Grounded Architecture

Part II: On Being Architect

Željko Obrenović

Grounded Architecture

Part II: On Being Architect

Željko Obrenović

This book is available at https://leanpub.com/groundedarchitecture

This version was published on 2025-05-26

Contents

1: On Being Architect: Introduction

image by azmanl from istock

IN THIS SECTION, YOU WILL: Get an overview of lessons I learned on what it means to be an architect in practice.

With this section, we begin the **second part of our book**, which presents a wealth of **practical ideas and inspiration** to help you implement the Grounded Architecture framework outlined in the first part. This section will equip you with **valuable tips and insights** for **running an IT architecture practice** (see Figure 1). In complex organizations, simply copying and pasting the approaches of other companies is not effective. Instead of relying on a rigid, structured framework, it is more practical to have a **toolbox of inspirational resources** for establishing a **successful architectural practice**.

We start this second part with various resources that explore **what it means to be an architect**. The role of an architect in the IT industry is akin to that of a **Swiss Army knife—multifaceted and essential**. It requires a blend of **technical expertise**, **strategic thinking**, and **interpersonal skills** to achieve successful outcomes in **complex organizational environments**. In this section, I will provide an **in-depth perspective** on what it truly means to be an architect in practice, offering a **comprehensive understanding** of the responsibilities and expectations associated with this role.

By examining these perspectives, I hope to uncover the **essential qualities and responsibilities** of architects. My exploration emphasizes the importance of a **well-rounded skill set** and a **strategic approach**, illustrating that being an architect involves much more than simply wearing stylish glasses and nodding thoughtfully in meetings. Whether you are an **aspiring architect** or a **seasoned professional**, I hope these insights will provide **valuable guidance** for navigating and excelling in this **dynamic and evolving field**.

1.1: Growing As An Architect

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:

- Skills
- Impact
- Leadership

Architects must have a **minimal "length"** of all of these "legs" to be successful (Figure 2). For instance, having skills and impact without leadership frequently leads to **hitting a glass ceiling**. Such architects plateau at an intermediate level and cannot direct 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.

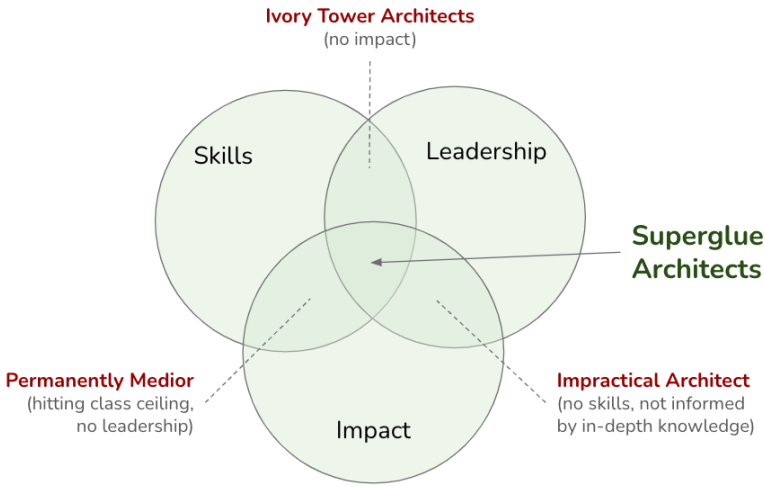

Figure 2: *Architects must have a minimal "length" of all "legs" to be successful.*

1.1.1: Skills

Architects must have a solid skill set, possessing both knowledge and the ability to apply relevant knowledge in practice. These skills should include technical (e.g., cloud architecture or Kubernetes technology) as well as communication and influence skills.

A typical skillset of an architect includes:

- **Hard (technical) skills**: including extensive knowledge of both new technology and legacy technology stacks,
- **Soft skills**: like the ability to calm a panicking developer or decode the cryptic language of a business executive,
- **Product development knowledge**: knowing what makes a product tick,
- **Business domain knowledge**: understanding those endless spreadsheets,
- **Decision-making skills**: choosing the right path, even when all options look doomed.

The section Skills provides more details.

1.1.2: Impact

Impact should be measured as a benefit for the business. Architects need to ensure that what they are doing profits the business. 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 organizational strategies,
- **Process** optimizations and improvements with real, measurable impact on the work of an organization,
- **Cost** optimizations of systems based on data-informed decisions,

- Developing pragmatic **technology strategies**, helping businesses reach goals sustainably,
- Driving **delivery of products**, supporting teams to increase quality and speed of delivery,
- Supporting **business innovation**, bringing new pragmatic ideas aligned with business strategy and goals.

The section Impact provides more details.

1.1.3: Leadership

Leadership 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 your organization** and reach the industry at large.
- They **lead efforts** that **solve important problems**.
- 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 organization.

I also support Gregor Hohpe's view that **mentoring other architects** is one of the most crucial aspects 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 section Leadership Traits and Leading with Language: Six Plays for Grounded IT Architects provides more details.

1.2: Thinking Like an Architect

In IT organizations, architects are the **essential connectors**, often referred to as "superglue." Their primary function is integrating different aspects of the organization—architecture, technical details, business requirements, and team dynamics—ensuring that everything works together seamlessly. This role is vital in large organizations and complex projects where cohesion and coordination are paramount to success. The section Thinking Like an Architect: Architects as Superglue provides more details.

Balancing curiosity, doubt, vision, and skepticism is essential for driving sustainable innovation and change in organizations. Architects must constantly reflect on how well they balance these forces. In doing so, they can ensure that their efforts are driven by a healthy combination of exploration, critical validation, strategic guidance, and cautious realism—all crucial to achieving lasting success in any organizational change. The section Balancing Curiosity, Doubt, Vision, and Skepticism provides more details.

The ideal career path for architects is rooted in a strong engineering background. This foundation provides the technical proficiency necessary for advanced architectural roles. Moving from an engineering role to an architecture role involves broadening scope, increasing diversity, and developing strong communication and influencer skills. Architecture roles can lead to tech leadership positions such as Engineering Director or Chief Technology Officer (CTO), leveraging strategic vision, decision-making, and leadership skills.

The section Architects' Career Paths provides more details.

2: Thinking Like an Architect: Architects as Superglue

image by pagadesign from istock

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 technical issues with business and broader issues.

To succeed as an IT architect, you need the right skills and a suitable **mindset**—a collection of attitudes, beliefs, and mental frameworks that shape how you perceive and respond to various situations. I have found the "**superglue" metaphor** to be a particularly effective way to encourage architects to adopt the mindset necessary for success and making a significant impact in organizations.

The concept of "superglue" in IT organizations highlights the critical role of individuals who serve as the **binding force** across various aspects of the organization. This idea, championed by Adam Bar-Niv and Amir Shenhav from Intel[1] and echoed by Tanya Reilly[2], emphasizes the importance of individuals who excel beyond just technical skills. Gregor Hohpe similarly describes the primary role of modern architects as those who connect different organizational functions by navigating the Architect Elevator[3] from the penthouse, where business strategies are established, to the engine room, where engineers implement essential components, services, and systems.

Architects who act as "superglue" should strive to integrate architecture, technical details, business needs, and people within large organizations or complex projects. They should serve as the "**organizational connective tissue**," with their main strength lying in their ability to relate technical issues to broader organizational and business contexts.

[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

[3] https://architectelevator.com/

2.1: Superglue Architects

While technical expertise is essential, what distinguishes superglue architects from technical specialists is their exceptional **relational skills**. They must communicate effectively, negotiate, and influence various stakeholders to achieve alignment and drive projects forward. This unique combination of technical knowledge and interpersonal skills is what makes architects invaluable for maintaining the organization's coherence and efficiency.

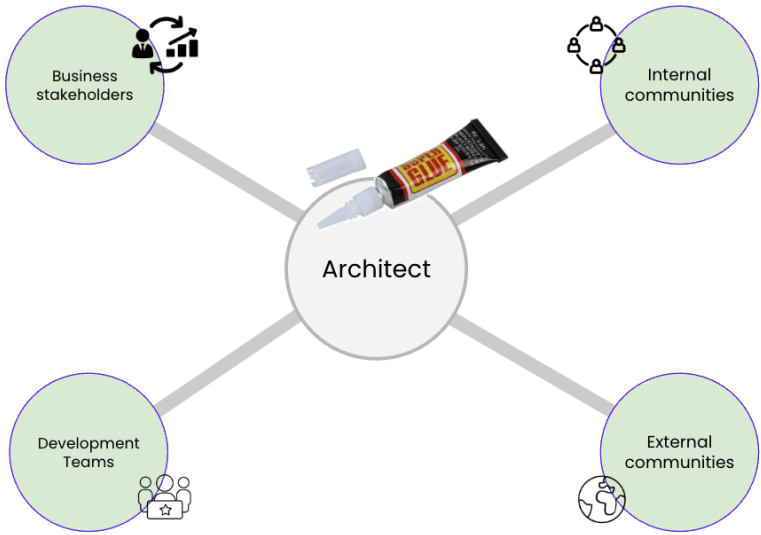

***Figure 1**: Architects serve as superglue, connecting development teams with business stakeholders and linking their teams with internal and external communities.*

Figure 1 illustrates the **superglue metaphor** for architects, demonstrating how they, like superglue, bond various parts of the organization:

- **Developer Whisperers**: Architects should work closely with developers to understand their challenges and ensure that architectural decisions enhance development efficiency and effectiveness.

- **Tech-to-Business Translators**: They should translate technical jargon into business terms, helping stakeholders understand the value and implications of technical decisions.
- **Cross-Functional Diplomats**: By engaging with operations, marketing, and finance, architects can ensure that technical solutions are practical, viable, and aligned with organizational strategies.
- **Community Connectors**: Active engagement within internal communities keeps architects informed and contributes to the collective knowledge pool.
- **Industry Influencers**: By being visible externally, architects can learn from and influence the broader industry, bringing in fresh perspectives and establishing their organization as a thought leader.

Superglue architects play a unique and critical role in the smooth operation of large and complex IT organizations. Unlike superheroes who save the day with dramatic rescues, superglue architects ensure continuous and seamless operations by connecting various aspects of the organization.

IT architects should focus on holding everything together, ensuring their organization's stability, coherence, and progress. Think of them as the organizational equivalent of duct tape—versatile, indispensable, and always ready to fix the seemingly unfixable.

2.2: Superglueing in Action #1: Aligning Organizational Goals

Tensions among **technology**, **product**, **organizational**, and **business functions** can slow down progress, lead to poor decisions, introduce complexity, and result in missed opportunities. Architects can act as the "superglue" that mitigates these issues by **fostering better communication and alignment** among these elements. The aim is **not to create new barriers** but to bring these functions closer together, ensuring a cohesive and efficient operation.

image by flamingoimages from istock

The primary value of "superglue" architects in complex organizations lies in their unique ability to align business, product, technology, and organizational functions.

2.2.1: Tensions

Technology, product, organizational, and business functions face specific challenges and change at different rates. Ideally, these structures

should **evolve simultaneously**, staying in perfect sync, much like a well-rehearsed dance troupe. However, in reality, they often resemble a **poorly synchronized** flash mob, full of tension and missteps, as illustrated in Figure 2.

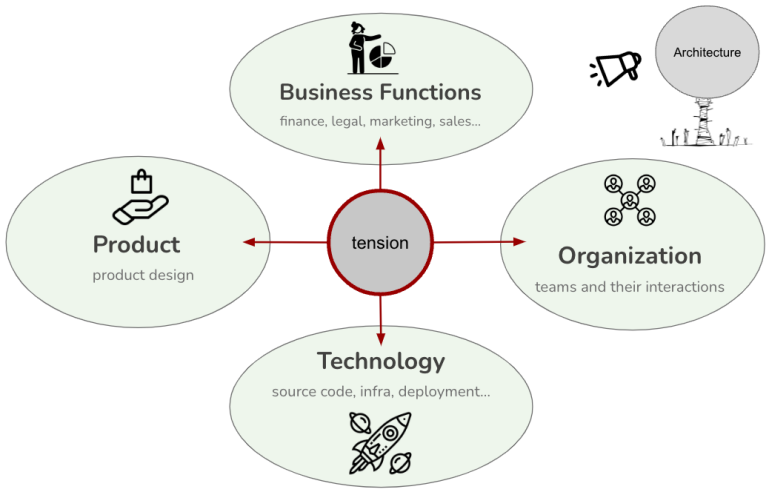

Figure 2: *The tensions between technology, product, organization, and business functions.*

Tension between technology, product, organization, and business functions can significantly impede progress. Miscommunication or misalignments can turn simple tasks into bureaucratic nightmares. Furthermore, these issues can introduce overwhelming and unnecessary complexity, causing missed opportunities. Here are specific examples of these tensions:

Technology vs. Organization:

- **Example**: An organization adopts a monolithic IT system that centralizes all operations for efficient management. However, development teams face bottlenecks when making changes because all teams are interconnected through a single codebase. This centralization creates a traffic jam of dependencies, slowing down

deployments and requiring teams to coordinate their work in unnecessarily complex ways.
- **Impact**: This structure hinders team autonomy, making rapid iteration and innovation impossible. Teams cannot work in isolation, and even a slight change in one area can inadvertently affect another, resulting in downtime and delays.

Product vs. Technology:

- **Example**: An organization implements a microservices architecture, allowing each team to own a service domain and enabling independent development and deployment. However, the product's feature requirements are organized differently—by user experience across multiple domains. Consequently, a seemingly simple product change, such as a new user interface, necessitates modifications across various microservices.
- **Impact**: This situation forces cross-team coordination for minor product changes, akin to playing "Whac-A-Mole." Each small adjustment can ripple across several teams, increasing complexity and slowing progress.

Business vs. Product:

- **Example**: A company's business strategy frequently shifts between cost-cutting and innovation. At one point, the priority is to reduce operational expenses and postpone new features. A few months later, the focus shifts toward rapid product innovation and cloud migration to remain competitive. Meanwhile, product teams struggle to align their development roadmaps with these conflicting objectives.
- **Impact**: These conflicting priorities result in confusion across teams. While the business insists on cost reduction, product teams are tasked with delivering new features and pivoting technology platforms, leading to overwhelmed teams and unsustainable project timelines.

Organization vs. Business Functions:

- **Example**: The organizational structure is designed around siloed departments such as IT, marketing, and finance, each with distinct goals and KPIs. However, the business function requires cross-department collaboration to achieve strategic objectives, such as launching a new product or migrating to the cloud. The departments, with conflicting processes and isolated goals, often miscommunicate and delay delivery.
- **Impact**: This tension causes project bottlenecks, unclear decision-making, and slow response times, as each department operates independently instead of moving toward a shared business objective. The lack of collaboration increases friction and raises the risk of missed market opportunities.

When not addressed, these tensions can severely limit an organization's ability to remain competitive, agile, and responsive to market and technological changes.

2.2.2: Superglue Role in Reducing Tensions

Superglue architects should be able to navigate the wild complexities of modern organizations. They should facilitate that the various architectures—business, product, technology, or organizational design—don't just coexist but thrive together. Architects' **holistic approach** not only reduces tension and misalignment but reassures the organization's smooth operation, like butter on hot toast.

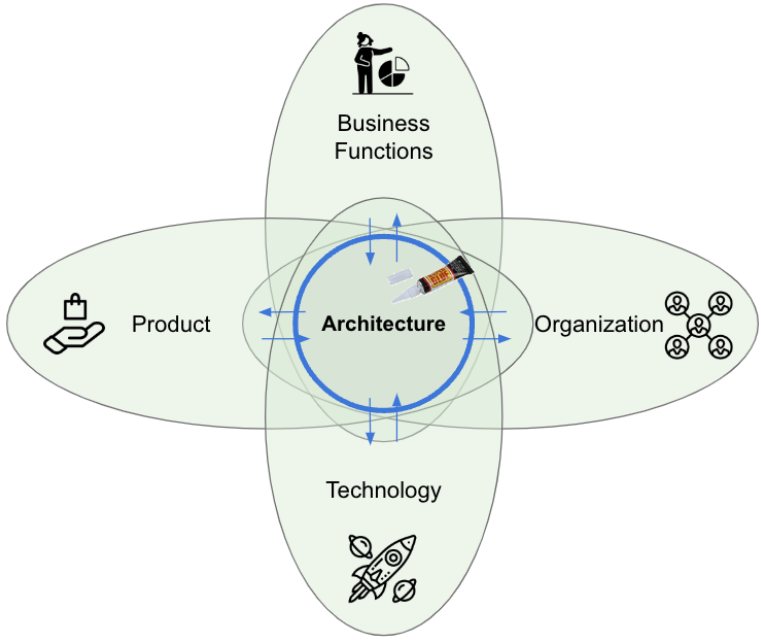

***Figure 3**: Architects should be in the middle of reducing tensions between technology, product, organization, and business functions.*

Acting as superglue, the architects can effectively **reduce tensions** among technology, product, organization, and business functions. This act includes facilitating **critical conversations** between these units and ensuring alignment and cohesion. As depicted in Figure 3, the goal of architecture is not to add new constructs between these four elements, but to **bring them closer together**. By fostering closer relationships and better communication, architects can help these elements work in unison rather than at cross purposes. Here are examples of how architects reduce tensions and ensure these different domains work cohesively:

Aligning Technology with Business Goals:

- **Example**: An organization is undergoing a digital transformation, adopting cloud infrastructure to reduce costs and increase scalability. The business wants to shift to cloud services to improve profitability quickly. The architect's role here is to assess the

current technology stack and highlight the potential trade-offs between speed, cost savings, and technical debt.
- **Architect's Role**: The architect ensures that the cloud migration strategy aligns with both business cost-cutting goals and the long-term sustainability of the technology stack. By communicating how aggressive cloud migration might lead to technical debt and future maintenance costs, the architect keeps both business and technology stakeholders aligned, avoiding unrealistic expectations or missed deadlines.

