resources to help them achieve their goals and reach their full potential.

One of the key benefits of a leader-leader model is that it creates a more collaborative and inclusive work environment. It allows individuals to contribute their unique perspectives, experiences, and skills to the group, promoting a sense of ownership and accountability for the team's success. This model also helps build trust and stronger relationships within the team, leading to increased productivity, creativity, and innovation.

image by istock

David Marquet's book "Leadership is Language" provides practical advice for leaders looking to create a more collaborative, innovative, and inclusive organizational culture. He emphasizes the importance of language and communication in leadership and introduces the phrase "I intend to" as a powerful tool for clarifying intent and empowering team members (Figure 1). When team members give intent, the psychological ownership of those actions shifts to them, making them the originators of thought and direction instead of passive followers. This shift in language helps to promote a more collaborative work environment.

| LEADER | LEADER |
|---|---|
| 7. I've been doing... | 7. What have you been doing? |
| 6. I've done... | 6. What have you done? |
| 5. I intend to... | 5. What do you intend? |
| 4. I would like to... | 4. What would you like to do? |
| 3. I think... | 3. What do you think? |
| 2. I see... | 2. What do you see? |
| 1. Tell me what to do. | 1. I'll tell you what to do. |
| WORKER | BOSS |

***Figure 1**: Leadership language. Based on Intent-Based Leadership, by David Marquet.*

I have found a phase "I intend to" to be a powerful catalyst for positioning architecture work. The phrase helps describe the work architect as someone others expect to take the initiative and lead efforts. But also to describe the desired interaction of architects with the teams they work with, where we hope teams share their intentions which architects can help improve.

# Netflix Valued Behaviors: Leadership Behaviors

I have found the Netflix overview of their valued behaviors[3] to be a leading inspiration for how I coach and develop architects. The following sections summarize these behaviors, borrowing from the Netflix original values but rearranging them in the order I see as more relevant for architects.

## Communication

Architects can only be successful if they are effective communicators. More specifically, as an architect, you need to have the following communication traits:

- You are **concise** and articulate in **speech and writing**
- You **listen well** and **seek to understand** before reacting
- You maintain **calm poise in stressful situations** to draw out the clearest thinking
- You **adapt your communication style** to work well with people from around the world who may not share your native language

## Judgment

People frequently call architects to be objective judges when others cannot agree or need an objective second opinion. As an architect, you'll be able to make sound judgments if:

- You are good at using **data to inform your intuition**
- You make wise **decisions** despite **ambiguity**

[3] https://jobs.netflix.com/culture

- You identify **root causes**, and go beyond treating symptoms
- You make decisions based on **the long term**, not near term
- You **think strategically**, and **can articulate** what you are, and are not, trying to do

## Impact

As discussed in the Architects as Superglue, the architect's impact should be measured as a benefit for the business. Architects need to ensure that what they are making profits the company. As an architect, you need to show the following impact traits:

- You accomplish significant amounts of **important work**
- You **make your colleagues better**
- You focus on **results over process**
- You demonstrate **consistently** strong performance so **colleagues can rely upon you**

## Inclusion

As superglue, architects need to work with many different people and groups inclusively. You will be able to do so if:

- You **collaborate** effectively with people of **diverse backgrounds and cultures**
- You nurture and **embrace differing perspectives** to make better decisions
- You **focus on talent and values**, rather than a person's similarity to yourself
- You are **curious about how our different backgrounds affect us** at work, rather than pretending they don't affect us

## Selflessness

Architects always need to consider the best interests of their organizations. This broader view is essential in group conflicts to enable resolutions that benefit the organization. To be able to operate in such way, you need to show the following selflessness traits:

- You seek what is **best for your organization**, rather than what is best for yourself or your group
- You **share information openly and proactively**
- You **make time to help colleagues**
- You are **open-minded** in search of the best ideas

## Courage

Being an architect is not always a comfortable position as you will need to be a part of difficult decisions many will not be happy about. You need to have enough courage to make such difficult calls. You will be able to do so if you show the following traits:

- You **say what you think** when it's in the best interest of your organization, even if it is uncomfortable
- You are willing to be **critical of the status quo**
- You make **tough decisions** without agonizing
- You **take smart risks** and are open to possible failure
- You **question actions inconsistent** with organization's values
- You are **able to be vulnerable**, in search of truth

## Integrity

Architects need to operate as trusted advisors. Integrity is essential for such a position of architects. To operates successfully as trusted advisor, you need to show the following traits:

- You are known for **candor**, authenticity, **transparency**, and being **non-political**
- You only say things about fellow employees that you **say to their face**
- You **admit mistakes** freely and openly
- You **treat people with respect** independent of their status or disagreement with you

## Curiosity

As architects, we must proactively identify relevant new technology developments. Based on our understanding of these developments, we must create pragmatic technology recommendations for concrete platforms across the organization. That means that as architect, you need to stay curious:

- You **learn rapidly and eagerly**
- You **make connections** that others miss
- You seek **alternate perspectives**
- You **contribute effectively** outside of your specialty

## Innovation

More than curiosity is required. To make an impact as an architect, you need to create useful innovations:

- You create **new ideas** that **prove useful**
- You keep your organization nimble by **minimizing complexity** and finding time to simplify
- You **re-conceptualize issues** to discover solutions to **hard problems**
- You **challenge prevailing assumptions**, and suggest better approaches
- You **thrive on change**

## Passion

Architects are frequently role models for others. As such you need to show the following traits:

- You **inspire others** with your thirst for excellence
- You **care intensely** about your customers and organization's success
- You are **tenacious and optimistic**
- You are **quietly confident** and **openly humble**

## Questions to Consider

- *Reflect on the Leader-Leader model of leadership in your work. How can you empower your team members and encourage them to take ownership of their work?*
- *Have you acted as a facilitator, coach, or mentor as an architect? Can you share an example of when you gave team members guidance, support, or resources to achieve their goals?*
- *How does the phrase "I intend to" resonate with your approach to architecture work? How can it change your perspective on taking the initiative and leading efforts?*
- *How effective do you believe your communication skills are? Have you adapted your style to work with people from diverse backgrounds and languages?*
- *How can you foster an inclusive working environment as an architect? How do you nurture and embrace differing perspectives to make better decisions?*
- *Reflect on a situation where you made a decision that was best for the organization rather than what was best for yourself or your group. What was the outcome?*
- *Have you ever had to take an uncomfortable stance but in your organization's best interest? Can you share this experience?*
- *How do you maintain integrity as a trusted advisor in your organization? Can you share an example where your honesty, authenticity, and transparency were vital?*
- *How have you maintained your curiosity in your role as an architect? Can you share an instance where your learning eagerness led to a significant outcome?*
- *What innovative solutions have you created as an architect? How have these innovations benefitted your organization?*
- *How do you inspire others with your passion for excellence? Can you share an instance where your optimism and tenacity led to a successful outcome?*

# Architects' Career Paths: Raising the Bar

image by istock tableatny (cc by 2 0)

**IN THIS SECTION, YOU WILL:** Get ideas and tips about developing architects' career paths.

**KEY POINTS:**

- Architects' career paths ideally stem from a strong engineering background.
- Hiring architects requires constantly raising the bar to ensure a strong and diverse team structure.

Hiring and developing architects will differ significantly per organization. Nevertheless, here I share some of the ideas and lessons learned here.

## Typical Architect's Career Paths

My view of architecture has a strong engineering bias. Architects' career paths ideally stem from a strong engineering background. While there may be exceptions, without significant real-world exposure to software engineering challenges, an architect cannot obtain enough practical knowledge to make technology decisions and build relations with developer teams.

Regarding career progression, Figure 1 shows an example of architecture career paths in relation to engineering, which I used to define architecture career paths.

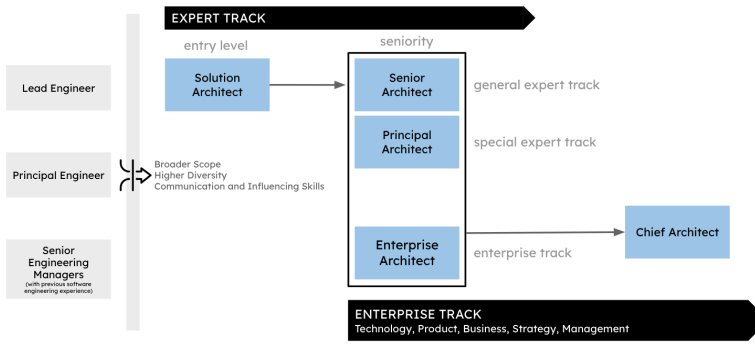



***Figure 1**: An example of architecture career paths in relation to engineering.*

Stepping from an engineering position to an architecture requires three changes:

- Getting a broader scope of work,
- Having a higher diversity of work, and
- Changing skills, as communication and influencer skills become crucial aspects of success.

All architect are responsible for the direction, quality, and approach within some critical area. They need to combine in-depth knowl-

edge of technical constraints, user needs, and organization level leadership.

After the role of an Architect, I usually envision three tracks of progression:

- **Senior Architect**, a generalist with broader responsibilities to, who digs deep into complex issues and identifies a suitable course of action. They often navigate from one critical area to another, guided by the organization's direction.
- **Principal Architect**, a senior architect with a special focus on some area of strategic interest for an organization (e.g., data, distributed systems, frontend).
- **Enterprise Architect**, being closer to product, management, strategy, and business functions, frequently serving as senior engineering leaders' right hand.

But an architect's path can take many different directions and have many other names. More important than a formal title is a continuous search for staying relevant and making an impact.

# Hiring Architects

Developing and hiring architects requires constantly raising the bar to ensure a strong and diverse team structure. Having more architects does not necessarily leads to a better team. Having good alignment and diversity of perspective is even more important for an architecture team than for other groups.

It is vital to take more active ownership of hiring architects. Due to the vast diversity of how different companies define the architect's role, recruiters may need help understanding the role's requirements.

While you will need to design your hiring process, the hiring process should ensure a solid evaluation of the candidate's:

- **Technical skills**: An architect must possess a strong technical background in the relevant areas, such as software development, infrastructure, cloud computing, and security. The process can assess their expertise through technical questions, tests, or case studies.
- **Communication and collaboration skills**: Architects often work with stakeholders, including business leaders, developers, and project managers. Therefore, the process could evaluate the candidate's ability to communicate effectively, work in a team, and manage stakeholders.
- **Leadership and problem-solving abilities**: As a senior member of the team, an architect should have strong leadership skills and the ability to solve complex problems. The process could assess the candidate's experience leading teams, making critical decisions, and resolving technical challenges.
- **Cultural fit**[1]: The process could also evaluate the candidate's fit with the company's culture, values, and mission. The cultural fit is vital to ensure the candidate shares the same vision and will likely thrive in the organization.

[1] Leadership

In terms of steps, I typically work with some version of the following process (after standard recruitment screening):

- **Step 1: Initial Screening Interview with Chief Architect**
  - Typical duration 60 min
  - In this step, it is crucial to assess the candidate's overall fit for the role and determine whether they possess the necessary skills, experience, and qualifications.
  - Overall, the initial screening aims to identify the most promising candidates who possess the necessary skills, experience, and fit for the role of a senior solutions architect and who should proceed to the next stage of the interview process.
  - Extra focus on:
    * Cultural fit
    * Leadership and problem-solving abilities
- **Step 2: In-Depth Interview with Senior/Principle/Enterprise Architects**
  - Typical duration 90 min
  - Extra focus on:
    * Evaluating the candidate's technical skills
    * Assessing the candidate's communication and collaboration skills
    * Understanding the candidate's leadership and problem-solving abilities
- **Step 3: In-Depth Interview with Architects and Senior Engineers**
  - Typical duration 90 min
  - Preparation:

    * A document describing a recent solution architecture of a candidate, providing the content for discussion and helping estimate the candidate's written skills.
    * (Optional) open-source code review of a candidate
  - Extra focus on:
    * Any topics identified during Step 2 as areas that needed to explore further.