Balancing Product Roadmaps with Organizational Capabilities:

- **Example**: A product team wants to roll out new features based on customer feedback. However, the development teams are already stretched thin with existing workloads, and the current organizational design doesn't support the speed required for this new product push.
- **Architect's Role**: The architect engages the product team and the organizational leadership to ensure the product roadmap is realistic, given the current team structure and workload. They suggest restructuring teams or adopting different DevOps practices to improve delivery speed. By facilitating these conversations, the architect helps product teams avoid bottlenecks and missed deadlines while aligning with the organization's resource capabilities.

Translating Technical Jargon for Business Stakeholders:

- **Example**: The business unit is planning an aggressive cost-cutting strategy to reduce IT expenditures by slashing infrastructure budgets. At the same time, the tech team is preparing for a major system upgrade to improve long-term scalability and security.
- **Architect's Role**: The architect steps in to explain the technical ramifications of cost-cutting to business leaders in non-technical language, such as the risks of underfunding system upgrades or deferring maintenance. They can frame technical debt and potential security vulnerabilities in terms of business risk, making it easier for business leaders to see the bigger picture. This understanding can ensure that critical upgrades are not sacrificed for short-term savings.

Ensuring Product and Technology Co-Evolution:

- **Example**: The technology team wants to adopt a new containerized microservices architecture to improve scalability, while the product team is still focused on rolling out features built on an outdated monolithic system. The mismatch creates tension between rapid product releases and long-term technology goals.
- **Architect's Role**: The architect can ensure that both teams' goals are aligned by creating a phased migration plan. They can propose that the product team start developing new microservices architecture features while maintaining the monolithic system until a complete transition is feasible. This dual track can enable continuous feature delivery without sacrificing long-term technical goals.

Facilitating Critical Conversations Across Teams:

- **Example**: An organization's technology team pushes for agile methodologies to enable rapid development and deployment. At the same time, the business unit still operates on annual planning cycles, making it challenging to sync timelines.
- **Architect's Role**: The architect can organize cross-team workshops to help both sides understand each other's processes and goals. By explaining how agile practices can enhance product delivery speed and adjust to market needs faster, they can help the business side adjust its planning cycles. This explanation can reduce the friction between long-term planning and agile execution, fostering better alignment between business and technology functions.

Mitigating Conflicting Business Objectives:

- **Example**: The business leadership wants to reduce costs and increase product innovation simultaneously, which may place contradictory demands on the IT department. Innovation requires investment in new technologies and R&D, while cost-cutting restricts resources.
- **Architect's Role**: The architect can propose solutions that balance both goals, such as adopting cloud services that can reduce operational costs while offering flexibility for innovation. By integrating the right cloud-native solutions, the architect can ensure the company can innovate without sacrificing budget constraints, thus harmonizing conflicting business objectives.

The “superglue architect” helps navigate these tensions by being deeply involved in the organizational, technical, and product conversations. They bridge the gap between different stakeholders, ensuring smooth collaboration. Their success is measured by the seamless operation of business, product, and technical goals, even when their role becomes less visible over time.

2.3: Supergluing In Action #2: Aligning Discussions About Problems, Solutions and Implementations

The Theory of Constraints (TOC)[4], developed by Eliyahu M. Goldratt, is another source of inspiration I used to work as a superglue architect. TOC is a management philosophy focusing on identifying and addressing the most critical bottleneck (constraint) to improve overall performance.

image by peopleimages from istock

While Goldratt's work is much broader, the part I find inspiring is his view on resistance to change, which he sees as manifesting in three forms: disagreement about the problem, disagreement on the solution, and disagreement on the implementation (Figure 4). This model is a useful guide to help people resolve disagreements. This model can also be useful in showing that many disagreements stem from people not realizing they are talking about different topics (e.g., trying to decide on low-level technology choices while there is no agreement on which problem they

[4] https://en.wikipedia.org/wiki/Thinking_processes_(theory_of_constraints)

are solving).

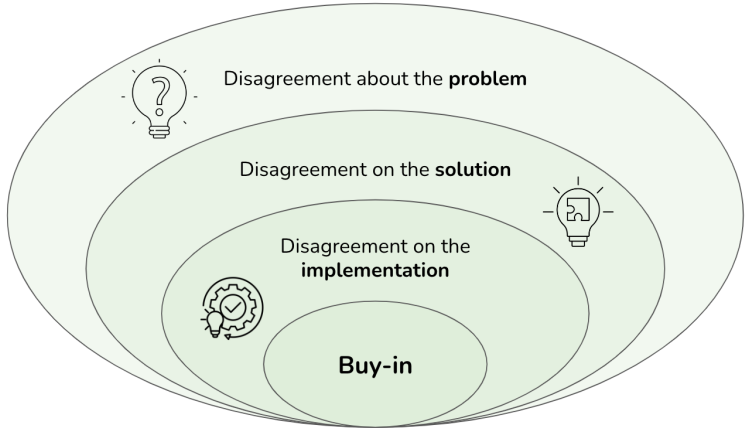

***Figure 4**: Goldratt's view on resistance to change: disagreement about the problem, disagreement on the solution, and disagreement on the implementation.*

2.3.1: Disagreement about the Problem (What to change?)

Disagreement about the problem occurs when stakeholders or team members perceive the issue differently. In IT architecture and software engineering, this can lead to significant delays and inefficiencies because you might direct resources toward solving a problem that isn't the root cause of the constraint.

For example, consider an IT team responsible for maintaining an e-commerce platform. The platform is experiencing slow response times, as users have reported. The database team believes the issue lies in inefficient database queries, while the network team thinks the problem is due to network latency. Meanwhile, the backend application developers suspect poorly optimized code. UX designers think that reported slowness is happening on the front end and is a consequence of the poor implementation of the user interface. These teams cannot agree on the primary problem.

They may waste time optimizing areas that aren't the real bottleneck, leading to a patchwork of partial solutions that don't fully resolve the performance issue.

2.3.2: Disagreement on the Solution (What to change into?)

Once you identify a problem, the next challenge is agreeing on the best solution. This disagreement can stem from different experiences, knowledge, or biases toward particular technologies or methodologies.

In the same e-commerce platform example, the team might disagree on the solution after identifying the database as the root cause of the problem. The database administrators might suggest migrating to a more powerful database server, while the developers propose rewriting the queries more efficiently. UI developers think they should significantly refactor frontend code to a new version of the UI library they wanted to use before but did not have time for the work. The operations team might push for a backend redesign and to scale the entire system horizontally by adding more servers. These conflicting solutions can cause delays as the team debates the best approach, potentially leading to a suboptimal compromise or a solution that introduces new problems.

2.3.3: Disagreement on the Implementation Approach (How to cause the change?)

The implementation can still be contentious even when you agree upon a solution. The implementation phase is critical, as poor execution can negate the benefits of the chosen solution.

Continuing with the e-commerce platform scenario, suppose the team agrees to optimize the database queries as the solution. Disagreements might arise regarding how to implement these changes. The development team might want to rewrite all queries in one go, while the operations team might prefer a phased approach to minimize downtime. Additionally, there might be disagreements about the testing environment, deployment strategies, or even the timeline. If you do not resolve these disagreements, the implementation could be delayed or poorly

implemented, leading to new issues such as data integrity problems or complete system outages.

2.3.4: Confusion about the Level of Disagreement

A more difficult-to-spot type of confusion can arise when people are unclear about the level of disagreement they are addressing. This misalignment can be particularly problematic in IT architecture and software engineering, where project complexity often involves multiple layers of decision-making.

People may think they are discussing and disagreeing on one aspect of a project—such as the implementation—when their disagreement stems from a deeper issue, such as not being aligned on the problem itself. This understanding can lead to more productive discussions, efficient use of effort, and decisions that effectively address the core issues.

2.3.4.1: Example 1: Misalignment on the Problem vs. Implementation

Imagine a software development team optimizing a web application's performance. The team starts debating how to implement changes—whether to refactor code, adjust server configurations, or optimize the database. However, underlying this discussion is a fundamental misalignment: they disagree on the problem. Some team members believe the issue is with the application code, while others think it's the database or the server infrastructure, and third think the problem is poor UX design.

As they discuss implementation strategies, they may argue whether to prioritize refactoring code or upgrading hardware, going into a heated debate about low-level technology choices and versions of the libraries they plan to use. But this debate is fruitless because they haven't first aligned on the problem. The result is that they might implement a solution that doesn't effectively address the real issue, or they might go in circles without reaching a consensus, leading to delays and frustration.

2.3.4.2: Example 2: Misalignment on the Solution vs. Problem

Consider a scenario where an IT department is trying to improve the security of its systems. The team is debating whether to implement a

new firewall or upgrade its encryption protocols. They might think they disagree on the solution—firewall vs. encryption—but the real issue is that they haven't agreed on the specific security problem they are trying to solve.

Some team members might be focused on external threats, while others are more concerned about internal data breaches. Because they are not aligned on the problem, their discussion about the solution is confused. One group pushes for a solution that addresses external threats, while another advocates for measures that protect against internal risks. Without recognizing this misalignment, the team risks implementing a solution that only partially addresses the organization's security needs.

2.3.5: Superglue Role in Resolving Disagreements

As connective tissue, the superglue architect can help resolve disagreements with effective communication, clear documentation, and a collaborative approach to the problem:

1. **Clearly Identify and Agree on the Problem**: Before discussing solutions or implementation, ensure that everyone has a shared understanding of the problem.
2. **Differentiate Between Problem, Solution, and Implementation**: Facilitate team members to recognize and articulate their discussion level. This differentiation helps prevent confusion and keeps discussions productive.
3. **Facilitate Clear Communication**: Use structured methods like root cause analysis or problem statements to align everyone on the problem before discussing potential solutions and implementations.
4. **Seeing Multiple Dimensions**: As Gregor Hophe elaborated nicely[5], when architects encounter stalemate situations, they may try to find a new model to demonstrate that everyone is simply looking at the same thing from different perspectives.

By being aware of the potential for confusion and actively working to clarify the level of disagreement, teams ican make more informed decisions, implement effective solutions, and, ultimately, improve project outcomes.

[5] https://architectelevator.com/architecture/multiple-dimensions/

2.4: Superglueing in Action #3: Navigating Organizational Conflicts

Conflicts in organizations are inevitable, particularly in complex environments where different teams, priorities, and personalities intersect. IT architects often find themselves in the middle of these tensions, acting as facilitators who help bridge gaps between stakeholders.

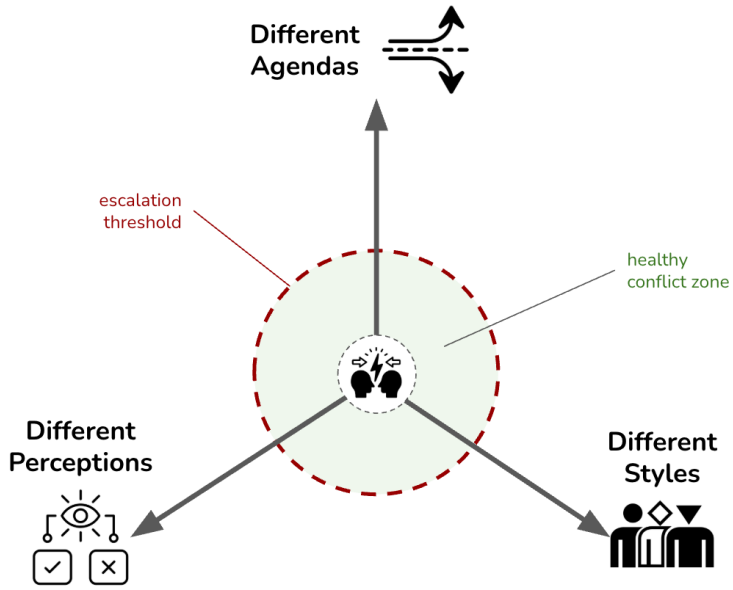

***Figure 5**: Three common sources of workplace conflict: different agendas, different perceptions, and different personal styles.*

Jeff Weiss, a conflict management expert, identifies three common sources of workplace conflict (Figure 5): different agendas, different perceptions, and different personal styles. Understanding these sources can help architects navigate disagreements more effectively.

2.4.1: Managing Conflicts Arising from Different Agendas

Teams within an organization often pursue different yet valid goals. For example, a product team may prioritize new features to gain market advantage, while an engineering team may push for reducing technical debt to improve long-term stability. These conflicting priorities can lead to friction, especially if discussions become a zero-sum game where one team's success is perceived as another's loss.

Architects can facilitate discussions that uncover the underlying motivations behind each agenda. For instance, rather than debating whether to prioritize feature delivery or technical refactoring, an architect might propose a phased approach—introducing foundational improvements alongside incremental feature releases. By framing decisions regarding business outcomes, architects can help teams find common ground rather than focusing on competing interests.

2.4.2: Addressing Conflicts Driven by Different Perceptions

Sometimes, conflicts emerge not from fundamental disagreements but from different interpretations of the same situation. A classic example is when a business leader walks away from a meeting, assuming a project will be delivered in three months, while engineers believe they agreed only to an exploratory phase.

To reduce such misunderstandings, architects can improve clarity and alignment. This improvement might involve summarizing key decisions and next steps in a shared document, using structured frameworks like architecture decision records (ADRs) or visual roadmaps. Additionally, they can encourage discussions where each party explains their perspective, helping to surface implicit assumptions before they lead to misalignment. Architecture analytics can further help understand the different perspectives and align them by providing data.

2.4.3: Navigating Conflicts Stemming from Different Personal Styles

Organizational conflicts are not always about goals or facts—sometimes, they stem from differences in how individuals prefer to work. Some professionals thrive in structured environments with strict deadlines, while others prefer flexibility and iterative decision-making. These different styles can lead to friction, particularly in cross-functional teams.

Architects can help by recognizing and adapting to these differences. For example, suppose a detail-oriented technical leader struggles to align with a high-level strategist. In that case, an architect might act as a bridge—translating broad strategic goals into actionable technical plans. Architects can foster understanding when team members have clashing communication styles by facilitating discussions that help each side appreciate the other's approach.

2.4.4: Superglue Architects as Conflict Resolvers Catalysts

IT architects are not necessarily conflict resolvers, but they often find themselves uniquely positioned to help navigate organizational tensions. Understanding the common sources of conflict—different agendas, perceptions, and styles—can facilitate productive conversations, improve clarity, and help teams work together more effectively. Whether through structured decision-making, improved documentation, or adaptive communication, architects have many ways to resolve conflicts while ensuring alignment between technology and business objectives. Tools such as the culture map can provide useful insights into the different personal styles and preferences within a team, helping architects tailor their approach to each individual.

2.5: Superglue Impact: Keeping Everyone in the Same Boat, Upon a Stormy Sea

While architects stay close to the technology, they must ensure it works for the business and customers, not the other way around. By keeping everyone **in the loop** and **aligned**, architects help clear the many **pitfalls of misalignment** and keep the organizational machine running smoothly. Their role is to facilitate communication, ensure alignment, and guide the organization toward cohesive and practical solutions, preventing the myriad risks associated with misalignment.

image by mbbirdy from istock

Misalignment between these elements can introduce several **key risks**, which superglue architects need to be aware of and can mitigate:

- **Building the Wrong Products**: If technology implementation is based on incorrect assumptions, you might create a product that doesn't meet the actual people needs. Misalignment can lead to producing irrelevant solutions, like trying to sell snow boots in the Sahara.

- **Wrong Prioritization of Activities**: Without clear business and product metrics, resources might be directed towards developing "interesting" but non-valuable products. Proper alignment ensures development efforts are focused on initiatives that add real value, not on quirky side projects.
- **Unexpected Delivery Delays**: Misalignment can lead to underestimating projects' complexity, effort, and dependencies, causing significant delays. This misalignment can make a project feel perpetually stuck, like being in a time loop where unforeseen obstacles continuously hinder progress.
- **Duplication of Effort**: Without harmonization across business and product strategies, efforts may be duplicated, leading to inefficiencies. This duplication is akin to repeatedly reinventing the wheel, which is wasteful and counterproductive.
- **Building Too Complex Products**: Overly complex and configurable systems can be developed to address all possible scenarios, resulting in cumbersome solutions where simpler, more harmonized approaches would suffice. This complexity is like using a Swiss Army knife when only a spoon is needed—overkill and overly complicated.
- **Overengineering**: Without pushback to simplify products and a lack of understanding of the technology, overengineering can occur. Imagine using a monster truck for a grocery run—impressive, but entirely unnecessary.
- **Building Too Simple, Unscalable Products**: Assuming processes will be simple when, in reality, essential complexity needs to be supported, can lead to fragile and rigid systems.
- **Building Low-Quality Products**: Unnecessary complexity and lack of critical knowledge and expertise can lead to low-quality products that fail under pressure, like a dollar-store umbrella in a hurricane. Ensuring quality requires aligning expertise and simplifying designs where possible.
- **Complicated Dependencies Between Teams**: Suboptimal organizational design and lack of awareness of system and team dependencies can slow down coordination, creating a bureaucratic nightmare. Efficient team structures and clear communication channels are crucial to maintaining momentum.
- **Creating Fragile, Unsustainable Team Structures**: Relying on a small number of developers for critical technologies can make

teams extremely vulnerable. Building resilient team structures with adequate support and redundancy is essential.

2.6: To Probe Further

- Thinking Like an Architect[6], by Gregor Hohpe, 2024
- Architects See More Dimensions[7], by Gregor Hohpe, 2020
- Architects Zoom In and Out, See Different Things[8], by Gregor Hohpe, 2020
- Architects Look For Causality[9], by Gregor Hohpe, 2020
- Architects See Shades of Gray, Look for Balance[10], by Gregor Hohpe, 2020
- Here's why enterprise IT is so complex[11], by Gregor Hohpe, 2018

[6] https://www.infoq.com/articles/thinking-like-architect/
[7] https://architectelevator.com/architecture/multiple-dimensions/
[8] https://architectelevator.com/architecture/architects-zoom/
[9] https://architectelevator.com/architecture/architects-causality/
[10] https://architectelevator.com/architecture/architects-shades-of-gray/
[11] https://architectelevator.com/architecture/it-complexity/

2.7: 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? How could you 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 of tension between your organization's technology, product, organization, and business functions? What caused this tension, and how was it addressed?*
- *How could your current architecture aid in reducing tension between these functions?*
- *Have you witnessed the architecture sitting on the side, being ignored? If so, what steps can you take to actively involve architecture in decision-making processes?*
- *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 are your strongest? Which might need more development?*

3: Architects' Career Paths

image by richvintage from istock

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

KEY POINTS:

- A strong engineering background is essential for architects to make informed technology decisions and build effective relationships with developer teams.
- Moving from an engineering role to an architecture role involves broadening scope, increasing diversity, and developing strong communication and influencer skills.
- Career tracks can include Senior Architects (broader responsibilities), Principal Architects (specialized focus), and Enterprise Architects (aligning technical strategy with business objectives).
- Architecture roles can lead to tech leadership positions such as Engineering Director or Chief Technology Officer (CTO), leveraging strategic vision, decision-making, and leadership skills.