For senior positions, I typically introduce an additional step of meeting senior leadership:

- **Step 4: Non-technical stakeholders evaluation**
  - Interview with Engineering Leaders
  - Interview with Product and Business Function Leaders (e.g., CPO, CMO, CFO)
  - Interview with a CTO
  - Extra focus on:
    * Leadership abilities
    * Communication and collaboration skills

With the described steps, you can get a solid overview of all critical aspects of superglue architects. In particular, the involvement of people outside architecture or engineering is crucial to minimize risk related to a lack of interest and ability to engage with all relevant stakeholders.

## Questions to Consider

- *Reflect on your career path in architecture. How has your engineering background impacted your effectiveness as an architect?*
- *How did your transition from an engineering position to an architecture role change your scope of work, diversity of work, and skills?*
- *As an architect, how do you balance technical constraints, user needs, and organization level leadership? Can you share an example of how you've navigated this in your work?*
- *Reflect on your career progression in architecture. How have you continuously stayed relevant and made an impact in your role?*
- *If you were involved in the hiring process for architects, how would you assess a candidate's technical skills, communication and collaboration skills, leadership and problem-solving abilities, and cultural fit?*
- *What strategies would you implement to ensure that you are continuously raising the bar in developing and hiring architects in your organization?*
- *How have you demonstrated your communication and collaboration skills in your role as an architect? Can you share an instance where these skills were crucial?*
- *How would you describe your leadership and problem-solving abilities? Can you share an example of how you've used these skills in your work?*
- *Reflect on the cultural fit between you and your organization. How do your values align with those of the company?*
- *What steps would you include in your hiring process for architects to ensure a solid evaluation of the candidates?*
- *How would you ensure diversity of perspectives within your architecture team, and why do you think this is important?*

- *In your experience, what are some critical aspects of superglue architects that a hiring process should evaluate?*

# Doing Architecture

# Doing Architecture: Introduction

image by enrique meseguer from pixabay

**IN THIS SECTION, YOU WILL:** Get a summary of the articles about doing architecture.

**KEY POINTS:**

- I introduce several principles and ideas I found helpful for running the Grounded Architecture practice.

In the following sections, drawing inspiration from different sources, I will introduce several principles and ideas I found helpful for running the Grounded Architecture practice:

- **The Culture Map: Architects' Culture Mindfield Compass**: In multinational organizations, architects will need to work with many different cultures. I have found the work of Erin Meyer, The Culture Map, to be a very helpful tool for architects to work harmoniously with people from a broad array of different cultures and backgrounds.
- **Managing Organizational Complexity: Six Simple Rules**: Six Simple Rules emphasis that in today's complicated business environment, you need to setup organisational structures based on cooperation. To deal with complexity, organizations should depend on the judgment of their people and on these people cooperating to utilize the organization's capabilities to cope with complex problems. This view is well aligned with the ideas of Grounded Architecture.
- **Product Development and The Build Trap**: Recognize bad product-development approaches. Recognize bad product manager archetypes. Know how a good product development process looks like.
- **Architecture Governance: Mandates, Taxation, Nudge**: I promote a technology governance model that combines three different styles of governing: mandates and bans, taxes, and nudging.

- **Economic Modeling: ROI and Financial Options**: I sketch two answers to the question of the economic value of technology investments and architecture: the return on investment metaphor, and the financial options metaphor.

# The Culture Map: Architects' Culture Mindfield Compass

image by maik from pixabay

**IN THIS SECTION, YOU WILL:** Get an introduction to The Culture Map, a helpful tool for architects to work harmoniously with people from a broad array of different cultures and backgrounds.

**KEY POINTS:**

- I have found the work of Erin Meyer, The Culture Map, to be a very helpful tool for architects to work harmoniously with people from a broad array of different cultures and backgrounds.
- Meyer's model contains eight scales, each representing a key area, showing how cultures vary from extreme to extreme: Communicating, Evaluating, Persuading, Leading, Deciding, Trusting, Disagreeing, and Scheduling.

In multinational organizations, architects will need to work with many different cultures. I have found the work of Erin Meyer, The Culture Map, to be a beneficial tool to work harmoniously with people from a broad array of different cultures and backgrounds. Awareness of cultural differences is even more important for architects, as they are bridging diverse cultures and domains (technology, business, domain, organization).

Meyer's model contains eight scales, each representing a key area, showing how cultures vary from extreme to extreme. The eight scales describe a continuum between the two ends which are diametric opposite or competing positions:

- **Communicating** – Are cultures low-context (simple, verbose, and clear), or high-context (rich deep meaning in interactions)?
- **Evaluating** – When giving negative feedback, does one give it directly or prefer being indirect and discreet?
- **Persuading** – Do people like to hear specific cases and examples or prefer detailed holistic explanations?
- **Leading** – Are people in groups egalitarian or prefer hierarchy?

- **Deciding** – Are decisions made in consensus or made top-down?
- **Trusting** – Do people base trust on how well they know each other or how well they work together?
- **Disagreeing** – Are disagreements tackled directly, or do people prefer to avoid confrontations?
- **Scheduling** – Do people see time as absolute linear points or consider it a flexible range?

The Culture Map shows positions along these eight scales for many countries. These profiles reflect the value systems of a society at large, not those of all the individuals in it, so if you plot yourself on the map, you might find that some of your preferences differ from those of your culture.

# 1. Communicating

Architects need to be good communicators[1]. But what do we mean when saying someone is a good communicator? The responses differ wildly from society to society.

image by getty images / istock / luckybusiness.

Meyer compares cultures along the Communicating scale by measuring the degree to which they are high- or low-context, a metric developed by the American anthropologist Edward Hall.

In low-context cultures, good communication is precise, simple, explicit, and clear. People take messages at face value. Repetition, clarification, and putting messages in writing are appreciated.

In high-context cultures, communication is sophisticated, nuanced, and layered. Statements are often not plainly stated but implied. People put less in writing, more is left open to interpretation, and understanding may depend on reading between the lines.

[1] https://architectelevator.com/strategy/complex-topics-stick/

Architects should be able to understand and adapt to different communication styles. But when actively communicating, I find it crucial that architects provide low-context explanations. Architects will deal not only with the diverse cultural backgrounds of people but with different professional communities (technology, product, marketing, sales, finance, strategy), each with their own specific cultures and buzzwords. To bridge such diverse communities, communicating in a culture-sensitive and buzzword-free way is a valuable skill for any architect.

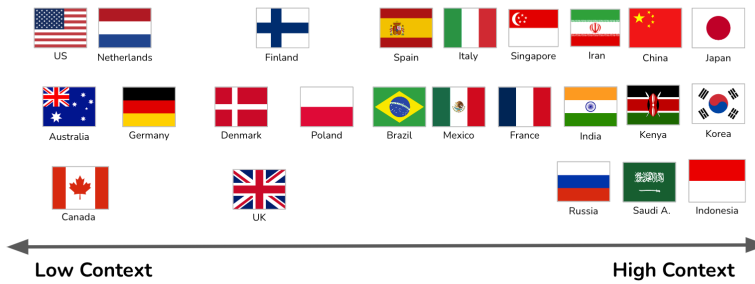



***Figure 1**: Countries on the Communicating scale.*

## 2. Evaluating

Architects need to provide constructive criticism of the plans and ideas of others. All cultures believe that people should give criticism constructively, but the definition of "constructive" varies greatly.

image by rickey123 from pixabay

The Evaluating scale measures a preference for frank versus diplomatic negative feedback. Evaluating is different than the Communicating scale and many countries have different positions on the two scales. According to Meyers, the French, are high-context (implicit) communicators relative to Americans. Yet they are more direct in their criticism. Spaniards and Mexicans are at the same context level, but the Spanish is much franker when providing negative feedback.

Providing constructive criticism in the right way is crucial for architects to make any impact. Sometimes the same feedback will lead to different reactions, even within the same teams with members from diverse backgrounds. Being too positive in some cultures leads

to underestimation of the significance of the feedback. Being too negative may result in pushback and rejection. In my experience, architects need to adapt their feedback to the audience and do lots of "duplication" by presenting the same feedback differently to diverse groups.

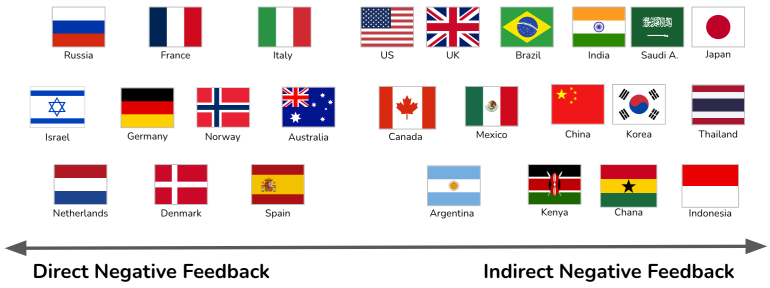



***Figure 2**: Countries on the Evaluating scale.*

## 3. Persuading

Architects frequently need to persuade others. How you influence others and the arguments you find convincing are deeply rooted in culture's philosophical, religious, and educational assumptions and attitudes.

image by istock / fizkes.

One way to compare countries along the Persuading scale is to assess how they balance holistic and specific thought patterns. According to Meyers, a Western executive will break down an argument into a sequence of distinct components (specific thinking). At the same time, Asian managers tend to show how the pieces fit together (holistic thinking). Beyond that, people from southern European and Germanic cultures tend to find deductive arguments (principles-first arguments, building the conclusion from basic premises) most persuasive. In contrast, American and British managers are more likely to be influenced by inductive logic (applications-first logic).

Architects need to be able to persuade in both applications-first

and principles-first ways. In addition to cultural differences, the additional complication comes from talking to diverse audiences. For instance, C-level executives typically have less time and may prefer applications-first presentations ("get to the point, stick to the point"). While in other parts of the company, you may need to spend a long time carefully building the argument following the principal first approach. I typically aim to prepare well for both, having a short management summary and easily retrievable all supporting evidence.

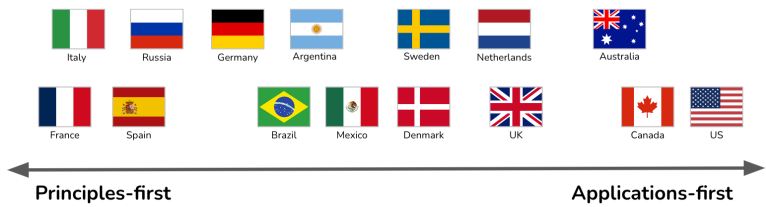



***Figure 3**: Countries on the Persuading scale.*

# 4. Leading

Architects have informal and sometimes formal authority. The leading scale measures the degree of respect and deference shown to authority figures.

This scale places countries on a spectrum from egalitarian to hierarchical. Egalitarian cultures expect leading to be in a democratic fashion. Hierarchical cultures expect leading to be from top to bottom.

image by istock / jacoblund.

The difference in leadership styles can make an architect's work challenging. The same leadership style can lead different people to perceive an architect as weak (no leadership) and too hard (a dictator). The only way to create a working situation is to have an open conversation with the team and agree on expectations and the leadership approach.

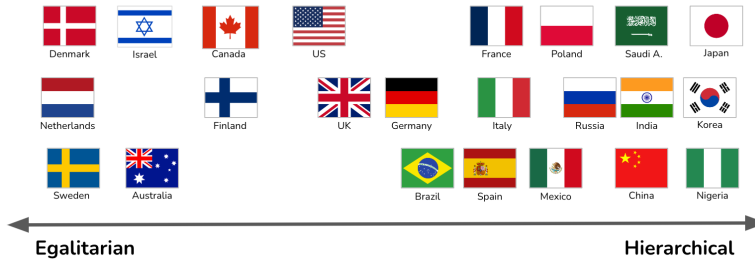



***Figure 4**: Countries on the Leading scale.*

# 5. Deciding

Architectural work is about making decisions[2]. The Deciding measures the degree to which a culture is consensus-minded.

image by istock / fizkes.