In any job, a **career path** is a sequence of jobs or roles that a person takes on throughout their professional life. It typically involves **progression** from entry-level positions to higher-level roles with increasing responsibilities and compensation. Career paths can be **linear**, with a clear upward trajectory, or **non-linear**, with lateral moves and changes in direction.

In this section, I elaborate on possible career paths of architects. In the Appendix, you can also find additional resources for managing, developing, and hiring architects[1].

[1] growing

3.1: Solid Engineering Background

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

Architects with a strong engineering background have hands-on experience with coding, debugging, and problem-solving, which is crucial for understanding the technical complexities and constraints of any project. This practical knowledge allows them to make **more informed** and **realistic decisions** about technology stacks, architectural patterns, and system design.

When architects have a solid engineering foundation, they can **earn the trust and respect of developer teams** more easily. Developers are more likely to follow the guidance and recommendations of someone who has walked in their shoes and understands their daily challenges. An engineering background equips architects with the technical language and concepts necessary to communicate effectively with developers. This shared language helps bridge the gap between high-level architectural visions and the detailed implementation work carried out by development teams.

image by kobus louw from istock

Engineers are trained to think critically and solve complex problems. This skill set is invaluable for architects, who must navigate technical challenges, identify potential issues, and devise robust solutions that align with business goals.

While the path to becoming a successful architect is multifaceted, a strong engineering background provides a solid foundation upon which to build the necessary additional skills and knowledge.

3.2: Entering Architecture Space

While a strong engineering background is essential, architects must also develop additional skills to succeed in their roles. Stepping from an engineering position to an architecture role requires three significant changes:

1. **Broader Scope**: Architects must look beyond individual components to see the system as a whole, considering the interactions and dependencies between various parts. This holistic view is essential for creating cohesive and scalable solutions that integrate seamlessly with the entire ecosystem of the organization.
2. **Higher Diversity**: The work becomes more varied, encompassing different technologies, processes, and organizational needs. Architects need to be versatile, adapting to various domains and understanding how different technologies and processes can be leveraged to solve complex problems. This diversity also includes interacting with different teams and stakeholders, each with unique perspectives and requirements.
3. **Changing Skills**: Communication and influencer skills become crucial to success, as architects need to articulate their vision and persuade others to follow it. They must effectively communicate complex technical concepts to non-technical stakeholders, build consensus, and drive strategic initiatives. This shift emphasizes the importance of soft skills alongside technical expertise.

image by vm from istock

Architects have multifaceted responsibility that requires them to be not only technically proficient but also adept at **understanding and aligning with business goals** and user requirements.

3.3: Career Progression in IT Architecture

An architect's path can take many different directions, and the roles of this path can have many different names.

image by bowie15 from istock

People typically enter an IT architecture space as hands-on solution architects. From that position, I usually envision three tracks of progression (Figure 1):

- **Generalist Track (Senior Architects)**: These architects are generalists with broader responsibilities who can dig deep into complex issues and identify suitable courses of action. They often navigate from one critical area to another, guided by the organization's direction. Senior Architects play a pivotal role in bridging the gap between different technical teams and ensuring the overall architectural integrity of projects.
- **Specialized Track (Principal Architects)**: Senior architects with a specialized track focused on an organization's strategic interest (e.g., data, distributed systems, frontend). Principal Architects dive deep into their specialized areas, providing thought leadership and

driving innovation. They often set standards, define best practices, and mentor other architects and engineers.

- **Enterprise Track (Enterprise Architects)**: Positioned closer to product, management, strategy, and business functions, Enterprise Architects frequently serve as the right hand to senior engineering leaders. They are responsible for aligning the technical strategy with the business objectives, ensuring that the architecture supports the organization's overall goals. Enterprise Architects often work on cross-functional initiatives and play a key role in strategic decision-making.

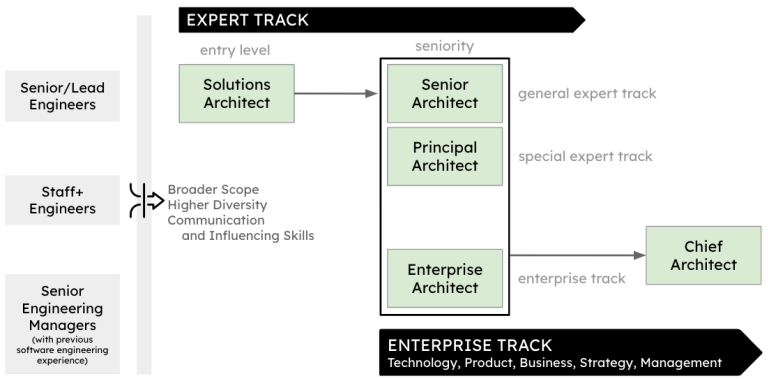

***Figure 1**: An example of IT architect career paths.*

More important than a formal title is the continuous search for relevance and the ability to **make an impact**. Successful architects are those who continuously learn, adapt, and drive meaningful change within their organizations. They focus on adding value through innovative solutions, effective communication, and strategic alignment with business goals. Whether formally recognized or not, their influence and contributions are pivotal to the success of their teams and the broader organization.

3.4: Career Progression Beyond IT Architecture

A career in IT architecture also opens possibilities of pursuing **tech leadership positions** such as Engineering Director or Chief Technology Officer (CTO). The reason for this is multifaceted.

image by miniseries from istock

Firstly, architects develop a strategic vision that **aligns technology with business goals**, a critical skill for any tech leadership role. They gain experience in making high-stakes decisions that impact the entire organization, which is directly relevant to positions like Engineering Director or CTO.

Secondly, architects often **work closely with senior management** and various departments, providing them with a broad perspective on organizational dynamics and strategic planning. This experience is invaluable for leadership roles that require a holistic understanding of both technology and business.

Thirdly, architects' roles require **strong leadership and mentorship abilities**, as they often guide and influence engineering teams. These

skills are essential for higher-level leadership positions, where the ability to inspire and manage large teams is crucial.

Lastly, the transition from an architect to a tech leadership role is a natural progression as both roles require a **deep understanding of technology**, **strategic vision**, and the ability to drive innovation within an organization. This career path leverages the architect's experience in building robust systems and their ability to foresee and mitigate potential risks, ensuring the technology infrastructure supports the organization's long-term goals.

3.5: To Probe Further

- Appendix: Resources for Managing, Growing, and Hiring Architects[2]
- Appendix: Architect Archetypes[3]
- Software Architect Archetypes[4], by Gergely Orosz, 2023

[2] growing
[3] archetypes
[4] https://newsletter.pragmaticengineer.com/p/software-architect-archetypes

3.6: Questions to Consider

- *Reflect on career paths in architecture. How can an engineering background impact effectiveness of an architect?*
- *Reflect on your career progression in architecture. How can you continuously stay relevant and make 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 you continuously raise the bar in developing and hiring architects in your organization?*
- *How could you demonstrate your communication and collaboration skills as an architect? Can you share an instance where these skills are 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 is this important?*

4: Building Skills

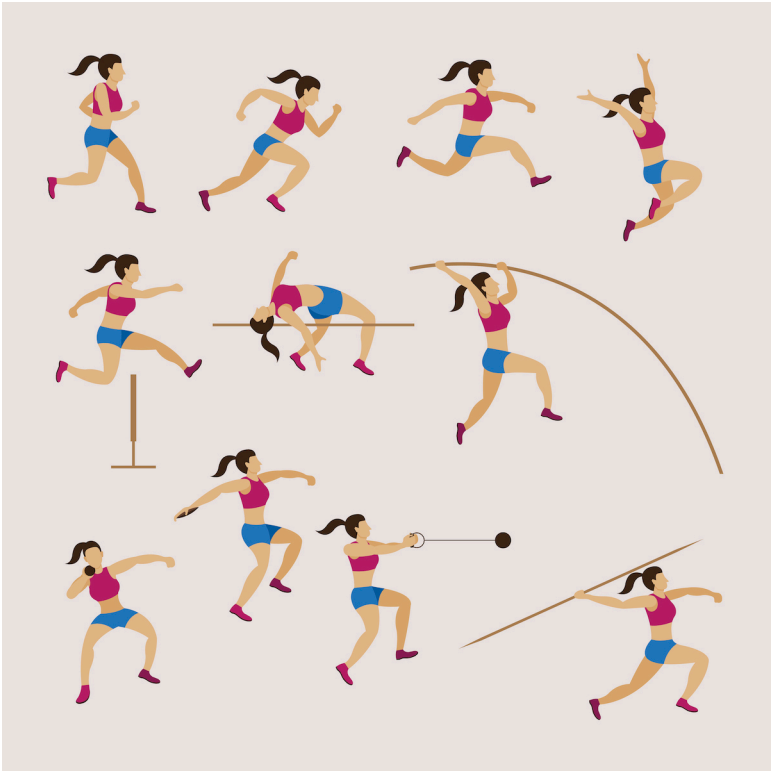

image by muchmania from istock

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

KEY POINTS:

- An architect's typical skillset includes hard (technical) skills, soft (people & social) skills, product development, business skills, and decision-making skills.
- Hard (technical) skills are essential for designing, implementing, and maintaining an organization's technology landscape.
- Soft skills are integral to social architecture, enabling individuals to navigate and contribute to these social systems effectively.

- Product development knowledge is the bridge that helps architects align technical solutions with customer needs and business objectives.
- Business domain knowledge is not just useful but essential for architects to create solutions that deliver real value.

- Decision-making skills ensure that architectural decisions are sound, sustainable, and aligned with long-term strategic objectives.

Architects must possess a comprehensive skill set to manage the complexities of modern IT environments effectively (Figure 1). By skills, I mean not only having relevant knowledge but also the ability to apply that knowledge in practical situations. These skills encompass a blend of technical expertise, communication proficiency, and influence capabilities, ensuring architects can navigate technological challenges and organizational dynamics.

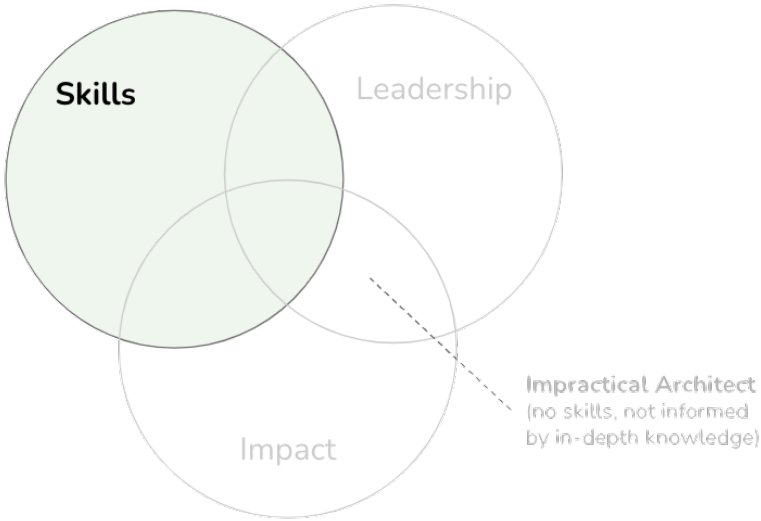

***Figure 1**: Skills are one of the three main elements of being an architect (skills, impact, leadership). Lack of skills leads to impractical decisions that are not informed by in-depth knowledge.*

Core skills of architects include:

- **Technical Skills**: Architects need a robust foundation in new and legacy technology stacks. This foundation includes proficiency in topics like cloud architecture, containerization technologies like Kubernetes, and a deep understanding of various programming languages and frameworks. Staying updated with emerging technologies and industry trends is essential for making informed decisions that align with organizational goals.
- **Communication Skills**: Effective communication is crucial for architects. They must articulate complex technical concepts in a way that is understandable to non-technical stakeholders. These skills involve clear verbal and written communication, active listening, and the ability to tailor messages to different audiences. Strong communication skills help bridge the gap between technical teams and business units, fostering collaboration and mutual understanding.

- **Influence Skills**: Architects often need to persuade and influence others, including stakeholders, developers, and executives. This requires negotiation skills, the ability to build consensus, and the credibility to advocate for necessary changes. Influence skills enable architects to drive strategic initiatives and ensure that architectural decisions are implemented effectively across the organization.
- **Product Development Skills**: Architects must understand the product development lifecycle. This understanding should include knowledge of product development methodologies, user experience design, and product management principles. Architects should be able to contribute to product strategy, ensuring that technical solutions align with customer needs and business objectives.
- **Business Domain Knowledge**: A deep understanding of the business domain is essential for architects to create solutions that deliver real value. This knowledge includes industry-specific regulations, market trends, and competitive landscape. Business acumen helps architects align technical initiatives with business goals, driving growth and innovation.
- **Decision-Making Skills**: Architects are often required to make critical decisions that impact the entire organization. This involves assessing risks, evaluating trade-offs, and making informed choices based on data and experience. Strong decision-making skills ensure that architectural decisions are sound, sustainable, and aligned with long-term strategic objectives.

4.1: Technical Skills

Hard or technical skills are the abilities and knowledge needed for **designing**, **implementing**, **and maintaining** various aspects of an **organization's technology landscape**.

image by rgstudio from istock

Some typical hard skills that architects need in their work include:

- **System design**[1]: This involves defining and developing a complex system's architecture. An architect with this skill set can create comprehensive system designs incorporating various components and sub-systems to achieve the desired functionality. System design skills ensure the architect can envision and structure systems effectively to meet organizational goals.
- **Engineering processes**[2]: An in-depth understanding of engineering processes, including the software development life cycle, Agile

[1] https://blog.pragmaticengineer.com/system-design-interview-an-insiders-guide-review/

[2] https://obren.io/tools/catalogs/?id=design-tactics-high-performing-technology-organizations

development, DevOps, and continuous delivery, is crucial. Architects must ensure their systems are developed efficiently and effectively, adhering to best practices and methodologies that enhance software development productivity and quality.

- **Design patterns**[3] **and tactics**[4]: Familiarity with design patterns and tactics such as Cloud Design Patterns, Model-View-Controller (MVC), Service-Oriented Architecture (SOA), and Microservices is essential. These patterns help architects design modular, scalable, and maintainable systems, providing solutions to common design problems and ensuring that the systems can evolve and adapt over time.
- **Security and privacy by design**[5]: With cybersecurity's increasing importance, architects need a deep understanding of security and privacy best practices. They must design secure and compliant systems with data protection regulations, incorporating security measures at every stage of the design process to protect sensitive information and mitigate risks.
- **System optimizations**[6]: Knowledge of optimizing systems for performance and scalability is critical. Architects should be adept at using tools and techniques for profiling and tuning systems to achieve optimal performance, ensuring that the systems can handle increased load and provide a seamless user experience.
- **Source code structures and maintainability**[7]: A good understanding of software engineering principles such as clean code, code maintainability, and refactoring is important. Architects should design systems that are easy to maintain and modify, promoting long-term sustainability and reducing the technical debt that can accumulate over time.
- **Reliability and stability (anti)patterns**[8] **and tactics**[9]: Understanding typical reliability and stability issues in complex systems is crucial. Architects should identify and address potential problems using patterns and tactics such as redundancy, failover, and graceful

[3] https://obren.io/tools?tag=design_patterns
[4] https://obren.io/tools?tag=design_tactics
[5] https://obren.io/tools?tag=security
[6] https://obren.io/tools/catalogs/?id=design-tactics-sig-performance
[7] https://obren.io/tools/catalogs/?id=design-tactics-sig-maintainability
[8] https://obren.io/tools/catalogs/?id=releaseit-stability-awareness
[9] https://obren.io/tools/catalogs/?id=releaseit-stability-tactics

degradation to ensure that systems remain stable and reliable under varying conditions.

- **Usability**[10]: A good understanding of usability principles is necessary for designing systems that are easy to use and provide a good user experience. Architects must ensure that their designs are user-friendly, intuitive, and accessible, enhancing end-users' overall satisfaction and productivity.

By mastering these skills, technical architects can effectively contribute to the success of their organizations, ensuring that technology solutions are well-designed, secure, efficient, and user-centric.

The section Appendix provides some pointers for resources to build your technical skills.

[10] https://obren.io/tools?q=usability

4.2: Soft Skills

> 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

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

image by peopleimages from istock

Critical soft skills include:

- **Communication skills**[11], including written[12], visual[13], verbal (presentation), and listening skills: Effective communication involves expressing oneself clearly and understanding and empathizing with others. These skills are essential for building and maintaining relationships, as well as for conveying ideas and facilitating discussions. Written communication ensures clear and concise messages, visual communication enhances understanding through diagrams and presentations, verbal communication is critical in delivering impactful presentations, and active listening fosters understanding and collaboration.
- **Networking and collaboration skills**[14]: Networking involves building and maintaining diverse professional connections.

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

Collaboration skills encompass working effectively with others, regardless of their role or seniority. These skills include partnering with peers, junior and senior colleagues, managers, and executives to achieve common goals. Effective networking opens doors to new opportunities and resources, while strong collaboration skills ensure that team efforts are synergistic and productive.

- **Organizational and time management skills**[15]: 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. Mastering these skills helps individuals stay on top of their responsibilities, reduces stress, and enhances overall productivity.
- **Analytical, strategic thinking, and problem-solving skills**[16]: Analytical skills involve assessing and interpreting complex information to make informed decisions. Strategic thinking is the capacity to envision and plan for long-term success, while problem-solving skills include identifying and addressing challenges creatively and effectively. These skills are essential for recognizing and capitalizing on unique opportunities and creating organizational value. By honing these abilities, individuals can contribute to innovative solutions, drive strategic initiatives, and confidently navigate complexities.