According to Meyers, we often assume that the most egalitarian cultures will be the most democratic, while the most hierarchical ones will allow the boss to make unilateral decisions. But this is not always the case. Germans are more hierarchical than Americans but more likely than their U.S. colleagues to build group agreements before making decisions. The Japanese are both strongly hierarchical and strongly consensus-minded.

Similar to the Leading scale, the difference in deciding styles can make an architect's work complicated. I have been in situations where the different members of the same team have had radically different expectations regarding decision-making: some were sitting and waiting for an architect to come up with a decision, and others were offended by any decision that was not complete consensus. Again, the only way to create a working situation is to have an open conversation with the team and agree on expectations

[2] https://architectelevator.com/gregors-law/

and the decision approach. One approach I used is a hybrid option: agreeing with a team to try to come up with a decision based on consensus but delegating the decision to an architect when an agreement was impossible.

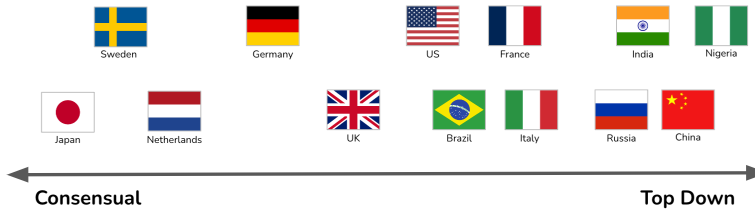



***Figure 5**: Countries on the Deciding scale.*

# 6. Trusting

Architects need to build trust with multiple stakeholders. The culture map scale defines two extremes; task-based cognitive trust (from the head) and relationship-based affective trust (from the heart).

image by istock / scyther5.

In task-based cultures, trust is built cognitively through work. We feel mutual trust if we collaborate well, prove ourselves reliable, and respect one another's contributions.

In a relationship-based society, trust results from weaving a solid affective connection. We establish trust if we spend time laughing and relaxing together, get to know one another personally, and feel a mutual liking.

Without trust, architects' impact is limited. In my view, the best way for architects to build trust is to align their working methods with the rituals of the teams they are working with. In particular, finding time to attend events such as all-hands or off-site gatherings of groups and having regular 1:1 meetings with key stakeholders can be an efficient way to gain trust.

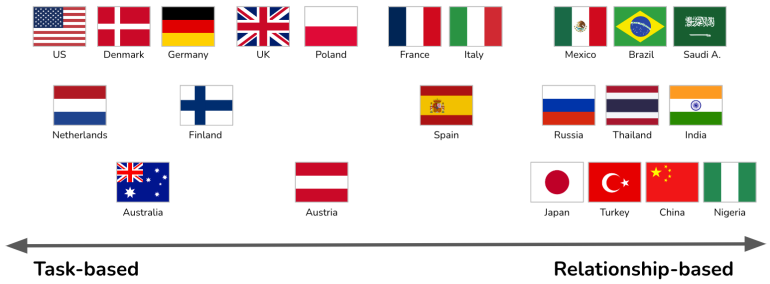



***Figure 6**: Countries on the Trusting scale.*

# 7. Disagreeing

Architectural work may lead to many disagreements and conflicts. Different cultures have very different ideas about how productive confrontation is for a team or an organization. This scale measures tolerance for open debate and inclination to see it as either helpful or harmful to collegial relationships.

image by istock / fizkes

Like the Leading and Deciding scales, architects need to have an open conversation with the team and agree on how to disagree. Disagreeing is an unavoidable part of the work of architects that want to make an impact. Due to the higher diversity of their audiences, architects must also be extra attentive to the cultural aspects of disagreeing to avoid taking too personally what others consider a routine work discussion.

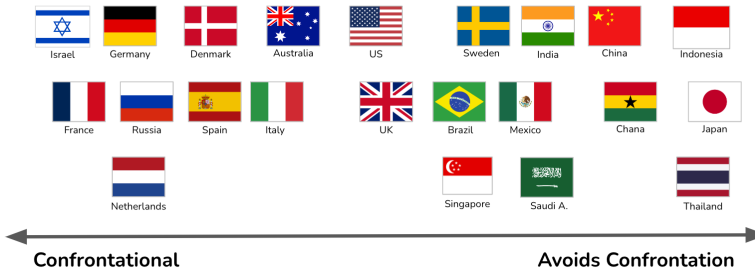



***Figure 7:*** *Countries on the Disagreeing scale.*

# 8. Scheduling

Architects will need to participate in many meeting and projects. All businesses follow agendas and timetables, but in some cultures, people strictly adhere to the schedule. In others, they treat it as a suggestion. The Scheduling scale assesses how much people value operating in a structured, linear fashion versus being flexible and reactive. This scale is based on the "monochronic" and "polychronic" distinction formalized by Edward Hall.

image by istock / bobex_73.

Due to more exposure to diverse audiences, my rule of thumb is that architects should be on time according to the more linear interpretation and tolerate those who are not. But more importantly, adapt to the overall rhythms.

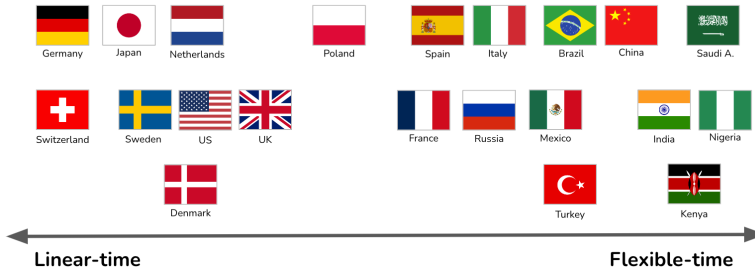



***Figure 8**: Countries on the Scheduling scale.*

## Rules

I also found Erin Meyer's four rules on how to bridge the cultural gaps:

- **Rule 1: Don't Underestimate the Challenge**. It's not always easy to bridge cultural gaps. Management styles stem from habits developed over a lifetime, which makes them hard to change.
- **Rule 2: Apply Multiple Perspectives**. Where a culture falls on a scale doesn't in itself mean anything. What matters is the position of one country relative to another.
- **Rule 3: Find the Positive in Other Approaches**. People tend to see the negative when looking at how other cultures work. But if you understand how people from varied backgrounds behave, you can turn differences into the most significant assets.
- **Rule 4: Adjust and Readjust, Your Position**. It's not enough to shift to a new position on a single scale; you'll need to widen your comfort zone to move more fluidly back and forth along all eight.

## Keep An Open Mind

While cultural generalizations, like the culture map, can be helpful, it is crucial to recognize that they are just that - generalizations. Not all individuals from a particular culture will fit neatly into these categories, and there can be significant variation even within a single culture. It is best to approach cultural differences with an open mind and a willingness to learn.

## To Probe Further

- The Culture Map: Decoding How People Think, Lead, and Get Things Done Across Cultures[3], by Erin Meyer, 2014
- Navigating the Cultural Minefield[4], by Erin Meyer, Harvard Business Review, 2014
- Increase Your Team's Performance with The Culture Map[5], by Andreea, 2019
- The Culture Map Lecture Notes[6], 2023

[3] https://www.goodreads.com/en/book/show/22085568
[4] https://hbr.org/2014/05/navigating-the-cultural-minefield
[5] https://awarenessagents.wordpress.com/2019/06/07/increase-your-teams-performance-with-the-culture-map/
[6] https://docs.google.com/presentation/d/1yn-IHK9iXJ1Qgxj_uw16WKQ1Qp9uXKGSSUbJg0xT9eE/edit?usp=sharing

## Questions to Consider

- *How would you describe your communication style based on Erin Meyer's model? Are you more low-context or high-context?*
- *How do you prefer to give and receive feedback? Do you prefer a more direct or indirect approach?*
- *When it comes to persuasion, do you prefer specific cases and examples or more holistic explanations?*
- *How do you see leadership? Do you prefer a hierarchical or egalitarian structure in your work environment?*
- *What's your approach to decision-making? Do you prefer consensus or top-down decisions?*
- *How do you build trust? Do you base it more on personal relationships or work-based achievements?*
- *How do you handle disagreements? Do you prefer to tackle them directly or avoid confrontations?*
- *How do you perceive time and schedule? Do you consider time linear and absolute or a flexible range?*
- *What strategies do you use to adapt to the communication styles of different cultures and professional communities?*
- *How do you adjust your leadership or decision-making approach when dealing with team members from different cultures?*
- *How do you maintain trust in multicultural environments? What challenges have you faced in this regard?*
- *How do you handle disagreements in a multicultural context?*
- *In which areas of Meyer's model could you improve?*
- *How would you handle a situation where different members of the same team have radically different expectations regarding decision-making or disagreeing?*

# Managing Organizational Complexity: Six Simple Rules

image by nat aggiato from pixabay

**IN THIS SECTION, YOU WILL:** Get an introduction to Six Simple Rules, a model for setting up organizational structures based on cooperation.

**KEY POINTS:**

- Six Simple Rules emphasis that in today's complicated business environment, you need to setup organisational structures based on cooperation.
- To deal with complexity, organizations should depend on the judgment of their people and on these people cooperating to utilize the organization's capabilities to cope with complex problems.
- This view is well aligned with the ideas of Grounded Architecture.

The book Six Simple Rules: How to Manage Complexity without Getting Complicated[1], by Yves Morieux and Peter Tollman, is another source of inspiration for my vision on the Architecture function. Morieux and Tollman introduced a concept of Smart Simplicity with six rules or strategies that enable organizations to promote new behaviors and improve performance.

Six Simple Rules emphasis that in today's complicated business environment, you **need to setup organisational structures based on cooperation**. To deal with complexity, organizations should depend on the judgment of their people, which requires giving them more autonomy to act. It also depends on these people cooperating to utilize the organization's capabilities to cope with complex problems.

Six Simple Rules has been a very inspirational source of my work. In my organizations, I constantly dealt with complex transformations to create organizational structures based on cooperation.

Grounded Architecture is well suited to support and grow organizational structures based on cooperation. In this section, I will explore

[1] https://www.bcg.com/capabilities/organization/smart-simplicity/six-rules-overcoming-complexity

the connection between the ideas of Grounded Architecture and Six Simple Rules.

## Background: Limitations of Hard and Soft Management Approaches

One of the book's central premises is that conventional management approaches, which the authors split into hard and soft, are neither sufficient nor appropriate for the complexity of organizations nowadays.

The **hard approach** rests on two fundamental assumptions:

- The first is the belief that **structures, processes, and systems** have a direct and predictable effect on performance, and as long as managers pick the right ones, they will get the performance they want.
- The second assumption is that the **human factor is the weakest and least reliable link** of the organization and that it is essential to **control people's behavior through the proliferation of rules** to specify their actions and through financial incentives linked to carefully designed metrics and key performance indicators (KPIs) to motivate them to perform in the way the organization wants them to.

When the company needs to meet new performance requirements, the **hard response is to add new structures, processes, and systems** to help satisfy those requirements, hence, the introduction of the innovation czar, the risk management team, the compliance unit, the customer-centricity leader, and the cohort of coordinators and interfaces that have become so common in companies.

On the other end, we have a soft management approach. According to the **soft approach**, an organization is a set of **interpersonal relationships and the sentiments** that govern them.

- **Good performance is the by-product of good interpersonal relationships.** Personal traits, psychological needs, and mindsets predetermine people's actions.

- To change behavior at work, you need to **change the mindset (or change the people)**.

Both approaches are limited in today's world and are harmful to cooperation. A **hard approach introduces complicated mechanisms** and compliance and "checking the box" behaviors instead of the engagement and initiative to make things work. The **soft approach's emphasis on good interpersonal feelings creates cooperation obstacles** as people do not want to ruin good feelings.

## Six Simple Rules Overview

Six Simple rules cover two areas: **autonomy** and **cooperation**. The first three rules create the conditions for **individual autonomy and empowerment** to improve performance.

- **Understand what your people do**. Trace performance back to behaviors and how they influence overall results. Understand the context of goals, resources, and constraints. Determine how an organization's elements shape goals, resources, and constraints.
- **Reinforce integrators**. Identify integrators—those individuals or units whose influence makes a difference in the work of others—by looking for points of tension where people are doing the hard work of cooperating. Integrators bring others together and drive processes.
- **Increase the total quantity of power**. When creating new roles in the organization, empower them to make decisions without taking power away from others.

The Six Simple Rules' authors emphasize the difference between Autonomy and Self-Sufficiency. **Autonomy** is about fully mobilizing our intelligence and energy to **influence outcomes**, including those **we do not entirely control**. **Self-sufficiency** is about **limiting our efforts** only to those **outcomes that we control entirely without having to depend on others**. Autonomy is essential for coping with complexity; **self-sufficiency is an obstacle** because it **hinders the cooperation** needed to make autonomy effective.

This difference between **Autonomy** and **Self-Sufficiency** leads us to the second set of rules that compels people to confront complexity and use their newfound autonomy to cooperate with others so that **overall performance**, **not just individual performance**, is radically improved.

- **Increase reciprocity**. Set clear objectives that stimulate mutual interest to cooperate. Make each person's success dependent on the success of others. Eliminate monopolies, reduce resources, and create new networks of interaction.
- **Extend the shadow of the future**. Have people experience the consequences that result from their behavior and decisions. Tighten feedback loops. Shorten the duration of projects. Enable people to see how their success is aided by contributing to the success of others.
- **Reward those who cooperate**. Increase the payoff for all when they cooperate in a beneficial way. Establish penalties for those who fail to cooperate.

# Simple Rule 1: Understand What Your People Do

First rule states that you need to truly understand performance: **what people do** and **why they do it**. When you understand why people do what they do and how it drives performance, you can define the **minimum sufficient set of interventions with surgical accuracy**.

image by istock

## General Guidelines

The Six Simple Rules approach states that you can truly understand performance by:

- Tracing **performance back to behaviors** and how they influence and combine to produce overall results.
- Using **observation**, **mapping**, **measurement**, **and discussion** to do this.
- **Understand the context of goals**, **resources**, **and constraints** within which the current behaviors constitute rational strategies for people.

- Finding out how your **organization's elements** (structure, scorecards, systems, incentives, and so on) shape these goals, resources, and constraints.

## The Role of Architecture

I have found architecture function can be very helpful in understand what people really do in organizations in two ways:

- Using the **Data Pillar** to provide a complete **overview of various data sources** that can show where activities are happening, what trends are visible, and how people cooperate. One of the principles of the Data Pillar **build maps**, **not control units** supports understanding and orientation rather than being a simple metric tool.
- Leveraging the **People Pillar** to **connect people** and enables them to **learn what is happening** in different parts of the organization.

# Simple Rule 2: Reinforce Integrators

Reinforce integrators by looking at those **directly involved in the work**, giving them **power and interest to foster cooperation in dealing with complexity** instead of resorting to the paraphernalia of overarching hierarchies, overlays, dedicated interfaces, balanced scorecards, or coordination procedures.

image by robert owen_wahl from pixabay

## General Guidelines

You can reinforce integrators by:

- **Using feelings** to identify candidates: emotions provide essential clues for the analysis because they are symptoms rather than causes.
- Finding **operational units** those that can be **integrators among peer units** because of some particular interest or power.

- **Removing managerial layers who cannot add value** and reinforce others as integrators by eliminating some rules and relying on observation and judgment rather than metrics whenever cooperation is involved.

## The Role of Architecture

Architecture function, in my view, is strongly related to reinforcing integrators:

- My view on architects as **superglue** defines architects as **critical integrators** and **integrator role-models** in an organization.
- Via the **People Pillar**, Grounded Architecture can **help identify integrators** and connect them to leverage their work.
- The **Data Pillar** can **support integrators with data and insights**, empowering them to do better, more informed work.

# Simple Rule 3: Increase the Total Quantity of Power

Whenever you consider an **addition to** your organization's **structure, processes, and systems**, think about **increasing the quantity of power**. Doing so may **save you from increasing complicatedness** and enable you to achieve a more significant impact with less cost. You can increase the quantity of power by enabling some functions to have an influence on new stakes that matter to others and performance.

image by istock

## General Guidelines

To increase the quantity of power, the Six Simple Rules approach recommends the following actions:

- Whenever you are going to make a design decision that will **swing the pendulum—between center and units**, between

functions and line managers, and so on—see if making some parts of the organization **benefit from new power bases** could satisfy more requirements in dealing with complexity so that you don't have to swing the pendulum in the other direction in the future (which would only compound complicatedness with the mechanical frictions and disruptions inherent to these changes).

- When you have to create new functions, make sure you give them the power to play their role and that this **power does not come at the expense of the power needed by others** to play theirs.
- When you **create new tools** for managers (planning, or evaluation systems, for instance), ask yourself if these constitute **resources or constraints**. Providing a few tools simultaneously is more effective (because it creates a critical mass of power) than many tools sequentially, one after the other.
- **Regularly enrich power bases** to ensure agility, flexibility, and adaptiveness

## The Role of Architecture

Architecture supports increasing power quantity with the **operating model**[2] that promotes distributing decision-making:

- I aimed to increase the quantity of the **decision-making power** and keep architectural decision-making **distributed across the organization** and embedded in the development teams. Development teams traditionally have the best insights and most information relevant for making a decision.
- Additionally, the **Data Platform** accessible to all interested people in the organization, can give them **data in insights that can increase their power in daily work**.

[2] activities

# Simple Rule 4: Increase Reciprocity

In the face of business complexity, work is becoming more interdependent. To meet multiple and often contradictory performance requirements, **people need to rely more on each other**. They need to **cooperate directly** instead of relying on dedicated interfaces, coordination structures or procedures that only add to complicatedness.

image by istock / natnan srisuwan.

## General Guidelines

Reciprocity is the recognition by people or units in an organization that they **have a mutual interest in cooperation** and that the success of one depends on the success of others (and vice versa). The way to create that reciprocity is by setting rich objectives and reinforcing them by:

- **eliminating monopolies**,

- **reducing resources**, and
- **creating new networks of interaction**.

## The Role of Architecture

Architecture function is directly related to increasing reciprocity:

- I consider the success of architecture dependent on **architects' impact**. Likewise, the developer teams support depends on architecture support. I also integrate feedback of people from teams that architects support in architects' performance evaluations.
- In addition, the **People Pillar** directly supports one of the ways of reinforcing reciprocity: **creating new networks of interactions**.

# Simple Rule 5: Extend the Shadow of the Future

Six Simple Rules emphasize the importance of making visible and clear **what happens tomorrow as a consequence of what they do today**. You can manage complex requirements by making simple changes while removing organizational complexity. With the strategic alignment typical of the hard approach, these simple solutions—for instance, career paths—often come at the end of a sequence that starts by installing the most cumbersome changes: new structure, processes, systems, metrics, etc. Simple and effective solutions are then impossible.

image by joe from pixabay

## General Guidelines

The Six Simple Rules approach identifies four ways to extend the shadow of the future:

- Tighten the feedback loop by making **more frequent the**

**moments when people experience the consequence** of the fit between their contributions.
- **Bring the end point forward**, notably, by shortening the duration of projects.
- **Tie futures together** so that successful moves are conditioned by contributing to the successful move of others.
- Make people **walk in the shoes they make** for others.

## The Role of Architecture

The architecture can extend the shadow of the future in multiple ways:

- The **Data Pillar** creates **transparency** and provides data necessary to **model the future**. I've used such data to create many **simulations and roadmap** options.
- The principle of applying **economic modeling** to architecture decision-making directly supports describing what happens tomorrow as a consequence of what they do today.

# Simple Rule 6: Reward Those Who Cooperate

Lastly, the Six Simple Rules approach recommends that when you cannot create direct feedback loops embedded in people's tasks, you need **management's intervention to close the loop**. Managers must then use the familiar performance evaluation tool but in a very different way.

image by stocksnap from pixabay

## General Guidelines

To reward those who cooperate managers:

- Must **go beyond technical criteria** (putting the blame where the root cause problem originated). In dealing with the business complexity of multiple and often conflicting performance requirements, the smart organization accepts that

problems in execution happen for many reasons and that the only way to solve them is to **reduce the payoff for all those people or units that fail uo cooperate in solving a problem**, even if the problem does not take place exactly in their area, and to **increase the payoff for all when units cooperate in a beneficial way**.

- They must not blame failure, but **blame failing to help or ask for help**.
- Instead of the elusive sophistication of balance scorecards and other counterproductive cumbersome systems and procedures, they can **use simple questions** to change the terms of the managerial conversation so that **transparency and ambitious targets become resources rather than constraints** for the individual. Managers then act as integrators by obtaining from others the cooperation that will leverage the rich information allowed by this transparency and help achieve superior results.

## The Role of Architecture

**Architecture** can help rewarding cooperation by making it easier for everyone to **help other people and ask for help**.

- I try to organize the **People Pillar** so that it provides the context and **networks of people** to more easily collaborate.
- Various data sources in the **Data Pillar** can help creating **transparency about cooperation** opportunities and problems.

## To Probe Further

- Six Simple Rules: How to Manage Complexity without Getting Complicated[3], by Yves Morieux and Peter Tollman, 2014.

[3]https://www.bcg.com/capabilities/organization/smart-simplicity/six-rules-overcoming-complexity

## Questions to Consider

- *How can the concept of Smart Simplicity apply to your current role or position within your organization?*
- *In your organization, do you feel the structures, processes, and systems directly and predictably affect performance?*
- *Do you feel the human factor is viewed as the weakest link in your organization? How does this affect how you and your colleagues perform?*
- *How do you perceive the balance between your organization's hard and soft management approaches? Is one approach more dominant?*
- *How does your organization currently promote autonomy and cooperation among employees? Are there areas for improvement?*
- *How do the assumptions of hard and soft management approaches hinder cooperation in your organization?*
- *How can you increase the total power within your organization without taking power away from others?*
- *How can your organization increase reciprocity and make each person's success dependent on the success of others?*
- *How can your organization extend the shadow of the future? Are there feedback mechanisms in place to make people accountable for their decisions?*
- *How are those who cooperate rewarded in your organization? Are there mechanisms in place to increase the payoff for all when they cooperate beneficially?*
- *How can the architecture function in your organization support the implementation of the Six Simple Rules?*
- *How do your organization's current systems and structures promote or hinder the cooperation needed to make autonomy effective?*

# Understanding Product Development and The Build Trap

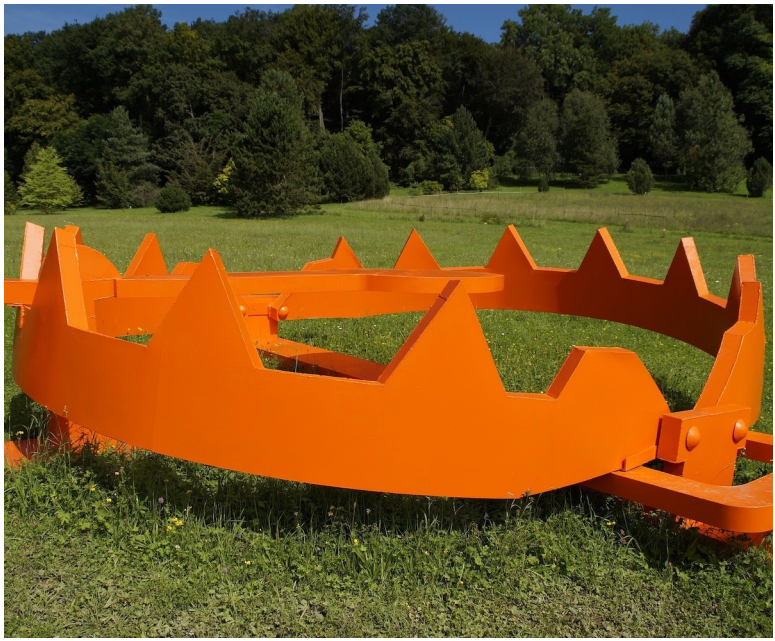


image by m w from pixabay

**IN THIS SECTION, YOU WILL**: Understand the importance of architecture in helping organizations become successful product-led organizations, focusing strongly on their customers' actual needs and preferences.