By developing these soft skills, individuals can significantly enhance their ability to function effectively within organizations. These skills foster environments where collaboration, efficiency, and innovation thrive, leading to personal success and contributing to the overall health and productivity of the organizations and communities they are a part of.

The Appendix provides some pointers for resources to build your soft skills.

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

4.3: Product Development Skills

Product development is creating and bringing new products or services to the market. It involves the entire journey from the conception of an idea to the product's final development, marketing, and distribution. Product development encompasses various activities and stages to transform an initial concept into a tangible and market-ready offering. 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.

image by tirachard from istock

Some processes specific to product development include:

- **Idea Generation**: This stage involves generating and exploring new product ideas. Ideas can come from various sources, such as market research, customer feedback, technological advancements, or internal brainstorming sessions. It's a creative phase where innovation is encouraged, and a broad range of ideas are considered.
- **Market Research**: Market research assesses the feasibility and potential success of the product concept. It involves gathering information about customer needs, preferences, market trends, competition, and other relevant factors to validate the product's

viability in the target market. This step helps understand the market landscape and identify the product's unique selling points.

- **Product Design and Development**: This phase involves turning the validated concept into a detailed product design. It includes creating prototypes, testing, and refining the product design based on feedback and performance metrics. Collaboration between design and engineering teams is crucial to ensure the product is functional, aesthetically pleasing, and manufacturable.
- **Testing and Validation**: The product undergoes rigorous testing to ensure it meets quality standards and performs as expected. This stage includes usability testing, user performance testing, and sometimes beta testing with a limited audience. Feedback collected during this phase is used to make necessary adjustments and improvements.
- **Marketing and Launch**: Before launching the product, marketing strategies and plans are developed to create awareness, generate demand, and promote the product to the target market. The activities include branding, pricing, distribution, and marketing communication activities. A successful launch plan ensures the product reaches the intended audience effectively.
- **Post-Launch Evaluation and Iteration**: After the product launch, it is important to monitor its performance in the market. This activity includes collecting customer feedback, analyzing sales data, and assessing overall market reception. Continuous improvement is essential; iterations based on this feedback help refine the product and address any issues.

Understanding the product development process is not just beneficial, but essential for architects. It requires a multidisciplinary approach involving teams from various functions such as product management, design, engineering, and marketing. This understanding is crucial for architects, as it helps them align their technical designs with the overall product strategy, ensuring that a product is not only technically sound but also meets market demands and customer expectations. The process aims to create innovative, desirable, and commercially viable products that meet customer needs and provide a competitive advantage in the market.

4.4: Business Skills

Regardless of their technical or design expertise, architects must have **a solid understanding of business processes** to effectively contribute to an organization's success.

image by azmanl from istock

Essential business skills for architects include:

- **General Business Concepts Knowledge**: A fundamental understanding of general business concepts is essential for architects to make informed decisions and effectively communicate with stakeholders. Familiarity with finance, marketing, sales, operations, and strategy can provide a strong foundation for architects to engage with various aspects of an organization. This broad knowledge base helps architects to see the bigger picture, understand how their technical decisions impact the business, and ensure their designs support the organization's strategic objectives. The Personal MBA[17] book is a valuable resource for familiarizing oneself with such

[17] https://personalmba.com/

concepts, offering insights into essential business principles and practices.

- **Specific Business Domains**[18] of the Organization: Besides general business concepts, architects should also develop a deep understanding of the specific business domain in which their organization operates. This knowledge includes 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. For example, an architect in the healthcare industry needs to understand healthcare regulations, patient care standards, and the evolving needs of healthcare providers and patients.
- **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. Practical business analysis ensures that the solutions architects design are aligned with business goals and deliver tangible value. This process often includes conducting interviews, facilitating workshops, and using techniques such as SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) and business model canvases to capture and prioritize requirements.
- **Stakeholder Management**: Communication and relationship management with stakeholders is crucial. Architects must understand the interests and concerns of different stakeholders, including executives, managers, employees, and customers. This understanding helps gain buy-in for architectural decisions and ensure that the solutions meet stakeholder needs.
- **Project Management**: Basic project management skills enable architects to oversee the implementation of their designs, ensuring projects are completed on time, within scope, and budget. Understanding project management methodologies can help architects work effectively with project managers and development teams.
- **Finance**: Architects should understand basic financial principles, including budgeting, cost-benefit analysis, return on investment

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

(ROI), and "Earnings Before Interest, Taxes, Depreciation, and Amortization" (EBITDA). This knowledge helps them make cost-effective decisions and justify the financial viability of proposed solutions.

- **Strategic Thinking**: Architects need to think strategically and understand how their work supports the organization's long-term goals. This thinking includes aligning technology initiatives with business strategy, identifying opportunities for innovation, and anticipating future business needs.
- **Change Management**: Implementing new systems and architectures often involves significant organizational change. Architects should be familiar with change management principles to help facilitate smooth transitions, manage resistance, and successfully adopt new technologies and processes.

By mastering these business skills, architects can ensure that their technical solutions are innovative, efficient, and aligned with their organizations' broader business goals and strategies. This holistic approach enables architects to contribute more effectively to organizational success and drive meaningful business outcomes.

4.5: Decision-Making Skills

Architects' work always requires pragmatic decision-making. Decisions are the steering wheel of organizations, and architects who are not involved in key decisions will have a limited impact on the organization.

image by gorodenkoff from istock

Architects will have three roles concerning decision-making:

1. **Be actual decision-makers**
2. **Be advisors to decision-makers**
3. **Evaluate and provide feedback on the decisions of others**

Essential decision-making skills include:

- **Understanding that a decision is an irrevocable allocation of resources**: This involves recognizing that every decision commits resources such as money, human effort, time, physical actions, and opportunities. Architects must consider the long-term implications of these allocations, ensuring that resources are used efficiently and effectively to achieve organizational goals.

- **Know the basics of decision intelligence**: Decision intelligence is the discipline of turning information into better action. It involves leveraging data, analytics, and cognitive science to enhance decision-making processes. By understanding and applying decision intelligence, architects can make more informed, evidence-based decisions that lead to better outcomes.
- **Understand key problems of poor decision-making, such as the outcome bias**: Outcome bias occurs when decisions are judged based on their results rather than the quality of the decision-making process itself. Architects need to recognize and mitigate this bias by focusing on the decision-making process, ensuring that decisions are made systematically and rationally, regardless of the outcomes.
- **Know when to use intuition**: Intuition can be a valuable tool in decision-making, especially when data is limited or time is of the essence. Architects should understand when to rely on their intuition and when to seek additional information.
- **Understand that there is no such thing as not making a decision**: Delaying or postponing a decision is a decision with its own set of consequences. Architects must know the implications of inaction and understand that deferring decisions can lead to missed opportunities, increased risks, and reduced organizational agility.
- **Risk Assessment**: Understanding and assessing risks is crucial for making informed decisions. Architects should evaluate potential risks associated with different options and develop strategies to mitigate these risks.
- **Collaborative Decision-Making**: Involving relevant stakeholders in decision-making ensures that diverse perspectives are considered, leading to more robust and well-rounded decisions. Architects should facilitate collaboration and consensus-building among team members.
- **Ethical Considerations**: Architects must ensure their decisions align with ethical standards and organizational values. These actions involve considering the broader impact of decisions on stakeholders, society, and the environment.
- **Adaptability and Flexibility**: In a rapidly changing business environment, architects must be adaptable and open to revisiting and revising decisions as new information becomes available. Flexibility allows for continuous improvement and responsiveness to emerging challenges and opportunities.

Architects can enhance their ability to influence and drive organizational success by developing these decision-making skills. Whether acting as decision-makers, advisors, or evaluators, architects are critical in steering their organizations toward strategic goals and sustainable growth.

4.6: Integrating Skills for Success

Architects must integrate these diverse skills into a cohesive approach to be truly effective. They must be lifelong learners, continuously updating their knowledge and adapting to new challenges.

image by colorsandia from istock

Here's how these skills come together in practice:

- **Technical Proficiency**: Enables architects to design robust, scalable solutions that leverage the latest technologies.
- **Effective Communication**: Ensures that architectural visions are conveyed and understood across all levels of the organization.
- **Influence and Persuasion**: Help garner support for architectural initiatives and drive change.
- **Product Insight**: Aligns technical solutions with customer needs and market demands.
- **Business Understanding**: Ensures that architectural decisions contribute to the organization's strategic objectives.
- **Decisive Leadership**: Guides the organization through complex technological landscapes with confidence and clarity.

Architects can effectively manage the interplay between technology, business, and organizational dynamics by developing and honing these skills. This comprehensive skill set enhances their ability to design innovative solutions and ensures they can lead teams and influence stakeholders to achieve a cohesive and successful architectural vision.

4.7: To Probe Further

- Appendix: Bookshelf[19]
- Old Books that Every Architect Should Read[20], by Gregor Hohpe, 2024
- Back from the engine room[21], by Gregor Hohpe, 2023
- Debugging Architects[22], by Gregor Hohpe, 2021

[19] bookshelf
[20] https://architectelevator.com/architecture/classic-architecture-books/
[21] https://architectelevator.com/transformation/debugging-architect/
[22] https://architectelevator.com/architecture/engine-room/

4.8: Questions to Consider

- *On a scale from 1 to 10, how would you rate your current architectural skill sets, considering technical, communication, product, business skills, and decision-making 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?*
- *Do you need to develop specific hard skills to enhance your architectural performance?*
- *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 you need to develop or improve to succeed 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 why?*

5: Making Impact

image by tuiphotoengineer from istock

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 on their impact on the organization.
- Architects can make an impact via three pillars: Big-Picture Thinking, Execution, and Leveling-Up.

Architects' work is evaluated based on **their impact on the organization** (Figure 1).

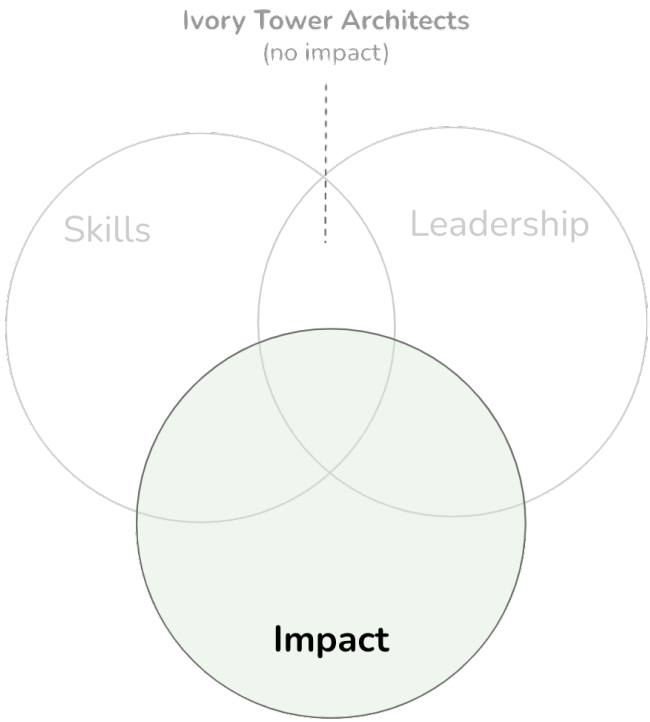

***Figure 1**: Impact is one of the three main elements of being an architect*

(skills, impact, leadership). Leadership without impact lacks foundation and may signal that you have become an ivory tower architect with a weak relation to reality.

Architects' contributions are typically assessed through the following key areas:

- **Identifying, Tackling, and Delivering on Strategic Problems**: Architects are responsible for recognizing and addressing strategic issues at both the organizational and area-specific levels (such as domains or technical areas). They can facilitate aligning work with the organization's broader strategic objectives, ensuring that efforts are prioritized to support global goals. By tackling these strategic problems, architects contribute to the organization's overall direction and success, providing solutions that align with long-term visions and targets.
- **Having a Deep and Broad Influence**: Architects must have a profound and extensive impact on their specific domain, product, or technology area. This influence requires them to delve deeply into particular critical issues, providing targeted solutions that address pressing challenges. Simultaneously, they need to maintain a broad perspective, creating value by leveraging their solutions across multiple teams and projects. This dual focus ensures that their influence is both concrete and widespread, enhancing the overall efficiency and effectiveness of the organization.
- **Delivering Solutions that Few Others Can**: Architects are often tasked with providing solutions that are beyond the reach of others, either through their direct efforts or by orchestrating large-scale group endeavors. Their unique combination of hard technical skills and soft skills in strategy, execution, and people management enables them to navigate complex challenges and drive significant progress. By leveraging these skills, architects can move the organization forward, ensuring that the solutions they deliver are both innovative and implementable, fostering growth and development across the board.

Architects make a significant impact by identifying and solving strategic problems, exerting deep and broad influence across domains, and delivering unique solutions through a blend of technical expertise and strategic leadership. This multifaceted approach ensures that their work is not only

aligned with organizational objectives but also drives substantial progress and innovation.

5.1: Pillars of Impact

Architects must possess **strong technical**, **people**, **and business skills**, which are ideally developed through extensive practice and experience. Building on this robust foundation, architects need to cultivate specific competencies that enable them to leverage their experiences and abilities to **positively impact organizational performance**. As architects advance in their careers, their competency development should be increasingly driven by the desired impact on the organization rather than solely on acquiring new skills.

I typically coach architects within the framework of concrete activities, focusing on real-world challenges and guiding their development through hands-on involvement in appropriate actions. This practical approach helps tailor their skill development to address specific challenges and achieve the expected impact in practice.

To develop a structured approach to teach architects how to make an impact, I draw inspiration from Staff Engineering roles. Tanya Reilly's book *The Staff Engineer's Path*[1] and Will Larson's book *Staff Engineer: Leadership beyond the management track*[2] provide valuable insights into defining the responsibilities and expectations of architects.

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

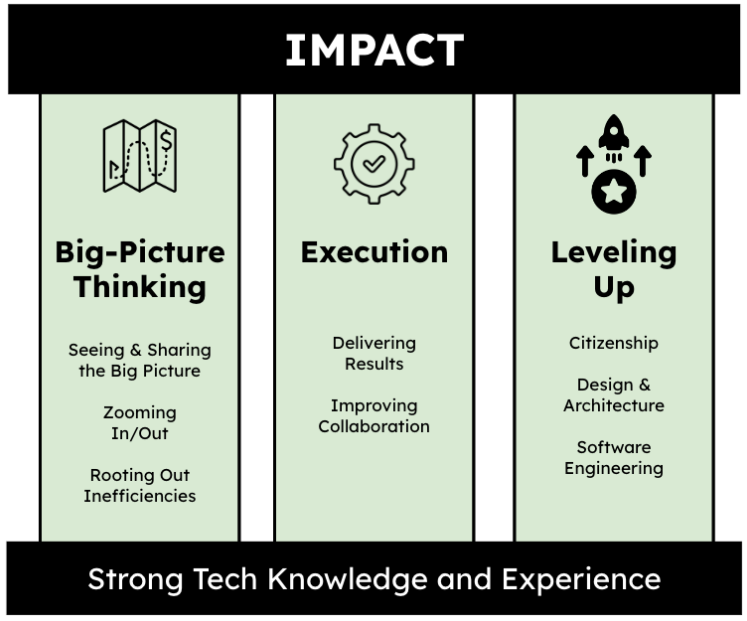

Figure 2: *Key competencies of architects. Inspired by The Staff Engineer's Path by Tanya Reilly.*

Inspired by *The Staff Engineer's Path* by Tanya Reilly (Figure 2), I categorize the competencies that enable architects to make an impact into three groups:

- **Big-picture Thinking**: Architects need to develop the ability to see the broader context of their work. This type of impact involves understanding how their architectural decisions align with organizational goals, market trends, and technological advancements. Big-picture thinking allows architects to foresee potential challenges and opportunities, ensuring their designs are future-proof and scalable. This competency involves strategic planning, vision setting, and the ability to articulate how technical solutions contribute to overall business objectives.
- **Execution**: Execution is about turning ideas into reality. Architects must help teams complete projects efficiently and effectively. This type of impact involves project management skills, the ability to coordinate with cross-functional teams, and a deep understanding of

the technical intricacies of bringing a project to fruition. Architects must also be capable of troubleshooting issues, adapting to changes, and ensuring that the final deliverables meet the required standards and expectations.

- **Leveling Up**: Leveling up refers to the continuous improvement of both the architect's skills and the capabilities of their teams. Architects must be committed to personal growth and the development of those around them. This type of impact involves mentoring junior team members, fostering a culture of learning, and staying abreast of the latest technological advancements and industry best practices. Architects should also focus on enhancing the overall skill set of their teams, ensuring that knowledge and expertise are shared and that the team collectively grows stronger and more capable.

By focusing on these three competency groups—big-picture thinking, execution, and leveling up—architects can significantly enhance their impact on organizational performance. This structured approach to competency development can ensure that architects are skilled practitioners and strategic leaders capable of driving meaningful change and innovation within their organizations.

5.1.1: Big-Picture Thinking

Architects are often the only individuals within an organization who possess a "**helicopter view**," allowing them to oversee vast domains and anticipate the broader consequences of decisions.

image by guvendemir from istock

This unique perspective positions them as crucial big-picture thinkers who can contribute to the organization in several significant ways:

- **Identifying High-Leverage Points for Maximum Impact**: With their ability to see the big picture, architects can pinpoint areas within the organization where small changes or strategic investments can yield substantial benefits. This capability allows them to recommend and implement solutions that maximize impact, ensuring resources are allocated efficiently and effectively. By focusing on high-leverage points, architects can drive significant improvements in performance and productivity across the organization.
- **Helping Others to See the Big Picture**: Architects can facilitate a broader understanding of the organizational landscape among their colleagues. By helping others see the big picture, they enable more informed decision-making at all levels. They can create tools and frameworks, such as a comprehensive Lightweight Architectural Analytics, that promote big-picture thinking. These tools help team members understand how their work fits into the larger organizational context and the impact of their contributions. Through workshops, presentations, and collaborative tools, architects can disseminate their vision, fostering a culture of strategic awareness.
- **Zooming In and Out**: Good architects have the rare ability to both maintain a strategic overview and delve into implementation

details when necessary. This flexibility allows them to bridge the gap between high-level strategy and on-the-ground execution. They can provide strategic guidance while also engaging deeply with technical teams to ensure that the finer details align with broader organizational goals. This dual capability ensures that strategic initiatives are feasible and that practical implementations stay aligned with the overall vision.