**KEY POINTS:**

- The product trap occurs when businesses focus too much on their product's features and functionalities, overlooking their customers' actual needs and preferences.
- There are three bad product-development organizations: sales, visionary, and technology led companies, .
- There are several bad product manager archetypes: Mini-CEO, Waiter, and Former Project Manager.
- Critical elements of successful product-led companies include: Understanding Customer Needs, Adopting a Customer-Centric Approach, Executing Iterative Product Development, Aligning Business Goals with Customer Needs, Embracing Innovation and Agility, and Measuring Success.

When it comes to product development, I generally recommend the book "Escaping the Build Trap: How Effective Product Management Creates Real Value[1]" by Melissa Perri. This book is a guide intended to help organizations shift their focus from simply building and shipping products to creating value for their customers. The "build trap" refers to the common pitfall where companies become fixated on building more features and products without considering whether they meet customer needs or generate desired outcomes.

Perri explores the reasons behind the build trap and provides practical strategies to help businesses escape it by adopting a value-centric, customer-focused approach. The main message is that by understanding customer needs, adopting a customer-centric approach, and embracing innovation and agility, companies can increase their chances of developing successful products that stand out in the market.

[1] https://www.goodreads.com/book/show/42611483-escaping-the-build-trap

In addition to numerous insights for architects, the book provides the following valuable tips necessary for architects' good interaction with product development:

- Recognizing bad product-development approaches
- Recognizing bad product manager archetypes
- Know how a good product development process looks like

## Bad Product Companies Archetypes

The product trap occurs when businesses focus too much on their product's features and functionalities, overlooking their customers' actual needs and preferences. Value, from a business perspective, is pretty straightforward. It can fuel your business: money, data, knowledge capital, or promotion. Every feature you build, and any initiative you take as a company should result in some outcome that is tied back to that business value.

Many companies are, instead, led by sales, visionaries, or technology. All of these ways of organizing can land you in the build trap.

1. **Sales-led companies** let their contracts define their product strategy. The product roadmap and direction were driven by what was promised to customers without aligning with the overall strategy.
2. **Visionary-led companies** can be very powerful — when you have the right visionary. But there aren't too many Steve Jobses floating around. Also, when that visionary leaves, the product direction usually crumbles. Operating as a visionary-led company is not sustainable.
3. **The technology-led companies** are driven by the latest and coolest technology. The problem is that they often lack a market-facing, value-led strategy.

Architects should be able to recognize and frequently challenge organizations with these archetypes.

## Bad Product Manager Archetypes

Architects will frequently need to collaborate closely with product managers. The fundamental role of the product manager in the organization is to work with a team to create the right product that balances meeting business needs with solving user problems. Product managers connect the dots. They take input from customer research, expert information, market research, business direction, experiment results, and data analysis.

To better understand the role of a product manager, it is useful to understand three bad product manager archetypes:

- **The Mini-CEO**: Product managers are not the mini-CEOs of a product, yet most job postings for product managers describe them as the mini-CEO. CEOs have sole authority over many things. Product managers can't change many things a CEO can do in an organization. They especially don't have authority over people because they are not people managers at the team level. Instead, they must influence them to move in a specific direction. Out of this wonderful CEO myth emerged an archetype of a very arrogant product manager who thinks they rule the world.
- **The Waiter**: The waiter is a product manager who, at heart, is an order taker. They go to their stakeholders, customers, or managers, ask for what they want, and turn those desires into a list of items to be developed. There is no goal, vision, or decision-making involved. More often than not, the most important person gets their features prioritized. Instead of discovering problems, waiters ask, "What do you want?" The customer asks for a specific solution, which these product managers implement. The waiter approach leads to what David J. Bland is calling the Product Death Cycle[2]:

[2] https://twitter.com/davidjbland/status/467096015318036480

- **The Former Project Manager**: Product managers are not project managers, although some project management is needed to execute the role correctly. Project managers are responsible for the when. When will a project finish? Is everyone on track? Will we hit our deadline? Product managers are responsible for the why? Why are we building this? How does it deliver value to our customers? How does it help meet the goals of the business? The latter questions are more challenging to answer than the former, and product managers who don't understand their roles often resort to doing that type of work. Many companies still think the project manager and product manager are the same.

## How a Good Product Development Approach Looks Like

Product-led companies understand that the success of their products is the primary driver of growth and value for their company. They prioritize, organize, and strategize around product success.

Critical elements of successful product-led companies include:

2. **Understanding Customer Needs**: Successful companies understand customer needs, desires, and expectations to develop successful products. Businesses can gain valuable insights and tailor their products by conducting thorough market research and engaging with customers.
3. **Adopting a Customer-Centric Approach**: Organizations need to adopt a customer-centric approach to product development to avoid the product trap. This approach means prioritizing customer satisfaction and incorporating their feedback throughout the entire product development process.
4. **Executing Iterative Product Development**: Iterative product development is essential, continuously testing and refining the product based on customer feedback. An iterative process helps businesses identify and address potential issues before they become significant problems.
5. **Aligning Business Goals with Customer Needs**: Businesses should align their goals and objectives with the needs of their customers. Doing so can ensure that their products deliver value and create a robust and loyal customer base.
6. **Embracing Innovation and Agility**: Businesses must be innovative and agile to adapt to rapidly changing customer preferences and market conditions. This adaptivity includes staying informed about the latest trends, technologies, and best practices in product development.

7. **Measuring Success**: Accurately measuring a product's success is essential. Such measuring involves tracking key performance indicators (KPIs) and using data-driven insights to make informed product improvements and enhancements decisions.

## Questions to Consider

- *Have you ever found yourself or your organization falling into the "build trap"? What were the signs, and how did you (or didn't you) address them?*
- *Reflecting on your organization, would you say it's sales-led, visionary-led, or technology-led?*
- *Can you identify instances where a product manager, have acted like a "Mini-CEO," "Waiter," or a "Former Project Manager"? What were the consequences?*
- *How does your organization currently understand and incorporate customer needs? Could there be improvements in this area?*
- *How does your company approach iterative product development? Is it a formalized process, or does it need more structure?*
- *Are your business goals aligned with customer needs? How do you maintain this alignment as business goals and customer needs evolve?*
- *How innovative and agile do you consider your organization to be? What areas need more flexibility or creativity?*
- *What metrics does your organization use to measure product success? Are these effectively capturing the value your products create for customers?*

# Architecture Governance: Mandates, Taxation, Nudge

image by nonbirinonko from pixabay

**IN THIS SECTION, YOU WILL**: Understand that a technology governance model should be a well-balanced hybrid of three different styles of governing: mandates and bans, taxes, and nudging.

**KEY POINTS:**

- Grounded Architecture supports governance models adaptable to organizations' complex and diverse needs. I see a technology governance model as a well-balanced hybrid of three different styles of governing: mandates and bans, taxes, and nudging.
- By governing with mandates and bans, I mean guiding people by explicitly defining what they should or should not do.
- Governing with taxes is a form of guiding in which people are not forbidden to make some decisions but need to "pay" some form of taxes on used resources.
- Nudging is a form of governing where we create subtle or indirect suggestion influencing someone's behavior or decision-making without forcing them or limiting their freedom of choice.

Grounded Architecture supports governance models adaptable to organizations' complex and diverse needs. By governing, I mean guiding technology choices in the organization in a particular direction aligned with the technology strategy. I see a technology governance model as a well-balanced hybrid of three different styles of governing:

- mandates and bans,
- taxes, and
- nudging.

Grounded Architecture supports governance models that are adaptable to the complex and diverse needs of organizations.

## Mandates and Bans

By governing with mandates and bans, I mean guiding people by **explicitly defining what they should or should not do**. In places I worked in, such mandates and bans have had a limited by important place to **define broader strategic boundaries of choices** people can make. For instance, restricting the usage of public cloud providers to specific vendors or following **strict privacy and security procedures** needs to be explicitly defined and controlled.


image by tumisu from pixabay

The role of architecture in this form of governing should be to be **a stakeholder but not a sole owner in defining mandates and bans**. Mandates and bans frequently need to be defined in collaboration with others, such as security and legal functions. The Grounded Architecture can help by **creating clarity and providing transparency**.

The Data Pillar is crucial in creating clarity and transparency, for

instance, via insights security reports or maps of areas in source code or infrastructure that needed monitoring and controlling based on privacy or security requirements.

The People Pillar can help propagate the decision and ensure its positive impact and acceptance. You should never order or forbid people to do some things routinely. Spending enough time with all stakeholders to explain the reasons and motivations behind introducing some limitations is crucial to ensure the effective working of mandates and bans. A strong People pillar ensures that you already have strong connections with key stakeholders and can leverage them to change ways of working more smoothly.

## Taxation

Governing with taxes is a form of guiding in which people are not forbidden to make some decisions but **need to "pay" some form of taxes on used resources**. For instance, costs of public cloud usage could be **cross-charged across the organization**, providing a helpful feedback loop to optimize our systems and avoid unnecessary resource consumption, avoiding the "tragedy of commons" which frequently happens when there are no limits on shared resource consumption (e.g., public cloud budget).

image by steve buissinne from pixabay

The role of Grounded Architecture in this form of governance should be to ensure that "taxation" is data-driven and transparent and to create efficient feedback loops on key metrics related to "taxes."

The Data Pillar can include and provide all data regarding "taxes," for instance, via insights based on public cloud cost reports. The

People Pillar can facilitate aligning processes, goals, and working methods to ensure that taxation leads to desired and meaningful change.

# Nudging

In behavioral economics and psychology, a nudge is a **subtle or indirect suggestion** influencing someone's behavior or decision-making **without forcing them or limiting their freedom of choice**. Nudges can be applied in various settings, such as policy-making, marketing, and personal interactions, to encourage people to make better choices, improve their well-being, or achieve specific goals.

A nudge can take many forms, such as a **small change in the environment**, a **gentle reminder**, a **positive reinforcement**, or a **default option**. For example, placing healthy food options at eye level in a cafeteria can nudge people to choose healthier meals. Or setting a default option for organ donation can increase the number of donors.

The concept of a nudge was popularized by the book "Nudge: Improving Decisions About Health, Wealth, and Happiness" by Richard Thaler and Cass Sunstein, which argues that various cognitive biases and heuristics often influence people's decisions, and that nudges can help people overcome these biases and make better choices.

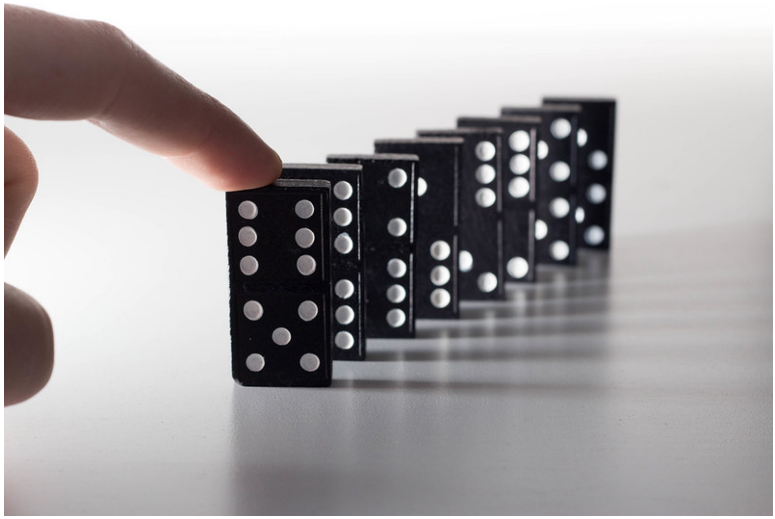


image by hand ready to push domino pieces by marco verch under creative commons 2.0

Richard Thaler and Cass Sunstein also introduced the concept of **choice architecture** as a key component of nudging. It refers to how the options are presented to individuals, which can significantly influence their choices. Choice architecture is the design of the decision-making environment, which includes the layout, structure, and organization of the options available.