- **Rooting Out Inefficiencies**: Using their big-picture perspective, architects are adept at identifying and eliminating inefficiencies within organizational processes and systems. They can lead the adoption of new technologies and processes that enhance the efficiency and effectiveness of multiple teams. By doing so, architects can help streamline operations, reduce redundancies, and improve overall performance. This proactive approach to efficiency ensures that the organization remains competitive and can adapt to changing market conditions and technological advancements.

Architects play a pivotal role in organizations by leveraging their helicopter view to identify high-leverage points, help others understand the big picture, seamlessly transition between strategic and detailed thinking, and root out inefficiencies. Their ability to see and act on the broader context makes them invaluable in driving organizational success and fostering a culture of strategic innovation.

5.1.2: Execution

As execution-focused practitioners, architects must not only deliver tangible results but also enhance collaboration across the organization. Their ability to combine technical expertise with **pragmatic approaches** and foster **effective teamwork** is crucial for achieving these goals.

image by laylabird from istock

Here's how architects can excel in these areas:

5.1.2.1: Delivering Results with Pragmatism

Architects must blend their technical expertise with practical approaches to ensure their solutions are **impactful and feasible**. By prioritizing practicality and feasibility, architects can ensure their contributions lead to tangible improvements and successful outcomes:

- **Create Meaningful Solutions**: Architects should focus on developing solutions that are practical and impactful rather than getting caught up in theoretical ideals and models. These solutions must address real-world problems and be implementable within the current organizational context. By prioritizing practicality, architects ensure that their contributions lead to tangible improvements and benefits.
- **Break Down Complex Problems**: To deliver impactful results, architects must be skilled at deconstructing complex issues into manageable components. This approach allows teams to tackle problems step-by-step, reducing overwhelm and increasing the likelihood of successful outcomes. By using this divide-and-conquer

approach, architects help ensure that projects progress smoothly and deliver the desired impact.

- **Craft Pragmatic Plans**: When planning projects, architects must consider various constraints, including technical limitations, logistical challenges, and organizational dynamics. Pragmatic planning involves balancing ideal solutions with what is feasible given these constraints. This approach ensures that plans are realistic, actionable, and more likely to succeed in the real world.

5.1.2.2: Enhancing Collaboration

Architects can play a crucial role in fostering **effective collaboration** and creating alignment within their teams and across the broader organization. By engaging meaningfully with various teams and departments, architects build trust and enhance execution speed, leading to better overall productivity and execution.

- **Creating Alignment**: Architects can be key in creating alignment within their teams and the broader organization. This role involves ensuring all stakeholders understand the strategic objectives and how their work contributes to these goals. Clear communication and shared understanding help align efforts and focus resources on common objectives.
- **Improving Collaboration**: Architects can be critical catalysts for improving collaboration within their teams and across different groups in the organization. This role facilitates communication, coordinates efforts, and resolves conflicts to ensure smooth cooperation. By fostering a collaborative environment, architects can help teams work more effectively together, leading to better execution and faster results.
- **Collaborating Meaningfully Across Groups**: Architects should actively engage with various teams and departments to build trust and improve execution speed. Meaningful collaboration involves understanding the needs and perspectives of different groups and finding ways to integrate their efforts. Building trust through consistent and open communication can help to break down silos and enhances overall productivity.

As execution-focused practitioners, architects must blend their technical skills with a pragmatic approach to deliver meaningful solutions and break down complex problems. They must also play a vital role in creating alignment and improving collaboration within and across teams. By doing so, they not only achieve impactful results but also enhance the overall efficiency and effectiveness of the organization.

5.1.3: Leveling Up

Architects are often seen as leaders and role models who can help organizations raise the bar on both technical and cultural fronts.

image by sanjeri from istock

This role I categorize into three key areas: citizenship, design and architecture, and software engineering.

5.1.3.1: Citizenship

Architects should look beyond their immediate responsibilities and work to elevate practices and behaviors across their organizations:

- **Contribute to the broader technical community** through tech talks, education, publications, and open-source projects. This involvement helps spread knowledge and innovation.
- **Extend their influence** beyond their organization to make an impact on the industry at large, sharing insights and best practices that benefit a wider audience.
- **Lead efforts** to solve **significant problems** in their areas, demonstrating leadership and commitment to continuous improvement.
- **Elevate the engineering culture** within the company, fostering an environment of excellence and continuous learning.

5.1.3.2: Design and Architecture

As leading authorities on systematic and strategic design, architects play a critical role in shaping the architecture of their organizations:

- Leverage their deep domain knowledge to **improve the definition of best practices** in design and architecture, ensuring that standards are maintained and enhanced over time.
- **Identify and solve systemic architectural problems** by quickly recognizing issues and articulating possible solutions. Their ability to address these challenges ensures system stability and scalability, ensuring high-quality overall architecture.

5.1.3.3: Software Engineering

Architects also need to stay deeply connected to software engineering practices, leveraging their experience to enhance technical execution:

- **Promote and demonstrate best-in-class practices** in coding, documentation, testing, and monitoring. By setting high standards, they can ensure that quality and efficiency are prioritized.
- Solve **challenging technical and execution problems** that few others can, using their expertise to tackle issues that require advanced skills and innovative thinking. Architects' influence can motivate others to develop new advanced skills.

By excelling in these areas, architects can significantly contribute to raising their organizations' technical and cultural standards, leading by example, and driving continuous improvement across all levels.

5.2: Questions to Consider

- *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 could you have taken to improve collaboration and build trust within your organization?*
- *Have you contributed to the broader technical community through tech talks, education, publications, open-source projects, etc.?*
- *How could you help solve significant 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 would you promote and demonstrate best-in-class practices in coding, documentation, testing, and monitoring?*

6: Leadership Traits

image by niserin from istock

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 Netflix's valued behaviors.
- Marquet focused on leadership and organizational management, particularly emphasizing the principles of Intent-Based Leadership.
- Borrowing from Netflix's original values, I see the following behavioral traits as crucial for architects: communication, judgment, impact, inclusion, selflessness, courage, integrity, curiosity, innovation, and passion.

"*A leader is anyone who takes **responsibility** for **recognizing** the **potential in people and ideas**, and has the **courage** to develop that potential.*"
—Brené Brown

My approach to architecture leadership draws inspiration from two standout sources: **David Marquet's**[1] **leadership principles**, articulated in his book "Turn the Ship Around![2]" and "Leadership Is Language[3]," and Netflix's valued behaviors. Marquet's ideas are about empowering team members, providing clarity, decentralizing decision-making, striving for continuous improvement, and practicing servant leadership. Meanwhile, **Netflix's valued behaviors**[4] offers a compact masterclass in coaching and developing people's leadership traits. Together, these resources form a robust framework for leading with insight and influence in the IT architecture (Figure 1).

[1] https://davidmarquet.com/
[2] https://davidmarquet.com/turn-the-ship-around-book/
[3] https://davidmarquet.com/leadership-is-language-book/
[4] https://jobs.netflix.com/culture

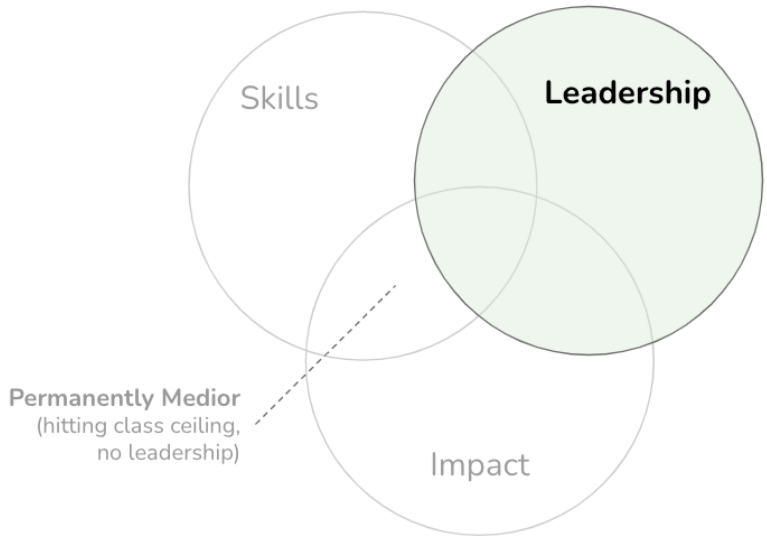

***Figure 1**: Leadership is one of the three main elements of being an architect (skills, impact, leadership). Having skills and impact without leadership frequently leads to hitting a glass ceiling. Such architects plateau at an intermediate level and cannot direct the company to innovative or transformative solutions.*

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

Marquet's work is closely tied to **the Leader-Leader model of leadership**, a leadership 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 should 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 ownership and accountability for the team's success. This model also **helps build trust and more robust team relationships**, increasing productivity, creativity, and innovation.

image by caiaimage/martin barraud from 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 2). 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.

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 can expect to **take the initiative and lead efforts**. In addition, this phase captures the desired interaction of the teams with whom architects work, expecting teams to take the initiative and share their intentions, which architects can support and level up.

The **Leader-Leader model** can and has been applied effectively in software engineering and IT architecture. Here are some examples:

- **Empowering Teams**: In Agile environments, authority is often distributed among team members, embodying the Leader-Leader

model. Instead of a project manager or lead developer dictating every detail, team members collaborate closely, take ownership of specific tasks, and make decisions within their areas of expertise. This approach empowers developers, testers, and other team members to contribute actively, promoting creativity and innovation.

- **Cross-functional Collaboration**: In a DevOps environment, the boundaries between development and operations are blurred. Teams are empowered to take end-to-end ownership of the product lifecycle, from development to deployment and monitoring. This shared responsibility encourages collaboration and reduces the reliance on a single decision-maker, aligning closely with the Leader-Leader model.
- **Continuous Improvement**: DevOps emphasizes continuous integration, continuous deployment, and continuous feedback. Teams are encouraged to experiment, learn from failures, and improve processes. Leaders in DevOps act as coaches, guiding teams to identify opportunities for improvement and innovation, rather than directing every action.
- **Mentoring and Coaching**: Senior architects often take on the role of mentors and coaches rather than top-down decision-makers. They guide less experienced team members through complex architectural challenges, fostering a culture of learning and development. This approach helps build a strong, capable architecture team that can independently contribute to the organization's success.
- **Community-Driven Leadership**: Open source projects often operate under the Leader-Leader model. Contributors from around the world collaborate on projects, and decision-making is distributed across the community. Leaders in open source projects often emerge organically, based on their contributions and the respect they earn from others, rather than through formal authority.
- **Collaborative Problem-Solving**: In open-source communities, problems are solved through collaboration and shared knowledge. Contributors propose solutions, review each other's code, and collectively decide on the best approaches. This inclusive and collaborative environment encourages innovation and creativity, reflecting the principles of the Leader-Leader model.
- **Decentralized Control**: In a microservices architecture, different teams manage different services independently. Each team is responsible for the full lifecycle of its services, from design to

deployment and maintenance. This autonomy aligns with the Leader-Leader model, where each team is empowered to make the best decisions for their service while collaborating with other teams to ensure the overall system's success.

- **Service Ownership**: Teams practicing microservices architecture are encouraged to adopt a "you build it, you run it" philosophy. This ownership means they develop the services and operate and maintain them in production. This ownership fosters accountability and drives teams to improve their services continuously.

In all these examples, the Leader-Leader model promotes a collaborative, inclusive, and empowering work environment. Individuals are encouraged to take ownership of their work and contribute to the collective success of the team or organization. This approach enhances productivity and fosters innovation, creativity, and continuous improvement.

Figure 2: *Leadership language. Based on Intent-Based Leadership, by David Marquet.*

6.2: Netflix's Valued Behaviors: Leadership Behaviors

The Netflix overview of their valued behaviors[5] is 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.

image by chaiyon021/martin barraud from istock

6.2.1: 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

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

- 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

6.2.2: 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**
- 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

6.2.3: Impact

As discussed in the Impact section, the architect's work 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 processes**
- You demonstrate **consistently** strong performance so **colleagues can rely upon you**

6.2.4: Inclusion

Architects must 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

6.2.5: 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 a 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

6.2.6: 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 the organization's values
- You **can be vulnerable** in search of truth

6.2.7: Integrity

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

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

6.2.8: 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 an 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

6.2.9: Innovation

More than curiosity is required. To make an impact as an architect, you need to create helpful 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**

6.2.10: 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**

6.3: Questions to Consider

- *Reflect on the Leader-Leader model of leadership model 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?*
- *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?*
- *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?*

7: Leading with Language: Six Plays for Grounded IT Architects

image by jacoblund from istock

IN THIS SECTION, YOU WILL: Understand how IT Architects can transform their effectiveness and foster a more collaborative, innovative, and grounded practice by consciously applying David Marquet's six linguistic leadership plays to better navigate the complexities of modern technology environments.

KEY POINTS:

- IT Architects can significantly enhance their leadership and influence by adopting six key communication "plays" from David Marquet's "Leadership is Language," moving beyond outdated Industrial Age command-and-control styles.
- These plays—Control the Clock, Collaborate, Commit, Complete, Improve, and Connect—provide practical linguistic tools to foster better thinking (Bluework), more effective execution (Redwork), and stronger team engagement.
- Applying these plays helps architects build trust, flatten power gradients, encourage psychological safety, and unlock discretionary effort, leading to more robust architectural solutions and greater buy-in.
- The principles align directly with a "Grounded Architecture" approach by promoting data-driven decisions, collaborative networks, adaptability, and strategic impact within organizations.
- By consciously changing their language, architects can cultivate a more innovative, resilient, and effective engineering culture.

Architects are not just technical designers; they have transformed into **strategic leaders**, and **facilitators of collaboration**, all essential for navigating complexity** and driving business value. This expanded role necessitates a **sophisticated approach to leadership** that moves away from traditional **"command and control" models**. These outdated paradigms, rooted in the Industrial Age, are ill-suited for the **dynamic, knowledge-driven environment** we face in modern IT, where architects often **lead by influence rather than direct authority**. In this context, I believe communication is not just a skill—it is the primary tool of leadership.

One resource that resonates with my experiences is L. David Marquet's seminal work, **"Leadership is Language."** As a former U.S. Navy Captain, Marquet illustrates how the **words leaders use** profoundly shape **team culture**, **effectiveness**, and **overall success**. His argument—that **subtle shifts in language** can catalyze **significant improvements** in

team performance and well-being—**challenges entrenched leadership paradigms** and offers what he calls a "**New Playbook**" for contemporary challenges. From my perspective, the core of this new playbook lies in recognizing that the nature of work has changed. Industrial Age leadership focused on **maximizing efficiency** and ensuring **compliance** for predominantly physical, repetitive tasks. However, the field of IT architecture requires **complex**, **cognitive**, **and collaborative teamwork**. This type of work thrives not on **mere compliance**, which often results in minimal effort, but on **commitment**, which unlocks discretionary effort and innovation. I've seen firsthand that applying **outdated leadership language** to architectural work can inadvertently stifle creativity and engagement. Organizations that fail to adapt their leadership language within their architectural practices may struggle with **innovation**, **adaptability**, and the ability to attract and retain top talent.

The principles in "**Leadership is Language**" resonate with my approach to "**Grounded Architecture**." Marquet's six leadership "**plays**" provide **practical linguistic tools** that align perfectly with these tenets. They empower architects to foster **clearer communication**, enable **deeper thinking**, and build **stronger**, **more committed teams**, helping us become more truly "**grounded**." A critical, and often invisible, barrier that I've observed IT architects must consciously address is the "**power gradient**"—the perceived **hierarchical distance** between individuals. I've found that steep power gradients can **stifle creativity**, **suppress valuable input**, and **hinder open communication**, all of which are detrimental to effective architectural practices. Architects frequently navigate complex organizational structures where these gradients are prevalent. If we don't **manage these dynamics through intentional language**, true collaboration—a cornerstone of Grounded Architecture—**cannot flourish**. Marquet's plays offer concrete strategies to **flatten these gradients** and promote **more inclusive and effective interactions**, which I believe are essential for our success.

7.1: Setting the Stage: From "Redwork/Bluework" to Effective Action

Before exploring specific plays, it's essential to grasp a fundamental concept from Marquet's framework: the distinction between "Redwork" and "Bluework." This distinction clarifies the different operational modes within any team and highlights the importance of balancing them for optimal performance.

Redwork refers to the "**doing**" **or** "**execution**" phase of any project. It emphasizes performance, **efficiency**, and **minimizing variability** to achieve a specific outcome. In the context of IT architecture, Redwork includes activities such as coding a proof-of-concept based on a defined specification, meticulously documenting a finalized architectural design, or implementing a system according to a detailed plan. The primary focus during Redwork is on execution and **achieving predetermined goals**.

Bluework, on the other hand, represents the "**thinking**" **or** "**decision-making**" phase. This mode **embraces variability** and is characterized by reflection, planning, strategic problem-solving, and collaboration. For IT architects, Bluework is their natural environment; it encompasses strategic design sessions, the analysis of various technological options, the evaluation of emerging technologies, and collaborative workshops aimed at addressing complex architectural challenges.

Marquet stresses the critical importance of **oscillation** between these two modes. Effective teams and their leaders must consciously and deliberately shift between periods of Redwork and Bluework. Being entrenched in one mode while neglecting the other can be detrimental. For instance, excessive Bluework can lead to "analysis paralysis," while an overemphasis on Redwork without sufficient Bluework can result in rushed and poorly conceived solutions.