In architecture, nudges could include:

- architectural **principles** as informal decision guidelines,
- recommendations for **best practices** to stimulate introduction and alignment around such practices,
- default options for technology choices via **golden paths**[1]
- **highlighting** bad quality software on a Data Pillar dashboards to create a subtle pressure for people to improve it,
- tracking of **tech debt**, to create awareness about it size and lead action to reduce it,

[1] https://engineering.atspotify.com/2020/08/how-we-use-golden-paths-to-solve-fragmentation-in-our-software-ecosystem/

- **visualizing cost trends** of cloud services per team to stimulate teams to improve the performance efficiency of their software.

In my experience, nudges can frequently lead to better alignment and more harmonization without the negative consequences of mandates, bans, or taxation.

Grounded Architecture is well aligned with ideas of nudging. I frequently designed many of the Data Pillar to highlight areas and issues we wanted people to improve. And the People Pillar can create mechanisms for sharing experiences, promoting positive examples, and capturing lessons learned to help people to make better, more informed decisions. And in the [Architecture Activities Platform] I use the operating model that stimulates people to make decisions autonomously but nudges them to stay well-aligned and connected to the organizational strategic direction.

## Questions to Consider

- *What are the key components of the governance model in your organization, and how do mandates, taxes, and nudging influence them?*
- *How does your organization currently handle mandates and bans? Are they explicit and aligned with the overall technology strategy?*
- *How effective is the enforcement of these mandates and bans in your organization? Could improvements be made in creating clarity and providing transparency?*
- *How does your organization approach taxation as a form of governance? Is it transparent, data-driven, and efficient?*
- *Can you identify any examples of 'nudging' in your current architectural environment? How effective are these subtle suggestions in influencing behavior or decision-making?*
- *How does your organization promote best practices and align around them? Are there any 'golden paths' for technology choices?*
- *How are your organization's tech debt and the cost trends of cloud services tracked and visualized? Do these methods create enough awareness to stimulate improvement?*
- *How could you better utilize nudging to improve organizational decision-making? What biases or barriers to effective decision-making could be targeted with this approach?*

# Economic Modeling: ROI and Financial Options

image by nattanan kanchanaprat from pixabay

**IN THIS SECTION, YOU WILL:** Get two answers to the question of the economic value of architecture: the return on investment metaphor and the selling options metaphor.

**KEY POINTS:**

- Architects are frequently asked about the (economic) value of architecture or technology investments.
- Answering this question is a crucial skill for any senior architect. But it may be difficult to answer this seemingly harmless question concisely and convincingly to a non-technical audience.
- Borrowing from existing literature, I sketch two answers to the question of the economic value of architecture: the return on investment metaphor and the selling options metaphor.

Architects frequently need to answer questions about (economic) value of technology investments and architecture. Answering this question is a crucial skill for any senior architect. But it may be difficult to answer this seemingly harmless question concisely and convincingly to a non-technical audience.

## Why Do We Need to Explain the Economics?

Having good architecture requires some investment. This investment is time and effort spent implementing some architecture pattern, reducing technical debt, or refactoring code to align with our architecture. Consequently, we need to explain the expected value of this investment.

Martin Fowler at #oop2014: "If you use the arguments on the left to justify refactoring, you're screwed." pic.twitter.com/b9ffsudckr— Matthias Bohlen (@mbohlende) February 6, 2014

In this post, I sketch two answers to the question of the economic value of architecture:

- the return on investment metaphor
- the financial options metaphor


**Figure 1**: *Two metaphors for explaining the economic value of architecture: return on investment (ROI) and financial options.*

## The Return-on-Investment Metaphor

In economic terms, return on investment (ROI) is a ratio between profits and costs over some period. In other words, ROI shows how much you get back from your investment. A high ROI means the investment's gains compare favorably to its cost. As a performance measure, ROI is used to evaluate an investment's efficiency or compare the efficiencies of several different investments.


**Figure 2:** *An illustration of the ROI metaphor. Investment leads to lower costs or higher value. It takes some time to reach a break-even point, a point when additional value has compensated for the investment. After the break-even point, we start to earn more profit than without the investment.*

An investment in good architecture can help increase ROI of the IT. An excellent example of using the ROI metaphor to argue for investing in architecture is the port of Martin Fowler, who uses this argument to argue for the importance of investing in improving internal quality[1]. Figure 3 summarizes his argument.

Well-architect systems are typically much easier to understand and change. As our systems continuously evolve, the return on investing in making a system easier to understand and change can be significant. The primary value of such investment comes from generating fewer errors and bugs, more straightforward modifications, short time-to-market, and improved developer satisfaction.

[1] https://martinfowler.com/articles/is-quality-worth-cost.html

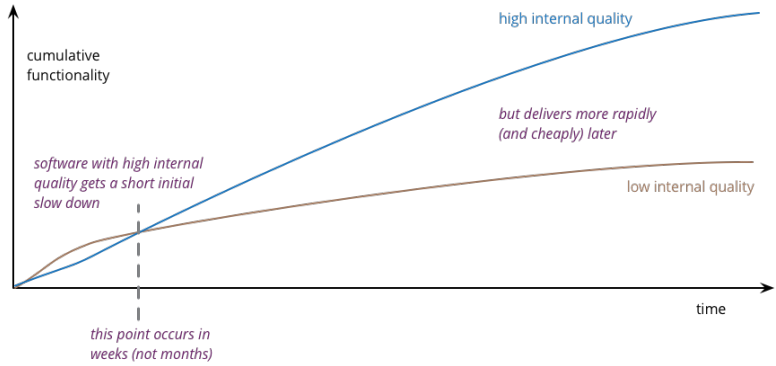



**Figure 3**: *Software with high internal quality gets a short initial slow down, but deliveres more rabidally and cheply later (source martinfowler.com/articles/is-quality-worth-cost.html).*

An ROI metaphor is easy to understand by a non-technical audience, but it has its limitations to describe the value of architecture. The first limitation lies in that it is challenging to measure architecture, quality, and productivity. Consequently, too much focus on ROI can lead to an obsession with cost-cutting. Costs are easy to measure, but the value of attributes like shorter time-to-market is much more difficult to quantify. Second, ROI is a good measure, but not every investment in architecture will increase profit. That is because we frequently have to make decisions with lots of uncertainty. Nevertheless, that does not mean that we should not make such investments. The following section explains why.

## The Financial Options Metaphor

Gregor Hohpe has frequently argued that the best way to explain architecture to non-technical people is by using a financial option metaphor. A financial option is a right, but not an obligation, to buy or sell financial instruments at a future point in time with some predefined price. As such, a financial option is a way to defer a decision: instead of deciding to buy or sell a stock today, you have the right to make that decision in the future at a known price.

Options are not free, and there is a complex market for buying and selling financial options. Fischer Black and Myron Scholes managed to compute the value of an option with the Black-Scholes Formula[2]. A critical parameter in establishing the option's value is the price at which you can purchase the stock in the future, the so-called strike price. The lower this strike price, the higher the value of the option.

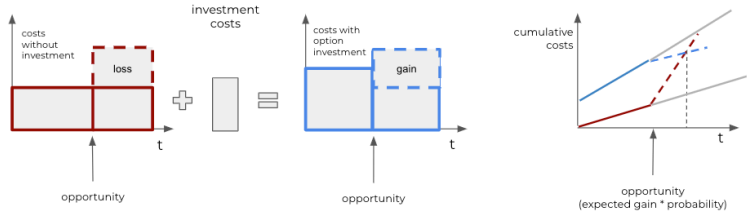



**Figure 4**: *An illustration of the financial option metaphor. Options have a price, leading to higher initial costs. However, if an opportunity can generate more value, we gain additional profit (or lose it if we do not invest).*

Applying the financial option metaphor to IT architecture, we can argue that buying options gives the business and IT a way to defer decisions. Gregor Hohpe gives an example of the server's size that you need to purchase for a system. If your application is architected to be horizontally scalable, you can defer this decision: additional (virtual) servers can be ordered later at a known unit cost.

[2] https://en.wikipedia.org/wiki/Black%E2%80%93Scholes_model

Another example of an IT option is architecting your system to clearly separate concerns. For instance, it may be challenging to decide early what authentication mechanism an application should use? A system that properly separates concerns allows changes to be localized so that updating one aspect of a system does not require expensive changing the whole system. Such isolation will enable you to change a decision late in the project or even after go-live, at a nominal cost. For example, if authentication is a well-isolated concern, you will need to refactor only a minimal part of the system to use another authentication system.

The option's value originates from being able to defer the decision until you have more information while fixing the price. In times of uncertainty, the value of the options that architecture sells only increases.

As with any analogy, the financial options analogy has its limits. Again, it isn't easy to quantify architecture values and have metrics for the value of separation of concerns or horizontal scaling. Second, while the metaphor may be easy to grasp for an economic audience, it may require explaining to other stakeholders, who may be less familiar with financial options markets.

## To Probe Further

- Martin Fowler (2019): Is High Quality Software Worth the Cost?[3].
- Martin Fowler (2011): Tradable Quality Hypothesis[4].
- Gregor Hohpe (2016): Architecture: Selling Options[5].

[3] https://martinfowler.com/articles/is-quality-worth-cost.html
[4] https://martinfowler.com/bliki/TradableQualityHypothesis.html
[5] https://architectelevator.com/architecture/architecture-options/

## Questions to Consider

- *How can you effectively communicate the value of architectural investments to non-technical stakeholders in your organization?*
- *How do you weigh the importance of short-term cost reductions against long-term architecture improvements?*
- *How could the return-on-investment metaphor be useful in explaining the benefits of architecture investment to your team or organization?*
- *If you were to use the ROI metaphor to explain architecture's value to non-technical stakeholders, what examples or case studies would you use to illustrate your points?*
- *What are some potential pitfalls of relying too heavily on the ROI metaphor when deciding on architecture investments?*
- *How could you use the financial options metaphor to explain the value of architectural investments? What are the benefits and challenges of using this metaphor in your organization?*
- *How can you better quantify the value of architectural investments, particularly in terms of attributes like time-to-market and developer satisfaction?*
- *How might the financial options metaphor apply to recent decisions facing your organization or team, and how could it influence those decisions?*
- *What do you think about Gregor Hohpe's argument about the best way to explain architecture to non-technical people? How could this approach be applied in your work?*

# Wrapping Up

# Summary


image by gerd altmann from pixabay

**IN THIS SECTION, YOU WILL:** Look back at the content in this book.

**KEY POINTS:**

- This playbook aims to share my approach to running an IT architecture practice in larger organizations based on my experience as Chief Architect at AVIV Group, eBay Classifieds, and Adevinta.
- I called this approach "Grounded Architecture," highlighting the need for any IT architecture function to stay well-connected to all levels of an organization and led by data.
- In this section, I summarize the key points from the playbook.

In this playbook, I explained an approach to running an architecture practice in larger organizations. This playbook aims to share my approach to running an IT architecture practice in larger organizations based on my experience as Chief Architect at AVIV Group, eBay Classifieds, and Adevinta. I called this approach "grounded architecture," highlighting the need for any architecture function to stay well-connected to all levels of an organization and led by data.

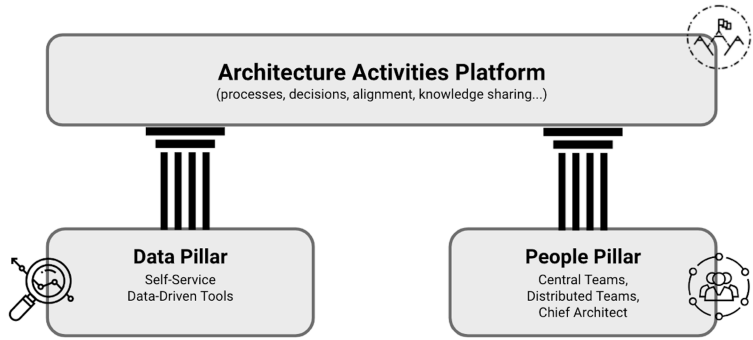
Grounded Architecture has two main parts:

- **Structure**, defining elements you need to have to run Grounded Architecture practice, and
- **Reflections**, a set of my considerations and guiding principles that can help put the ideas of Grounded Architecture into practice.