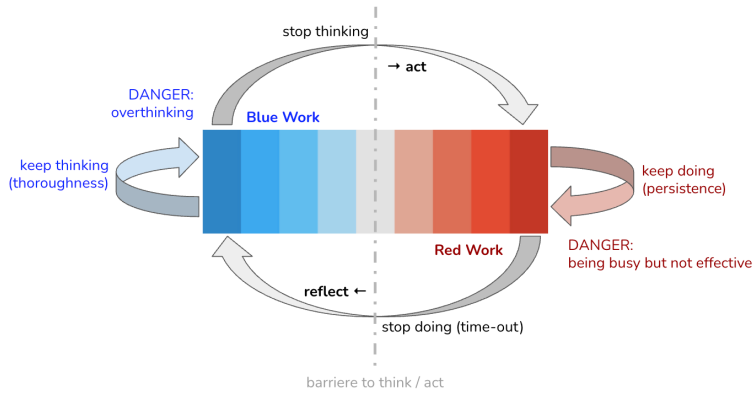

***Figure 1**: The dynamic between "Blue Work" (thinking/deciding) and "Red Work" (doing/executing), and the importance of consciously switching between the two modes to lead more effectively. This model encourages a conscious, rhythmic cycle of thinking and doing, where teams pause to reflect, adjust, and act deliberately, breaking the habit of reflexive execution or analysis.*

This Redwork/Bluework framework is profoundly relevant for IT Architects. Their role inherently demands significant time dedicated to Bluework—strategic thinking, careful design, and thorough problem analysis are the bedrock of sound architecture. However, the **relentless pressure for rapid delivery** in many IT environments can inadvertently **push teams, including architects, into a state of perpetual Redwork**, sacrificing crucial thinking time. Recognizing the current "work mode" of the team allows architects to choose the appropriate language and leadership play to either facilitate deeper thinking or drive effective execution.

Failing to distinguish and manage the balance between Redwork and Bluework often leads to common architectural pitfalls. Symptoms like "**analysis paralysis**," where teams become stuck in endless deliberation (excessive Bluework), or "**rushed, flawed implementations**," which stem from inadequate thinking and premature execution (insufficient Bluework before Redwork), are frequent occurrences. Marquet's plays, such as "Control the Clock" to shift into Bluework or "Commit" to transition into Redwork, offer the necessary mechanisms for effectively navigating these modes.

Moreover, Marquet's framework suggests a **democratization of Bluework**; it should not be the exclusive domain of architects or senior leadership. This perspective challenges traditional hierarchies that may portray architects as the sole "thinkers." For the "Grounded Architecture" approach, which advocates for collaborative networks, architects should use language that actively invites all team members into Bluework activities, such as design sessions or problem-solving workshops. Such **inclusivity enriches the decision-making process** by incorporating diverse perspectives and fostering a shared sense of ownership.

7.2: The Six Leadership Plays in the Architect's Arena

Marquet outlines six specific "plays" that leaders can use to transform their communication and, consequently, their team's performance. These plays offer a new language for leadership, moving away from outdated Industrial Age scripts. Figure 1 illustrates a cyclical framework depicting how IT Architects can enhance their effectiveness and foster a collaborative, innovative, and grounded practice through the intentional application of David Marquet's six linguistic leadership plays:

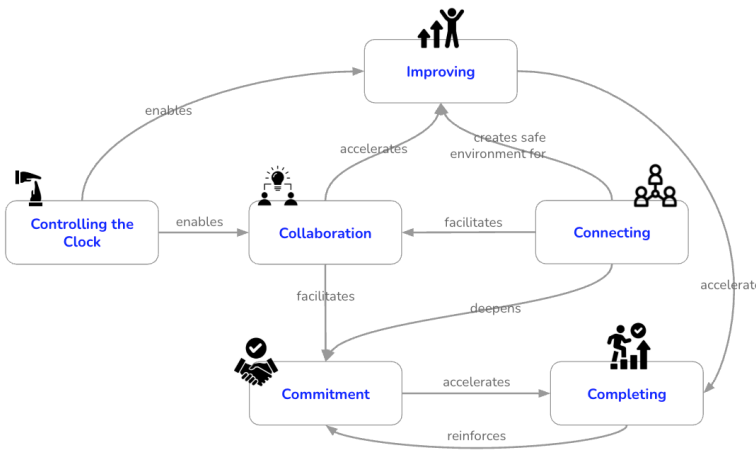

Figure 2: *David Marquet's six linguistic leadership plays and their key interdependencies.*

The following table provides a concise overview of these plays and their relevance for IT Architects:

Play	Shift From	Key Benefit for IT Architects	Key Marquet Phrase/-Concept
Control the Clock	~~Obeying the clock~~	Enables strategic pauses, better decisions, reduces errors	"Make a pause possible," "Shift to Bluework"
Collaborate	~~Coercing~~	Leverages collective wisdom, fosters innovation, builds buy-in	"Let the doers be the deciders," "Vote first, then discuss"
Commit	~~Complying~~	Drives ownership, unlocks discretionary effort, ensures follow-through	"Commit to learn, not just do," "Discretionary effort"
Complete	~~Continuing indefinitely~~	Provides closure, facilitates learning, focuses on outcomes	"Chunk it small," "Celebrate success"
Improve	~~Proving ability~~	Cultivates a learning culture, continuous enhancement, reduces fear	"What can we learn?" "How can we make it better?"

Play	Shift From	Key Benefit for IT Architects	Key Marquet Phrase/-Concept
Connect	~~Conforming to roles~~	Builds trust, encourages psychological safety, authentic engagement	"Flatten power gradients," "Trust first"

7.2.1: Play 1: Control the Clock, Don't Obey the Clock

The "Control the Clock" play focuses on the intentional act of **pausing action** (Redwork) to allow space for **thinking, reflection, and decision-making** (Bluework). It challenges the pervasive pressure to **constantly "do"** and promotes **strategic thinking**. Marquet emphasizes that **leaders have a responsibility** not only to make these **pauses possible** but also to **call for them**, especially when teams are deeply immersed in Redwork and may not recognize the need to pause. For instance, language such as, "**We have time to do this right, not twice**," can signal that a pause is not only acceptable but encouraged.

image by peopleimages from istock

IT architects often work under **significant pressure to deliver solutions quickly**. The "Control the Clock" play empowers them to carve out crucial moments for strategic architectural reviews. This ensures **alignment with broader goals** and helps **prevent costly errors** that may arise from rushed decisions. This play is instrumental in helping architects avoid the common trap of "**solutioneering**," which is **jumping to solutions before fully understanding the problem** or exploring alternative approaches — a frequent occurrence in fast-paced IT environments.

Examples:

- **Scenario 1 (Design Phase)**: An architect leading the design of a critical new system notices the team converging prematurely on a specific technology choice. They can "control the clock" by saying, "Let's pause the 'how' for a moment. Are we all aligned on the 'what' and 'why'? What are the 2-3 core problems this specific component needs to solve?" This language shifts the team from Redwork (selecting a tool) back into Bluework (clarifying requirements and problem definition).
- **Scenario 2 (Incident Response)**: During a significant system outage, instead of merely issuing directives, an architect might call for a brief "Bluework huddle": "Okay team, let's pause the

immediate fixes for ten minutes. What do we know for sure? What are our top two hypotheses for the root cause? What is the safest next diagnostic step we can take?" This practice of calling a time-out, even when not explicitly demanded by the situation, helps normalize the act of pausing.

- **Scenario 3 (Agile Context)**: An architect can proactively incorporate "architectural reflection" slots into sprint planning or review meetings. This ensures dedicated time for Bluework regarding upcoming epics or addressing accumulated technical debt. For example: "Before we commit to these user stories for the next sprint, let's spend 30 minutes discussing the architectural implications of Feature X and any potential long-term impacts."

Effectively "Controlling the Clock" serves as a **crucial prerequisite** for genuine "**Collaboration**" (Play 2) and meaningful "**Improvement**" (Play 5). Without the **intentional pause** created by this play, there is simply no space for **diverse opinions**, **thorough discussions**, or **valuable learning** to take place. If architects do not consciously create these moments for reflection, discussions are likely to be rushed, **dominant voices may overshadow others**, and true collaboration will remain elusive. **Improvement requires reflection**, which is only possible during such pauses. Thus, architects who master this play unlock the potential of several other vital leadership practices.

For architects, this play is also a key mechanism for ensuring their work remains grounded in **strategic objectives** and **data-driven insights**, rather than being swept away by the tide of **short-term project momentum**. **Rushing (obeying the clock)** often leads to **cutting corners** on essential data gathering or strategic thinking. By "controlling the clock," architects create opportunities to **review data**, **consult stakeholders**, and ensure architectural choices **align with broader organizational goals**. This practice helps prevent the "**Ivory Tower" architect syndrome**, where architects become disconnected from practical realities and the needs of the organization.

7.2.2: Play 2: Collaborate, Don't Coerce

The "Collaborate, Don't Coerce" play promotes **genuine collaboration** by actively **inviting and valuing diverse perspectives** rather than allowing

leaders to **push their own agendas**—whether subtly or overtly. A central tenet of this approach is to "**let the doers be the deciders**," empowering those **closest to the work** to make meaningful contributions to decisions. Key techniques to foster this collaboration include "**vote first**, **then discuss**," which helps prevent the group from **anchoring on the leader's opinion**. Leaders are encouraged to **speak last**, after listening to others, and to **cultivate genuine curiosity** about dissenting viewpoints by asking questions like, "What do you see that we don't?" Creating an environment of **psychological safety**, where individuals feel comfortable sharing their honest thoughts without fear of judgment, is essential for the success of this play.

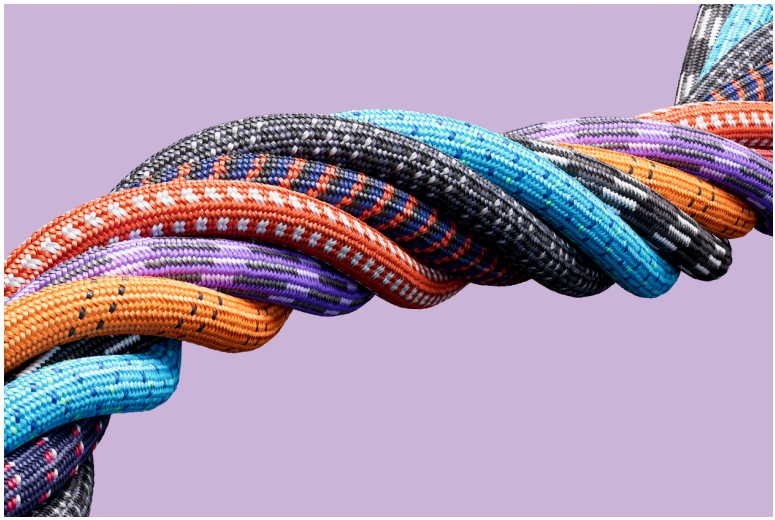

image by olivier le moal from istock

Architects need **buy-in and active participation** from a **diverse array of stakeholders**, including developers, product managers, operations teams, and business units. While coercion might yield superficial compliance, **genuine collaboration fosters deeper commitment**, more robust solutions, and **shared ownership**. Complex architectural problems, which are common in modern IT, greatly benefit from **cognitive diversity**; collaboration is key to harnessing this collective intelligence.

Examples:

- **Scenario 1 (Technology Selection):** An architect is tasked with selecting a new messaging queue technology for the organization. Instead of stating their initial preference, they present the problem, outline the selection criteria, and then ask team members to independently write down their top one or two choices along with their reasoning ("vote first"). Following this, the architect facilitates a discussion, ensuring all voices are heard by asking clarifying questions and exploring different perspectives. They provide their own assessment only after everyone else has had a chance to speak.
- **Scenario 2 (Design Review):** During a review of a proposed microservice architecture, a junior engineer expresses a concern about potential data consistency issues. Instead of dismissing this concern or providing an immediate rebuttal, the architect responds with curiosity: "That's an interesting point. Tell me more about that. What specific scenarios are you envisioning where consistency might become an issue? What do you see that the rest of us might be missing?" This approach validates the contribution and encourages deeper exploration.
- **Scenario 3 (Architectural Principles Co-creation):** An architect facilitates a workshop with lead engineers from various teams to define a new set of architectural principles for the company. They provide a guiding framework and some initial examples but actively encourage the team to generate, debate, and refine the principles themselves. The architect acts as a facilitator and guide rather than a dictator, asking questions like, "Our goal is to co-create these guiding statements. What are the most critical principles we need to ensure consistency, scalability, and maintainability for our platform moving forward?"

The ongoing trend toward **distributed systems**, and **cross-functional agile teams** in IT makes the "Collaborate, Don't Coerce" play increasingly vital. Architects operating in such environments cannot effectively **dictate solutions from above**; their success relies on their ability to **facilitate**, **integrate**, and **harmonize diverse technical expertise and perspectives**. An architect cannot be the foremost expert in every component of a complex, distributed system, so their capacity to **collaborate effectively with specialists**—rather than **coercing them into a singular, preconceived vision**—becomes crucial for architectural quality and adoption. This play is a direct enabler of leveraging the "collaborative networks" central to the Grounded Architecture philosophy.

However, a common pitfall for architects is "**disguised coercion**"—believing they are collaborating when, in fact, they are **subtly steering conversations** toward their preferred outcome. Architects, often being senior and highly experienced, might **unintentionally coerce** through the **strength of their opinions**, the **way they frame questions**, or by **selectively amplifying certain viewpoints**. This play requires **genuine humility** and a **sincere willingness to be proven wrong** or to adopt a solution that differs from one's initial inclination. For some architects, this represents a significant mindset shift, moving away from traditional top-down approaches.

7.2.3: Play 3: Commit, Don't Comply

The "Commit, Don't Comply" play emphasizes obtaining **genuine**, **internally motivated commitment** from the team, which is essential for **unlocking discretionary effort**. This approach contrasts with **mere compliance**, which usually results in only the **minimum effort** necessary to meet requirements. **Commitment is an active choice** made by individuals and is often nurtured through **true collaboration** during the decision-making process. This play encourages **commitment to learning** (rather than executing tasks blindly) and **commitment to actions** (even when individual beliefs or preferences do not fully align with the chosen path). The linguistic shift from *"I can't"* (implying an external force and fostering compliance) to *"I don't"* (indicating internal resolve and commitment) illustrates this principle.

image by jacob wackerhausen from istock

Architectural standards, **patterns**, and **strategic decisions** are only effective if development teams **genuinely commit** to their adoption and implementation. **Compliance** often leads to **superficial adherence**, **workarounds**, or even eventual **abandonment**. Significant architectural changes, such as **platform modernization** or the **adoption of new paradigms**, require **deep and sustained commitment** from engineering teams to navigate inevitable challenges and successfully complete the initiative.

Examples:

- **Scenario 1 (Adopting a New Standard)**: After a collaborative process (as described in Play 2) to select a new API security standard, the architect seeks explicit commitment from the involved teams. They might say, "We've thoroughly discussed our options and collectively chosen this standard. What support do you need from the architecture team and from each other to fully commit to implementing this standard for all new services moving forward? What potential roadblocks can we anticipate now and plan for together?" This language positions the adoption as a shared goal and responsibility.

- **Scenario 2 (Proof of Concept)**: An architect initiates a Proof of Concept (PoC) for a new, potentially transformative technology. Instead of merely assigning tasks, they frame the initiative to encourage a commitment to learning: "Our primary goal for this PoC is to *learn* whether this technology can effectively solve X problem for us and to gain a clear understanding of its operational complexities and integration challenges. Let's commit to these specific learning objectives and the actions required to achieve them over the next two weeks."
- **Scenario 3 (Decision Disagreement)**: A team has decided on an architectural approach that the architect has some reservations about, though it's not a critically flawed decision. The architect might express their support by saying, "While I see some potential challenges with this path, the team has made a strong case and collectively decided to proceed. I commit to supporting your decision and will help you succeed in its implementation. Let's agree on the key actions, milestones, and check-in points to ensure we can address any issues that arise." This shows commitment to the team's chosen action, even if the architect's personal belief isn't absolute.

True **commitment** often results from effective **collaboration**. Attempts by architects to secure commitment without first engaging in **genuine, inclusive collaboration** are likely to yield, at best, **superficial compliance**. If architects **skip or poorly execute the Collaborate play**—for instance, by **subtly coercing the team toward a predetermined solution**—team members will not feel a **sense of ownership** over the decisions made. Therefore, their adherence will be driven by compliance (*"I'm doing this because the architect said so"*) rather than true commitment (*"I'm doing this because I believe in its value, understand its rationale, and had a meaningful say in the decision"*). This significantly impacts the **quality**, **sustainability**, and **ultimate success** of architectural implementations.

The "Commit, Don't Comply" play is also vital for the effective functioning of **adaptable governance models** (*nudging, taxation, mandates*) as described within the **Grounded Architecture** framework. While **mandates** represent a **top-down enforcement of compliance**, mechanisms like **nudging** and **taxation** rely more on **influencing behavior** and **fostering commitment** rather than imposing strict adherence. These softer forms of governance aim to **guide choices** by making desirable

architectural behaviors easier or more attractive, requiring a degree of **voluntary buy-in** or **commitment** from the teams.

7.2.4: Play 4: Complete, Don't Continue

The "Complete, Don't Continue" play challenges the Industrial Age mindset of continuous, undifferentiated, and often unending work. It emphasizes **breaking down tasks** into **defined**, **manageable chunks** with **clear completion points**. A key aspect of this play is the importance of **celebrating successes** upon completion and **learning from each finished cycle**. The principle of *"Chunk it small, but do it all"* is particularly relevant when dealing with **high levels of uncertainty or complexity**. This principle involves **explicitly deciding when to stop** a particular phase of work and having agreed-upon stopping criteria or **definitions of "done"**.

image by photo_concepts from istock

Architectural initiatives—such as **platform migrations**, **system redesigns**, or the **rollout of new enterprise-wide standards**—can be **large**, **complex**, **and long-running**. Breaking these substantial undertakings into **smaller**, **completable phases or milestones** is crucial

for **maintaining momentum**, providing opportunities for **iterative learning and adjustment**, and **fostering a sense of accomplishment** within the team. This play helps architects avoid *"architectural drift,"* where designs are endlessly refined without delivering tangible value, or *"analysis paralysis,"* which can **stall progress indefinitely**.

Examples:

- **Scenario 1 (Platform Modernization)**: An architect leading a multi-year platform modernization initiative works with teams to define clear, completable stages with measurable outcomes. For instance: "Stage 1: Migrate User Authentication Service to the new microservices platform. Target completion: End of Q2. Success criteria: 100% of user authentication requests handled by the new service with improved performance metrics." Upon successful completion, the team celebrates this milestone.
- **Scenario 2 (Design Spike)**: When faced with a complex new feature requiring significant architectural design, an architect initiates a time-boxed design spike (e.g., one week) with a specific, completable goal: "By the end of this week, we will have a documented decision on the data storage strategy for Feature Y, including a comparative analysis of the top two alternatives and the rationale for our choice. This document will represent 'complete' for this design spike."
- **Scenario 3 (Architectural Debt Reduction)**: Instead of pursuing a vague and potentially demoralizing goal like "reduce overall technical debt," an architect collaborates with engineering teams to identify specific, completable pieces of technical debt to address each quarter. The successful refactoring or elimination of these items is then acknowledged and celebrated. For example: "This quarter, our focus is to complete the refactoring of the Legacy Billing Module's outdated interface, thereby eliminating a significant source of maintenance overhead."

The "Complete" play creates **defined endpoints** for work increments and incorporates **celebrations of achievement**, directly boosting **team morale** and **reinforcing positive behaviors**. This, in turn, creates a **virtuous cycle**, making future **commitment** (Play 3) easier to achieve. When **successes are recognized and celebrated**, it reinforces the **specific behaviors** that led to those successes, allowing individuals to feel a

tangible sense of accomplishment. This **positive momentum** and the **trust it builds** make teams more likely to **commit enthusiastically** to the next defined chunk of work.

Furthermore, this play is vital for **demonstrating tangible progress** and **showcasing the value delivered** by the architecture function. This aligns directly with the Grounded Architecture principle, which emphasizes the need to **show impact** and **execute effectively at scale**. Architecture can sometimes be perceived as an **abstract**, **slow-moving discipline**, leading to accusations of architects operating in an *"Ivory Tower,"* disconnected from **practical delivery**. The **Complete** play focuses on delivering defined architectural outcomes in **manageable chunks**, making the **value of architecture visible**, **measurable**, **and timely**. This helps justify architectural investments, builds credibility for the architecture practice within the organization. Additionally, it provides **clear points** at which *"Lightweight Architectural Analytics"* can be applied to **measure progress**, **impact**, and **adherence to architectural goals**.

7.2.5: Play 5: Improve, Don't Prove

The "Improve, Don't Prove" play advocates for a fundamental shift in mindset—moving away from an environment where individuals feel the need to **constantly prove their competence** and the **infallibility of their decisions**. Instead, it fosters a collective focus on **improving outcomes**, **processes**, and **shared learning**. This play promotes a **culture of curiosity and reflection**, encouraging questions such as *"How could we make this better?"* or *"What can we learn from this experience?"* It develops a **learning culture** where mistakes and setbacks are seen not as reasons for blame or shame, but as **valuable opportunities for growth and refinement**. Leaders embody this mindset by **openly reflecting** on their actions and considering how they could have been improved.

image by sergei chuyko from istock

Architecture is inherently an iterative and evolutionary process; initial designs are rarely perfect and must **adapt to new information**, **changing requirements**, and **feedback from implementation**. An improve mindset allows **architectural solutions to evolve and adapt effectively**. This play is also crucial for **fostering psychological safety** within teams, encouraging engineers and other stakeholders to **identify potential issues or weaknesses** in architectural designs or decisions without fear of retribution or of appearing incompetent.

Examples:

- **Scenario 1 (Post-Incident Review)**: After a significant system failure traced back to an unforeseen architectural flaw, the architect leads a blameless post-mortem. The focus is entirely on learning and improvement: "What can we *learn* from this incident as a team? How can we *improve* our design process, review mechanisms, or monitoring capabilities to prevent similar issues in the future?"
- **Scenario 2 (New Technology Adoption)**: An architect leads the introduction of a new software framework in the organization. Rather than demanding immediate perfection or creating pressure to prove the framework's viability, they frame the initiative as an opportunity for improvement: "Let's pilot this new framework on

Project X. Our main goal is to *improve* our development velocity and code quality for this type of service. What metrics will help us understand if we're achieving that improvement? What challenges are we facing, and how can we adapt our approach or the framework's configuration to overcome them?"

- **Scenario 3 (Architect Self-Reflection)**: An architect shares a learning moment with their team, modeling vulnerability and an "improve" mindset: "Looking back at the initial design for System Z, I realize I didn't fully account for the long-term scalability needs under peak load conditions. That was an oversight on my part. How can *we*, as a team, build in more robust scalability checks and forecasting earlier in our design process next time to avoid this?"

The "Improve, Don't Prove" mindset is fundamental to creating and sustaining a **culture of psychological safety** within technical teams. Without this safety, team members are more likely to **hide mistakes**, **downplay concerns**, or **avoid flagging potential architectural flaws**—all to avoid the risk of **appearing incompetent** or **being blamed** for problems. When the environment prioritizes **proving**, individuals naturally become **risk-averse and defensive**, which **stifles learning and innovation**.

Architects who **champion the improve mindset** actively **cultivate the safety** needed for **open communication** about what is not working. This dialogue is critical for **identifying and addressing architectural weaknesses** before they escalate into major issues. As a **significant cultural enabler**, this play supports the core themes of **adaptability** and **continuous learning** central to the Grounded Architecture philosophy. Grounded Architecture is designed for fast-moving global organizations and emphasizes the need for adaptability in architectural practices and solutions.

The "Improve, Don't Prove" play **institutionalizes the process of learning and adaptation**. By consistently asking, *"How can we make it better?"*, architects ensure that both the architecture and the surrounding practices are **continuously refined and enhanced**. This approach aligns perfectly with the need for an **adaptable and evolving architectural strategy** capable of responding to **changing business needs and technological landscapes**.

7.2.6: Play 6: Connect, Don't Conform

The "Connect, Don't Conform" play encourages leaders to **transcend the limitations of hierarchical roles** and the **implicit expectations of conformity**. Instead, it emphasizes the importance of **building genuine human connections** with team members. This approach involves **intentionally flattening power gradients**, valuing individuals for their **unique perspectives and contributions**, and fostering an environment of **mutual respect and trust**. Leaders are encouraged to **demonstrate vulnerability**, **show care** for their team members' thoughts, feelings, and personal goals, and **extend trust first** rather than waiting for it to be earned. The play contrasts the Industrial Age tendency to conform to hierarchical positions with the **modern need to connect** with others as individuals.

image by pakin jarerndee from istock

IT architects often find themselves in roles where they need to influence outcomes **without having direct managerial authority** over all stakeholders. In these situations, **building trust and rapport** through **genuine human connection** is far more effective than relying on **job titles** or **perceived hierarchical standing**. Additionally, **understanding**

the diverse perspectives, **motivations**, **and concerns** of various teams—such as development, operations, security, and product management—is crucial for creating **holistic**, **well-rounded**, and **widely accepted** architectural solutions. This deep understanding is cultivated through connection, not conformity.

Examples:

- **Scenario 1 (Cross-Team Collaboration)**: An architect working on a complex, enterprise-wide initiative makes a deliberate effort to have informal conversations (e.g., virtual coffee meetings or brief one-on-one chats) with key members of different engineering teams. The purpose is not solely to discuss the project agenda but to better understand their experiences: "I'd like to understand your team's perspective more deeply. What are your biggest pain points with the current platform, and what are your aspirations for how it could better support your work?" This indicates a care for people's thoughts and feelings.
- **Scenario 2 (Handling Dissent)**: In a design meeting where a particular architectural proposal is being debated, one engineer persistently voices strong criticisms. Instead of shutting down the critique or becoming defensive, the architect employing the "Connect" play might say: "I appreciate you pushing back on this and clearly sharing your concerns. It's evident you have strong reservations. Let's ensure we fully understand your viewpoint. Help me see what you're perceiving." This language values the individual and their input, fostering a more open dialogue beyond simple role-based interaction.
- **Scenario 3 (Architect Admitting Uncertainty)**: When faced with a novel and particularly challenging technical problem for which they don't have an immediate solution, an architect shares their uncertainty with the team: "Honestly, I don't have a ready-made answer for this one. This is new territory for me as well. Let's explore this challenge together. What are your initial thoughts, ideas, or even gut feelings on how we might approach this?" Admitting "I don't know" demonstrates vulnerability, builds trust, and invites collaborative problem-solving.

"Connect, Don't Conform" serves as an **overarching strategy** that enhances the effectiveness of all other leadership plays. Without **genuine**

connection, **mutual respect**, and **trust**, attempts to truly *Collaborate*, gain deep *Commitment*, or foster a safe environment to *Improve* are likely to be **less effective and more superficial**. If team members feel they are merely expected to conform to their roles or to the perceived hierarchy—acting as **cogs in a machine**—they are less likely to **openly share their best ideas**, **genuinely commit to decisions**, and will certainly not feel **safe enough to point out flaws or suggest improvements**.

Connection builds the **psychological safety and trust** that are essential for **positive team interactions**. It forms the **relational foundation** upon which effective architectural leadership is built. This play is crucial for nurturing **collaborative networks** and fostering an understanding of **human factors**, which are central pillars of the Grounded Architecture philosophy. Grounded Architecture emphasizes the importance of *Collaborative Networks* and dedicates attention to *On Human Complexity*, which includes understanding cultural differences and cognitive biases. The act of **connecting** is precisely how architects build these vital relationships.

7.3: IV. Conclusion: Integrating Leadership Principles into Your Architectural Practice

The six leadership principles outlined in David Marquet's *Leadership is Language—Control the Clock*, *Collaborate*, *Commit*, *Complete*, *Improve*, and *Connect*—provide a **valuable toolkit for IT Architects** looking to enhance their **leadership effectiveness**. By adopting these linguistic shifts, architects can transform their leadership style from a traditional, potentially less impactful approach to one that is more **empowering**, **collaborative**, and **results-oriented**. These concepts are not just abstract theories; they are **practical**, **actionable tools** that can be implemented immediately through a **conscious change in the language** we use in our daily interactions.

Architects should start by **observing their own language** and the **prevailing communication patterns** within their teams and organizations. Which principles resonate most strongly with them? Which ones address the current challenges they face? **Starting small**, perhaps by focusing on intentionally applying one or two principles, can help **build confidence** and **demonstrate tangible benefits**. **Consistent application** of these principles by architects can create a **positive ripple effect**, influencing not only their **immediate teams** but also promoting a **more constructive and innovative engineering culture** across the broader organization. As architects model this new language of leadership, others may begin to adopt similar communication patterns. This can lead to a **gradual but significant positive shift** in how technical discussions are conducted, **decisions are made**, and how **individuals and teams interact**, ultimately fostering a **more psychologically safe and innovative environment**.

Embracing these leadership principles aligns directly with the core tenets of Grounded Architecture. By mastering this new language, architects can **build stronger and more effective collaborative networks**, make **more robust and data-informed decisions** by creating the necessary space for **diverse input** and thorough analysis, enhance the **adaptability** of their architectures and practices, and deliver greater strategic value to their organizations. The journey to becoming a truly *Grounded Architect* is, in many ways, a journey of mastering the language of modern leadership. This transformation is not a one-time event but a **continuous path of improvement**, similar to mastering any complex technology or methodology. It requires ongoing practice, conscious reflection, and

a **willingness to adapt and refine** one's approach over time. This **commitment to evolving one's language** is a hallmark of an architect dedicated to not only technical excellence but also to **impactful and empowering leadership**.

7.4: Questions to Consider

- *Which of the six leadership plays (Control the Clock, Collaborate, Commit, Complete, Improve, Connect) do you feel is most lacking in your current team interactions, and what's one small linguistic change you could make this week to start addressing it?*
- *Think about a recent architectural decision or discussion. How could applying the "Control the Clock" play (e.g., calling a deliberate pause for Bluework) have potentially improved the outcome or the process?*
- *When was the last time you consciously used language to "Collaborate, Don't Coerce"? How did you ensure diverse perspectives were heard before your own (e.g., "vote first, then discuss")?*
- *Consider a current architectural standard or initiative. Are your teams truly "Committed," or are they merely "Complying"? What language could you use to shift towards genuine commitment?*
- *How can you apply the "Complete, Don't Continue" play to break down a large, ongoing architectural effort into smaller, more manageable, and celebratable milestones?*
- *Reflect on a recent project setback or architectural challenge. How could an "Improve, Don't Prove" mindset have changed the team's approach to learning from the experience?*
- *In what ways can you intentionally "Connect, Don't Conform" to build stronger relationships and flatten power gradients with stakeholders outside your direct team?*
- *How often do you find yourself or your team stuck in "Redwork" (doing) without sufficient "Bluework" (thinking/planning)? What triggers could you establish to prompt a shift?*
- *How might the "power gradient" in your organization be subtly influencing architectural discussions, and what specific phrases from Marquet's plays could help mitigate this?*
- *What is one "Industrial Age" leadership phrase you commonly use or hear that you could replace with a "New Playbook" alternative to foster better architectural outcomes?*

8: Balancing Curiosity, Doubt, Vision, and Skepticism

image by vawiley from istock

IN THIS SECTION, YOU WILL: Understand that balancing curiosity, doubt, vision, and skepticism is essential for driving sustainable innovation and change in organizations.

KEY POINTS:

- **Curiosity and wonder** spark exploration, leading to technological breakthroughs. However, without caution, it can result in premature adoption of immature solutions.
- **Doubt** forces a critical evaluation of progress, ensuring that innovative ideas are grounded in practical and validated approaches. Over-reliance on doubt can stifle risk-taking and hinder breakthrough innovation.
- **Vision and belief** provide a long-term perspective, guiding efforts toward significant, transformative goals. Yet unchecked vision can lead to confirmation bias or misdirected resources.
- **Skepticism** helps prevent overcommitment to unproven ideas, ensuring teams remain grounded. However, excessive skepticism can result in missed opportunities and discourage creative risk-taking.

- Architects must constantly reflect on how well they balance these forces. In doing so, they can ensure that their efforts are driven by a healthy combination of exploration, critical validation, strategic guidance, and cautious realism—all crucial to achieving lasting success in any organizational change.

IT architects must help organizations innovate and transform responsibly and sustainably. They must bring new ideas and support or voice concerns about others' plans and proposals here. In pursuing **sustainable innovation** and **organizational change**, understanding the underlying motivators that drive individuals and teams is essential. Drawing on my previous research, I propose the metaphor of a compass[1] to illustrate how balancing four fundamental forces—curiosity, doubt, vision, and skepticism—guides this process. Each of these motivators plays a distinct but interconnected role (Figure 1):

- **Curiosity** sparks exploration,

[1] https://researcher-practitioner.com/research-compass

- **Doubt** ensures deeper scrutiny,
- **Vision** aligns efforts with long-term goals, and
- **Skepticism** questions feasibility and assumptions.

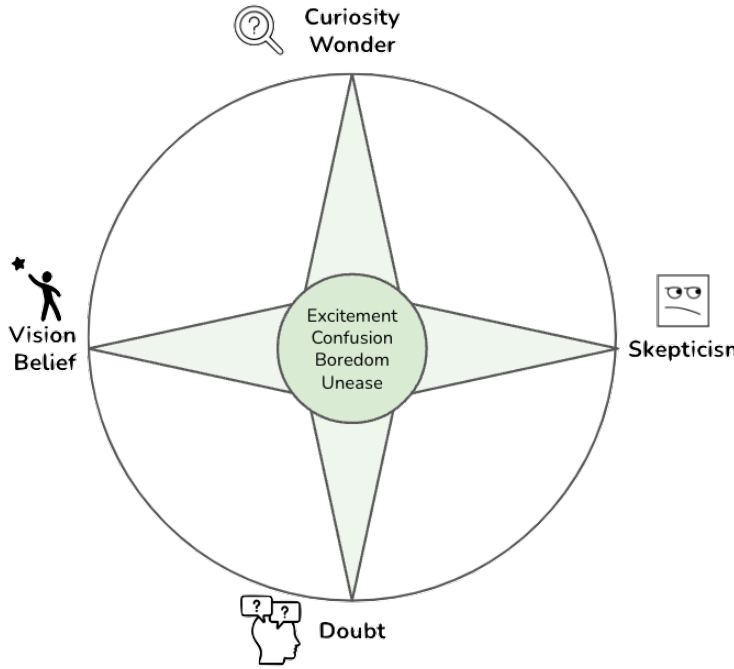

***Figure 1**: The compass for driving sustainable innovation and change in organizations.*

IT architects can be crucial in recognizing these different motivators and actively supporting contributions from all four. Their role is to ensure that:

- **Curiosity** is harnessed productively through experimentation and exploration.
- **Doubt** and **skepticism** are encouraged to critically evaluate ideas without stifling innovation.

- **Vision** aligns the work with strategic priorities, ensuring that technical innovations contribute to broader organizational goals.

The question for organizations and IT architects is not about choosing one direction (whether curiosity, vision, etc.) but about finding the delicate **balance between all four forces**. This balance drives sustainable innovation, allowing for creative growth tempered with critical thought and long-term planning.

By balancing these motivators, organizations can successfully navigate the **complexities of innovation**, ensuring that change is not only driven by bold ideas but also thoroughly validated, strategically aligned, and pragmatically grounded.

In the following sections, I describe these four forces in more detail.

8.1: Curiosity and Wonder

Curiosity and wonder are often the most intuitive and powerful motivators for change. Curiosity usually drives innovation—developers and engineers are drawn to explore new technologies, tools, or methods, and their **natural desire to learn** fuels progress. Organizations that encourage autonomy and innovation create environments where curiosity thrives.

image by mtstock studio from istock

Architects too frequently become less curious and focus on reviewing and criticizing other ideas. However, I see them as critical drivers for the **responsible exploration of new technologies**. Architects have a background knowledge that enables them to quickly understand the pros and cons of new directions. As per ThoughWorks analysis[2], there is a massive gap between the exponential growth of technology and organizations' ability to harness it, and this gap will only increase with time. Staying curious and being able to follow new technology developments is a crucial competitive advantage of any organization.