The structure of the Grounded Architecture consisting of three elements:

- The Data Pillar,

- The People Pillar,
- The Architecture Activities Platform.




The *Data Pillar* ensures that architects can make data-informed decisions based on a real-time and complete overview of the organization's technology landscape.

The *People Pillar* is another essential element of Grounded Architecture. A strong network of people doing architecture across the organization is crucial to ensure that architecture function has any tangible impact.

Lastly, the *Architecture Activities Platform* defines a set of processes and agreements enabling architects to do everything architecture typically does, leveraging data and people pillars to create a data-informed, organization-wide impact.

As a part of my work on Grounded Architecture, I also provided key lessons I learned while developing ideas of Grounded Architecture in practice, grouped into two parts:

- Being an Architect:
  - Architects as Superglue
  - Skills
  - Impact

  - Leadership
  - Architects' Career Paths

- Doing Architecture:

  - Culture Map: Architects' Culture Mindfield Compass
  - Managing Organization Complexity: Six Simple Rules
  - Product Development and The Build Trap
  - Architecture Governance: Mandates, Taxation, Nudge
  - Economic Modeling: ROI and Financial Options

When Grounded Architecture is in place, it can have a significant positive impact on the functioning of an organization:

- Enable Execution of Architecture Function At Scale,
- Increase the Quality of Decision-Making with Data,
- Maximize Organizational Alignment,
- Maximize Organizational Learning, and
- Increase Architecture Function Adaptivity.

Remember that while you may borrow ideas from others, every organization is different in the end, and your approach must adapt to your context. When forming architecture functions, I always advise not to forget these two pieces of advice from Gregor Hohpe[1]:

- "*Your architecture team's job is to solve your biggest problems. The best setup is the one that allows it to accomplish that.*"
- "*Your organization has to earn its way to an effective architecture function. You can't just plug some architects into the current mess and expect it to solve all your problems.*"

[1] https://architectelevator.com/architecture/organizing-architecture/

# Cheat Sheet

# Introduction

This simple cheatsheet serves as a reference guide for topics discussed in follow-up sections.

## Context: Fast Moving Global Organizations

- To better understand any idea or solution, it is crucial to understand the context in which these ideas developed.
- The grounded architecture approach has developed in the context of global, loosely coupled organizations that are diverse, with non-linear growth dynamics, and under transformation pressures.
- More specifically, the work I present here reflects my experiences as a Chief or Principal Architect at AVIV Group, eBay Classifieds, and Adevinta.

## Goals

- I identified the following needs that an architecture function should support: Executing At Scale, Increasing the Quality of Decision-Making with Data, Maximizing Organizational Alignment & Learning, Higher Adaptivity.

# Grounded Architecture Structure

## Data Pillar

- The architecture data pillar serves as a medium to create a complete, up-to-date picture of critical elements of technology landscapes of big organizations.
- The data pillar provides an architecture-centric view on data about a technology landscape based on source code analyses, public cloud billing reports, vibrancy reports, or incident tickets.
- To facilitate creation of data pillars, I have been working on development open-source tools, such as Sokrates[1], that can help obtain valuable architectural insights from data sources, such as source code repositories.

## People Pillar

- Developing the architecture function requires having competent, empowered, and motivated architects. Any architecture function must carefully organize, empower, and leverage scarce architecture talent.
- We should not take architectural talent for granted. Architects are bridging local business, product, organizational, and technology issues. Architects are difficult to hire talent as they need not only in-depth technical knowledge but also domain-specific and organizational knowledge.
- In my work in the past few years, I was working by combining, in different forms, two teams of architects: a small central architecture team and a cross-organizational distributed virtual team.

[1] https://sokrates.dev

## Activities Platform

- The Architecture Activities Platform defines a set of processes and agreements enabling architects to do everything architecture typically does, leveraging data and people pillars to create a data-informed, organization-wide impact.
- Examples of activities include: supporting teams in their daily work; tracking tech debt, defining tech debt reduction programs; performing technical due diligence; standardization of processes and documentation; defining cloud, data, and platform strategies.

# Being Architect

## Architects as Superglue

- Architects in IT organizations should develop as "superglue," people who hold architecture, technical details, business needs, and people together across a large organization or complex projects.
- Architects need to be technically strong. But their unique strengths should stem from being able to relate, or glue, technical issues with business and broader issues.
- Superglue architects should stand on three legs: Skills, Impact, Leadership.

## Skills

- A typical skillset of an architects includes: hard (technical) skills, soft (people & social) skills, and business skills.

## Impact

- Architects' work is evaluated based the impact they've had on the organization.
- Architects can make an impact via three pillars: Big-Picture Thinking, Execution, and Leveling-Up.

## Leadership

- My view of architecture leadership is inspired by David Marquet's work and the Netflix's valued behaviors.
- Marquet focused on leadership and organizational management, with a particular emphasis on the principles of Intent-Based Leadership.

- Borrowing from Netflix's original values the following behavioral traits are crucial for architects are: communication, judgment, impact, inclusion, selflessness, courage, integrity, curiosity, innovation, and passion.

## Architects' Career Paths

- Architects' career paths ideally stem from a strong engineering background.
- I usually envision three tracks of seniority progression: Senior Architect, Principal Architect, and Enterprise Architect.
- Hiring architects requires constantly raising the bar to ensure a strong and diverse team structure.

# Doing Architecture

## The Culture Map: Architects' Culture Mindfield Compass

- I have found the work of Erin Meyer, The Culture Map, to be a very helpful tool for architects to work harmoniously with people from a broad array of different cultures and backgrounds.
- Meyer's model contains eight scales, each representing a key area, showing how cultures vary from extreme to extreme: Communicating, Evaluating, Persuading, Leading, Deciding, Trusting, Disagreeing, and Scheduling.

## Managing Organizational Complexity: Six Simple Rules

- Six Simple Rules emphasis that in today's complicated business environment, you need to setup organisational structures based on cooperation.
- To deal with complexity, organizations should depend on the judgment of their people and on these people cooperating to utilize the organization's capabilities to cope with complex problems.
- This view is well aligned with the ideas of Grounded Architecture.

## In the Eye of the Storm: Aligning Business, Product, Technology and Organization Strategies

- I want architecture always to address one of the most challenging problems our organization faces: a tension between

technology, product, organization, and business functions.
- Architecture should ensure that conversations happen between the technical, product, organizational, and business functions.

## Product Development and The Build Trap

- The product trap occurs when businesses focus too much on their product's features and functionalities, overlooking their customers' actual needs and preferences.
- There are three bad product-development organizations: sales, visionary, and technology led companies, .
- There are several bad product manager archetypes: Mini-CEO, Waiter, and Former Project Manager.
- Critical elements of successful product-led companies include: Understanding Customer Needs, Adopting a Customer-Centric Approach, Executing Iterative Product Development, Aligning Business Goals with Customer Needs, Embracing Innovation and Agility, and Measuring Success.

## Architecture Governance: Mandates, Taxation, Nudge

- Grounded Architecture supports governance models adaptable to organizations' complex and diverse needs. I see a technology governance model as a well-balanced hybrid of three different styles of governing: mandates and bans, taxes, and nudging.
- By governing with mandates and bans, I mean guiding people by explicitly defining what they should or should not do.
- Governing with taxes is a form of guiding in which people are not forbidden to make some decisions but need to "pay" some form of taxes on used resources.

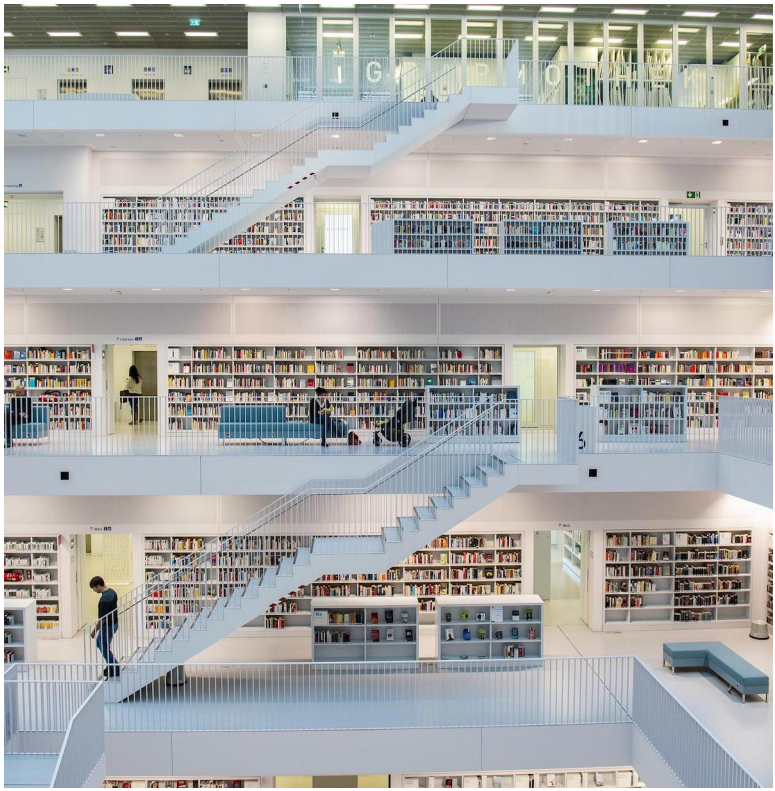
- Nudging is a form of governing where we create subtle or indirect suggestion influencing someone's behavior or decision-making without forcing them or limiting their freedom of choice.

## Economic Modeling: ROI and Financial Options

- Architects are frequently asked about the (economic) value of architecture or technology investments.
- Answering this question is a crucial skill for any senior architect. But it may be difficult to answer this seemingly harmless question concisely and convincingly to a non-technical audience.
- Borrowing from existing literature, I sketch two answers to the question of the economic value of architecture: the return on investment metaphor and the selling options metaphor.

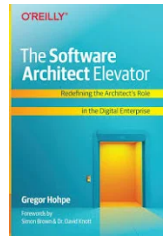
# To Probe Further

# Bookshelf



**IN THIS SECTION, YOU WILL:** Get an overview of the background work for you to probe further, linking several external resources inspiring my work.

# Introduction




The Software Architect Elevator: Redefining the Architect's Role by Gregor Hohpe defines architects as people that can fill a void in large enterprises: they work and communicate closely with technical staff on projects but are also able to convey technical topics to upper management without losing the essence of the message. Conversely, they understand the company's business strategy and can translate it into technical decisions that support it.

---

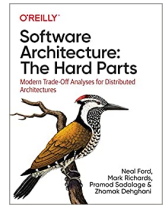
SE Radio: On Architecture[1]: for those interested in IT Architecture, I've created a curated Spotify playlist of Software Engineering Radio Episodes focusing on Architecture.

---

[1] https://open.spotify.com/playlist/7GVD86edcILnVPjsZFTy28?si=f44d0bef360e4818


Software Architecture for Developers by Simon Brown is a practical, pragmatic and lightweight guide to software architecture, specifically aimed at software developers.

---




Software Architecture: The Hard Parts is structured as a narrative about a team breaking down a faulty outdated monolithic application into a modern microservices-based architecture. The authors compare different aspects of how a monolithic architecture might have been written to do something in the past, then how a modern microservice architecture could do the same thing today, offering advice and approaches for effective tradeoff analysis when refactoring a large monolith app.

---


Just Enough Software Architecture by George Fairbanks is an approachable and comprehensive book.

---


Software Systems Architecture by Nick and Eóin is a practitioner-oriented guide to designing and implementing effective architectures for information systems.

# Career Development


The Staff Engineer's Path by Tanya Reilly. Similar to my view of architects the staff engineer's path allows engineers to contribute at a high level as role models, driving big projects, determining technical strategy, and raising everyone's skills.

---


Staff Engineer by Will Larson, defined technical leadership beyond the management track. I share many of the views presented in this book regarding development and skills of architects.

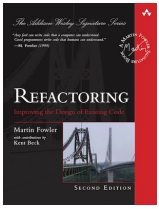
# Hard Skills

This section is heavily influenced by Gregor Hohpe's article The Architect's Path - Part 2 - Bookshelf[2]

---


Clean Code by Bob Martin. Good software starts with good code, and good code is clean. The basics of naming, functions that do one thing well, and formatting.

---



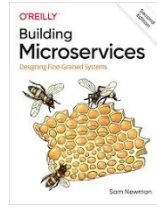

Refactoring: Improving the Design of Existing Code by Martin Fowler. Good software evolves, gains entropy, and is then re-structured. Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

[2] https://architectelevator.com/architecture/architect-bookshelf/

---

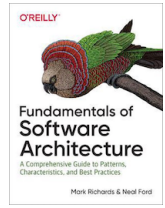

Design Patterns by Gamma, Helm, Johnson, Vlissides. Design patterns help us make balanced decisions on the design of our code.

---

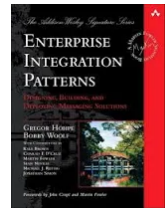



Building Microservices (2nd Edition) by Sam Newman is a book about architectural trade-offs and considerations in distributed system design. Very accessible.

---




Fundamentals of Software Architecture by Neal and Mark, covers soft skills, modularity, component-based thinking, and architectural styles.

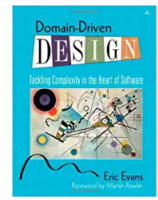



Enterprise Integration Patterns for anyone trying to connect systems without coupling them too tightly.


Pattern-Oriented Software Architecture: one of the most comprehensive references for distributed system design.


Building Evolutionary Architectures describes how to use fitness functions to guide architectural change over time. I thought the authors could have gotten a bit more our of this topic - perhaps we'll see a second edition.

---




Domain-Driven Design by Eric Evans promotes the idea that that to develop software for a complex domain, we need to build Ubiquitous Language that embeds domain terminology into the software systems that we build. DDD stresses defining models in software, and evolving them during the life of the software product.

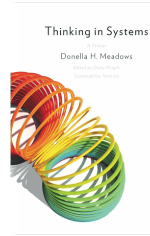
---


Release It! (2nd Edition) by Mike Nygard's is about architecture design and decisions regarding stability and how to build "cynical" software. Besides the well known stability patterns like circuit breakers and bulkheads the book introduces foundational knowledge about running systems in production, from processes to network to security, also covering more process oriented questions like deployments or handling security.

---


Designing Data Intensive Systems by Martin Kleppmann is a mini-encyclopedia of modern data engineering. Not a practice or a cookbook for a particular Big Data, NoSQL or newSQL product, the book lays down the principles of current distributed big data systems. Covers replication, partitioning, linearizability, locking, write skew, phantoms, transactions, event logs, and much more.

---




Thinking in Systems by Donella Meadows describes a system is more than the sum of its parts. It may exhibit adaptive, dynamic, goal-seeking, self-preserving, and sometimes evolutionary behavior.

---


Don't make me think, revisited: a common sense approach to Web usability by Steve Krug provides a practical guide for understanding web usability and user experience.

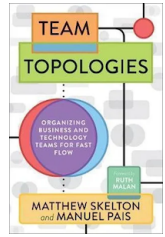
---


System Design Interview (Volume 1 & 2) by Alex Xu. A "real-world" systems design books, not just for preparing for the systems design interview, but to strengthen your systems design muscle for the day-to-day architectural work.

---

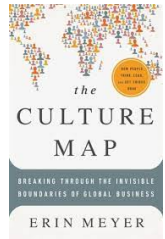
# Soft Skills

---



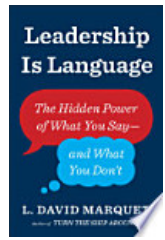

Team Topologies by Manuel Pais and Matthew Skelton describes four fundamental topologies (stream-aligned teams, enabling teams, complicated subsystem teams, and platform teams), and three fundamental interaction modes (collaboration, X-as-a-Service, and facilitation).

---

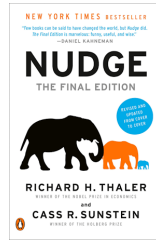



The Culture Map: Decoding How People Think, Lead, and Get Things Done Across Cultures: provides a framework for handling intercultural differences in business and illustrates how different cultures perceive the world. Awareness of intercultural differences is crucial for the success of an architect in an international setting. It helps us understand these differences and, in doing so, improves our ability to react to specific behaviors that might have once seemed strange.


Turn the Ship Around! by L. David Marquet: Around reveals the story of how one United States Navy captain managed to turn a dissatisfied submarine crew into a formidable and respected team. But how did he do it? By changing the way we think about leadership, this story will show you that inside, we all have the power to be leaders.




Leadership is Language by L. David Marquet is a playbook for successful team management. Written by a former US Navy captain, it teaches leaders how they can change their language and mindsets in order to improve decision-making, empower workers, and achieve better results..

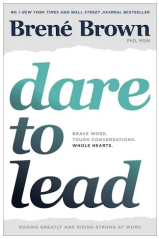



Nudge by Richard Thaler draws on research in psychology and behavioral economics to define libertarian paternalism as an active engineering of choice architecture. The book also popularised the concept of a nudge, a choice architecture that predictably alters people's behavior without restricting options or significantly changing their economic incentives.

---

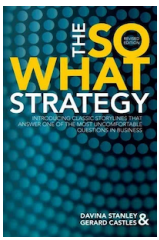

No Rules Rules by Erin Meyer and Reed Hastings provides insights into Netflix culture. When you give low-level employees access to information that is generally reserved for high-level executives, they get more done on their own. They work faster without stopping to ask for information and approval. They make better decisions without needing input from the top.

---



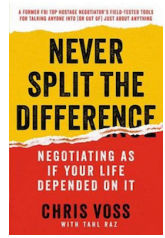

Dare To Lead by Brené Brown, defines a leader as anyone who takes responsibility for finding the potential in people and processes and has the courage to develop that potential.

---



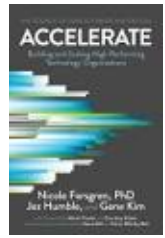

The So What Strategy by Davina Stanley and Gerard Castles. Any architect has been caught at the end of a presentation when their audience, perhaps a leadership team or a Steering Committee, looks at them blankly and asks this most uncomfortable question: 'So what?' How does that help? The book provides a pragmatic approach to answer that question in one single, powerful sentence. Or how to set yourself up so nobody asks it.

---




Never Split the Difference by Chris Voss. Negotiations take place in many fields of life, such as business, and in some critical situations, like hostage situations (and architecture). The book is a guide on how to best behave when certain things happen, regardless of whether that includes the need for negotiation techniques in hostage situations or in business.

---

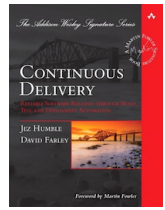



Accelerate—acceleration is the second derivative of position (speed being the first), so if you want to move faster you need to accelerate. Sometimes, you need to jerk the system a bit, which in fact the proper term for the third derivative.

---

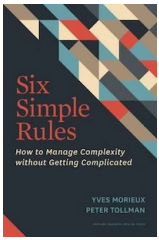

The Phoenix Project and The Unicorn Project by Gene Kim The Phoenix Project is a best-selling novel about DevOps. The book's characters reveal through their actions why it's so important for organizations to put security first and tear down the silos that have traditionally existed between development and operations teams.

---



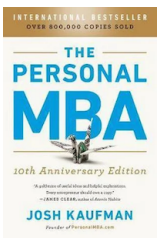

Continuous Delivery— through automation of the build, deployment, and testing process, and improved collaboration between developers, testers, and operations, delivery teams can get changes released in a matter of hours-sometimes even minutes-no matter what the size of a project or the complexity of its code base.

# Business, Product, Strategy



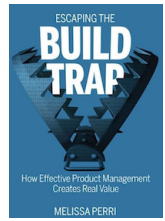

Six Simple Rules: How to Manage Complexity without Getting Complicated by Yves Morieux and Peter Tollman: the book emphasis that in today's complicated business environment, you need to setup organisational structures based on cooperation. To deal with complexity, organizations should depend on the judgment of their people, which requires giving them more autonomy to act. It also depends on these people cooperating to utilize the organization's capabilities to cope with complex problems.

---




The Personal MBA 10th Anniversary Edition by Josh Kaufman provides an overview of the essentials of every major business topic: entrepreneurship, product development, marketing, sales, negotiation, accounting, finance, productivity, communication, psychology, leadership, systems design, analysis, and operations management.


Technology Strategy Patterns by Eben Hewitt provides a shared language—in the form of repeatable, practical patterns and templates—to produce great technology strategies.




Escaping the Build Trap by Melissa Perri is a practical guide to product management, product strategy, and ensuring product success.

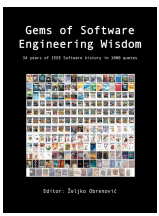
# Tools

**Sokrates**[1]: Sokrates is a tool I built to implement my vision of documenting and analyzing software architectures of complex systems. Sokrates provides a pragmatic, inexpensive way to extract rich data from source code repositories. Sokrates can help you understand your code by making visible the size, complexity, and coupling of software and all people interactions and team topologies.

---

**Productivity Tools**[2]: a collection of online tools I built to help me in my daily work as an architect. I reuse these tools and lessons learned in building these tools when designing data pillar parts of the grounded architecture.

---




[1] https://sokrates.dev
[2] https://obren.io/tools

**obren359.com**[3]: I've created a curated collections and high-quality IT resources (articles, videos, podcasts).

[3] https://obren359.com

# Favorite Quotes


---

*Architecture is **not so much about the software**, **but about the people** who write the software. The core principles of architecture, such as **coupling and cohesion**, aren't about the code. The code doesn't 'care' about how cohesive or decoupled it is; … But people do care about their coupling to other team members.* -James O. Coplien

---

*Software design is **an exercise in human relationships**.* -Kent Beck

---

*Bad software architecture is a people problem. When **people don't work well together they make bad decisions**.* -Kate Matsudaira

---

*To change the architecture of a software-intensive system ensconced in a large organization, you often have to change the architecture of the organization. And ultimately, that is **a political problem, not just a technical one**.* -Grady Booch

---

## Changing Role of Architecture

---

*The role of architects has changed from trying to be the smartest person to **making everyone else smarter**.* -Gregor Hohpe

---

*The architect's role is changing from being primarily a decision maker to being a coordinator, advisor, and knowledge manager ... **a central knowledge hub**.* -Rainer Weinreich & Iris Groher

---

*Rather than inject itself in every decision loop,* ***architecture should design mechanisms for teams to make better decisions*** *- they'll know better anyhow, and slowing them down carries a high cost..* -Gregor Hohpe

---

*Your architecture team's job is to* ***solve your biggest problems****. The best setup is the one that allows it to accomplish that.* -Gregor Hohpe

---

*Your organization has to* ***earn its way to an effective architecture function****. You can't just plug some architects into the current mess and expect it to solve all your problems.* -Gregor Hophe

---

## Complexity

---

*Excessive complexity is nature's punishment for* ***organizations that are unable to make decisions****.* -Gregor Hohpe

---

***Metawork is more interesting than work****. … we love complicated little puzzles to solve, so we keep overengineering everything.* -Neal Ford

---

*Developers are **drawn to complexity like moths to a flame** often with the same result.* -Neal Ford

---

## Other

---

*Architecture is just a collective hunch, a **shared hallucination**, an assertion by a set of stakeholders on the nature of their observable world, be it a world that is or a world as they wish it to be.* -Grady Booch

---

***Architects code**, but not to deliver code. Rather, to understand the ramifications of the decisions that they're making.* -Gregor Hohpe, Dave Farley

---

*Architectural decisions are design decisions that are **hard to make** or **costly to change**.* -Olaf Zimmermann

---