[2]https://www.thoughtworks.com/en-ec/insights/blog/seven-tenets-new-age-business-analysis

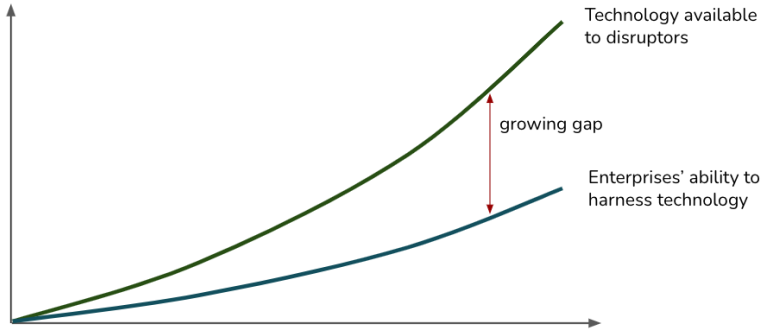

Figure 2: *Over time, technology grows at an exponential rate. But businesses aren't able to keep up fast enough. Curiosity is an important force that helps organizations adopt new technologies. Source: thoughtworks.com*

However, over-reliance on curiosity can lead to a focus on novelty rather than sustainability. Rapid, uncritical adoption of new technologies may lead to **naive solutions**, overlooking potential risks or **technical debt**. There is a fine line between innovation fueled by curiosity and a risk of unsustainable growth if curiosity is not tempered with critical thinking.

Here are a few examples where curiosity and wonder have driven innovation yet also highlight the risks of uncritical adoption of new technologies:

- **Microservices Architecture**
  - *Curiosity and innovation:* The shift from monolithic architectures to microservices was driven by curiosity and the need to solve scalability and flexibility issues. Developers were fascinated by decoupling services to allow for independent deployment and scaling, which led to significant innovations in cloud-native development, such as containerization and Kubernetes.
  - *Risk of over-reliance on novelty:* However, organizations that rushed to adopt microservices without understanding the complexity of managing distributed systems faced orchestration, monitoring, and increased operational overhead issues. In some cases, the shift introduced technical debt, making systems more complex to maintain without proper tooling.

- **Serverless Computing**
  - *Curiosity and innovation:* Serverless architectures emerged due to developers' curiosity about reducing infrastructure management. The desire to focus solely on writing business logic and letting cloud providers handle scaling and infrastructure led to major efficiency gains, especially for dynamic workloads.
  - *Risk of uncritical adoption:* On the downside, developers who rushed into serverless without considering factors like cold start delays, vendor lock-in, or limited control over execution environments ended up with architectures that were costly or hard to maintain in the long term.
- **Agile Methodology**
  - *Curiosity and innovation:* Agile methodologies like Scrum and Kanban grew from a desire to overcome the rigidity of traditional waterfall project management. The idea of iterative development cycles and continuous improvement resonated with engineers who wanted more flexibility in their work.
  - *Risk of novelty over sustainability:* Some organizations adopted Agile in name only, leading to chaotic workflows without truly understanding the principles of collaboration and continuous feedback. This has sometimes resulted in "fragile Agile" environments, where the novelty of faster iterations creates pressure without improving actual delivery outcomes.
- **Blockchain in Software Development**
  - *Curiosity and innovation:* Blockchain's promise of decentralized and immutable records intrigued many in software engineering, driving curiosity to explore use cases beyond cryptocurrency, such as secure voting systems, supply chain tracking, and data integrity.
  - *Risk of uncritical adoption:* However, in many instances, blockchain was applied to problems that did not require such a complex solution, leading to increased costs, slower performance, and added technical debt when more straightforward databases would have sufficed.

Curiosity has driven **innovation and progress** in all of these examples, but **unchecked enthusiasm** for new technologies can lead to long-term challenges if sustainability and critical evaluation are overlooked.

8.2: Doubt

When we make some visible progress, we may ask ourselves if our results are **wrong**, **coincidental**, or a result of **wishful thinking**. Such questions are the beginning of doubt, one of the most important motivators behind creating robust results. Any evaluation can be viewed as an effort to reduce doubts about our findings.

Doubt is a critical motivator that drives rigorous validation and testing of new ideas. Doubt is primarily a **positive force**. Contrary to skepticism, doubt does not question the possibility of knowing something or the validity of pursuing some direction. When we doubt some finding, we want to set it on **firmer ground** and add **more certainty**.

image by hiraman from istock

In software engineering, doubt manifests in practices like code reviews, testing frameworks, and peer reviews of architectural designs. IT architects must foster a **culture of constructive doubt**, where teams constantly question whether their solutions are robust, scalable, and fit for purpose.

However, excessive doubt can **stifle innovation**. If teams focus only on incremental improvements and avoid risk, they may miss opportunities

for breakthrough innovations. Balancing doubt and innovation is essential to creating sustainable, high-performing solutions.

Here are a few examples from IT architecture and software engineering where doubt drives rigorous validation, ensuring the robustness of systems while balancing innovation and risk:

- **Unit Testing and Test-Driven Development (TDD)**
  - *Doubt and validation:* In software engineering, doubt about whether code functions (and will keep functioning) correctly leads to the widespread use of unit testing and Test-Driven Development (TDD). Engineers write tests to ensure that each piece behaves as expected. This rigorous approach minimizes errors, reduces technical debt, and ensures robust software.
  - *Excessive doubt risk:* However, focusing too much on testing and validation can slow development. In some cases, teams may overly rely on TDD, creating rigid tests that prevent refactoring and limit innovative solutions, stifling the exploration of new design patterns.
- **Code Reviews**
  - *Doubt and validation:* Code reviews are driven by doubt that any single developer's solution is flawless. Engineers critically examine each other's code to identify flaws, improve code quality, and enforce consistency. This process helps reduce bugs, ensures compliance with design principles, and fosters knowledge sharing.
  - *Excessive doubt risk:* If the review process becomes overly critical, it can lead to delays, a decrease in team morale, and resistance to experimenting with novel approaches. An overly cautious environment may discourage developers from innovating.
- **Continuous Integration/Continuous Deployment (CI/CD) Pipelines**
  - *Doubt and validation:* CI/CD pipelines are rooted in doubt about whether the software will behave correctly in different

environments. Automated tests, builds, and deployments are triggered to ensure that changes don't break functionality, and every small change is validated before release. This approach minimizes errors and makes systems more robust.
  - *Excessive doubt risk:* An over-reliance on automation can sometimes create rigid workflows. Teams may hesitate to make significant architectural changes or take on ambitious projects, fearing that the automated checks will flag too many issues or create bottlenecks.

- **Architectural Peer Reviews**

  - *Doubt and validation:* In IT architecture, peer reviews of system designs are based on the doubt that a single architect's vision is complete. These reviews focus on scalability, security, and maintainability, ensuring that systems are designed to meet long-term goals and handle future demands.
  - *Excessive doubt risk:* Excessive critique in these reviews can lead to overly conservative decisions. Architects may avoid riskier, innovative technologies that could offer better performance or scalability out of fear that they introduce too much uncertainty or complexity.

- **Security Testing (Penetration Testing and Threat Modeling)**

  - *Doubt and validation:* Security is an area where doubt is especially critical. Practices like penetration testing and threat modeling are based on the assumption that systems are vulnerable. Engineers actively seek out system weaknesses to minimize risks and improve overall security posture.
  - *Excessive doubt risk:* If security concerns are over-prioritized, they can block new features, slow development, and create a climate where innovation is too risky. This balance is particularly tricky in industries like fintech and healthcare, where innovation and security must coexist.

In all these cases, doubt catalyzes the creation of **robust, validated solutions**. Yet, fostering a balance between doubt and innovation is essential to ensure teams don't shy away from bold new ideas.

8.3: Vision and Belief

While the term belief may have a negative connotation, it is difficult to imagine any transformation activity without some form of belief or guiding vision. Vision and belief are **long-term strategic motivators** that guide large-scale change initiatives. Successful IT architecture often begins with a bold vision—such as a belief in the transformative potential of a new technology.

image by georgepeters from istock

However, vision without critical evaluation can lead to **confirmation bias**. Architects must balance visionary thinking with the rigor of curiosity and doubt to avoid pursuing misguided or overly narrow goals.

Vision and belief are much more complex motivators than curiosity and wonder. When driven by curiosity, we follow interests and the desire to learn something new. Vision and belief, on the other hand, require a **longer-term commitment** to some idea and constant effort to focus and organize our activities. While curiosity and wonder encourage exploration, vision and belief guide and sustain efforts toward transformative outcomes.

Here are some examples where vision and belief played critical roles in driving large-scale change initiatives and transformative innovations:

- **The Internet**
  - *Vision and belief:* Early pioneers like Vannevar Bush and J.C.R. Licklider believed in a global network connecting people, information, and machines. Their vision of an "Intergalactic Computer Network"[3] became the foundation of the modern internet. Licklider's belief in the collaborative potential of interconnected computers drove funding and research, eventually leading to the ARPANET, the precursor to today's internet.
  - *Balancing with critical evaluation:* Licklider's vision was tempered by ongoing research and testing to solve problems like network protocols, security, and scalability. Engineers balanced this visionary belief with the practical work of building resilient, fault-tolerant systems.
- **Agile Development**
  - *Vision and belief:* The Agile Manifesto was born from the belief that traditional waterfall software development was too rigid and inefficient. Visionary software engineers believed in a more collaborative, iterative approach that would allow for faster delivery, continuous feedback, and greater adaptability. This belief led to the widespread adoption of Agile methodologies, revolutionizing how software is developed and deployed.
  - *Balancing with critical evaluation:* While Agile was visionary, its success depended on carefully testing its principles in real-world projects. Teams needed to critically evaluate which Agile practices worked best for them. Excessive belief in Agile without adaptation has sometimes led to failed implementations, as teams focused too much on process over outcomes.
- **DevOps Movement**
  - *Vision and belief:* The DevOps movement was driven by a belief in the transformative potential of breaking down silos between development and operations teams. Early DevOps pioneers believed collaboration, automation, and continuous

[3] https://en.wikipedia.org/wiki/Intergalactic_Computer_Network

delivery could improve software quality and speed dramatically. This vision transformed how organizations manage software lifecycles, driving innovation in automation tools like Jenkins, Docker, and Kubernetes.
  - *Balancing with critical evaluation:* While DevOps was visionary, teams had to critically evaluate its feasibility in different contexts. Not every company could seamlessly adopt a DevOps culture, and early adopters had to address the challenges of tooling, integration, and operational complexity to ensure long-term success.

- **Artificial Intelligence (AI) and Machine Learning (ML)**

  - *Vision and belief:* Visionaries like Alan Turing and, later, engineers at companies like Microsoft and Google believed in the transformative potential of AI and machine learning. Their belief in creating machines that could learn and think has led to breakthroughs like natural language processing, autonomous systems, and AI-driven decision-making. This vision continues to push the boundaries of computing and software engineering.
  - *Balancing with critical evaluation:* The AI and ML fields are filled with successes and failures. While belief in AI's potential is substantial, developing robust AI systems requires extensive testing, validation, and doubt about these technologies' reliability, ethics, and real-world application. Without a balance of critical evaluation, belief in AI's potential could lead to inflated expectations and misuse.

In all these examples, vision and belief served as powerful, **long-term forces** that propelled transformative technological changes. However, these bold ideas were successful because they were accompanied by curiosity, rigorous doubt, and constant critical evaluation to ensure their feasibility and sustainability in the real world. These revolutionary ideas might have failed to materialize without balancing belief with practical testing and adaptation.

8.4: Skepticism

Skepticism is a loaded term with several definitions. Closest to the meaning I use here is the Wikipedia definition of skepticism as *"doubt regarding claims that are taken for granted elsewhere"*. Similarly, I view skepticism as a **reality checker** that questions the fundamental premises we usually take as a given. As such, skepticism can call attention to the **viability**, **feasibility**, **or practicality** of a direction or approach. Contrary to doubt, which can motivate us to investigate some topic further to obtain more certainty, skepticism may call us to abandon some lines of inquiry and consider alternatives.

image by izusek from istock

Fred Brooks's paper **"No Silver Bullet—Essence and Accidents of Software Engineering"**[4] is probably one of the best examples of helpful skeptical thought in software engineering. Brooks expressed his skepticism toward approaches to software engineering research that aim to discover a single solution that can improve software productivity by an order of magnitude. Brooks seriously questioned the possibility of ever finding such

[4] https://en.wikipedia.org/wiki/No_Silver_Bullet

"startling breakthroughs," arguing that such solutions may be inconsistent with the nature of software. Brooks also made clear that his **skepticism is not pessimism**. While Brooks questioned the possibility of finding a single startling breakthrough that would improve software productivity by an order of magnitude, he believed that such improvement can be achieved through disciplined, consistent effort to develop, propagate, and exploit a number of smaller, more modest innovations.

Skepticism plays a crucial role in keeping organizations grounded. In IT architecture, skepticism questions whether specific approaches or technologies are viable. Skeptical thinking can help **prevent costly mistakes** by challenging assumptions and forcing teams to consider alternative approaches. This type of critical thinking is essential for ensuring that decisions are sound and that innovation is pursued responsibly.

Yet too much skepticism can lead to **missed opportunities**. Overly critical attitudes can reject promising innovations before they've had a chance to prove themselves. Furthermore, if skepticism is not a well-argued result of long experience, it may trigger an emotional debate without contributing much to it. Similar to vision, practical skepticism requires deep knowledge and a fundamental understanding of the field.

Here are a few examples where skepticism has played a crucial role in questioning underlying assumptions, helping to prevent misguided decisions while still allowing space for innovation:

- **Monolithic vs. Microservices Architecture**
  - *Skepticism about microservices:* While many organizations have embraced microservices as the modern solution for building scalable and flexible systems, some IT architects have expressed skepticism about whether microservices are the right solution for every use case. This skepticism arises from concerns over complexity, the operational overhead of managing numerous services, and potential performance bottlenecks.
  - *Impact:* Skeptical architects have questioned the assumption that breaking down every application into microservices is inherently better, advocating for alternatives such as modular monolithic architectures. This skepticism has led some teams to avoid unnecessary complexity and better tailor their architectural choices to the project's needs.

- **Blockchain for Everything**
  - *Skepticism about blockchain use cases:* Blockchain technology has been widely hyped as a transformative innovation for everything from finance to supply chains to healthcare. However, skeptics have questioned whether blockchain is necessary or even practical for many of these use cases, citing concerns about performance, scalability, and the complexity of decentralized systems.
  - *Impact:* This skepticism has led to a more nuanced and careful consideration of where blockchain is a good fit versus where traditional databases or systems are more effective. As a result, many organizations have avoided costly mistakes by not adopting blockchain for applications where it offers little to no advantage.
- **Artificial Intelligence (AI) Hype**
  - *Skepticism about AI's current capabilities:* While AI is often viewed as the future of many industries, there is considerable skepticism about its real-world capabilities, particularly in areas like fully autonomous driving, generalized AI, and complex decision-making. Skeptics question whether AI technology, as it currently stands, can live up to the hype without significant advancements in data, algorithms, and ethics.
  - *Impact:* This skepticism has led to a more cautious and realistic approach to AI adoption. Organizations are more likely to adopt AI that can offer tangible, practical benefits (e.g., automation of routine tasks, pattern recognition) rather than diving headfirst into ambitious, underdeveloped AI projects.
- **Agile Methodology Skepticism**
  - *Skepticism about Agile's universality:* While Agile methodologies have been highly successful for many teams, some engineers and IT architects remain skeptical about its applicability in all contexts. Critics argue that Agile when applied dogmatically, can lead to chaotic environments where long-term planning and architectural considerations are neglected.

  - *Impact:* This skepticism has encouraged organizations to adopt Agile principles to fit their needs rather than blindly adopting the methodology. Some teams blend Agile with other structured approaches to balance flexibility and long-term planning, leading to more sustainable development practices.

- **Skepticism about "Serverless" Computing**

  - *Skepticism about the limits of serverless:* Serverless computing promises to abstract away the complexities of managing infrastructure, allowing developers to focus on writing code. However, skeptics point out limitations such as cold starts, vendor lock-in, and the complexity of debugging in serverless environments.
  - *Impact:* This skepticism has led some organizations to avoid over-committing to serverless architectures for mission-critical applications. Instead, they use serverless selectively, adopting hybrid approaches that balance the benefits of serverless with the control offered by traditional infrastructure.

In these examples, skepticism has served as a valuable "**reality check**," challenging assumptions that may otherwise be taken for granted. While skepticism can prevent teams from rushing into unproven technologies, it's also essential to strike a balance—excessive skepticism can stifle innovation and prevent organizations from exploring potentially transformative ideas. Skeptics in IT architecture and software engineering remind us to be thoughtful, deliberate, and pragmatic in approaching change.

Skepticism is probably one of the most complex forces. Contrary to doubt, which can rely on several tools and methods (e.g., experiments, ethnographical methods), there are no simple and structured tools for skepticism. Practical skepticism requires careful thought, experience, and an excellent overview of the field.

8.5: Balancing Motivators in IT Architecture

The four points of the compass—curiosity, doubt, vision, and skepticism—are not mutually exclusive. Effective IT architecture teams **blend these motivations** into their work. For example, curiosity may drive the exploration of new technology, while doubt leads to rigorous testing, and vision provides the strategic framework for long-term development.

image by happyphoton from istock

Balancing these motivators at an organizational level is essential for fostering a culture of innovation, resilience, and continuous improvement. Without vision and belief, there would be no forward momentum; without skepticism and doubt, there would be no accountability or critical evaluation.

To guide teams, IT architects should ask the following questions:

- Are we exploring new technologies and approaches with enough curiosity or only looking at things we already know?
- Do we have a clear long-term vision guiding our architectural decisions? Or are we blindly following industry trends, jumping

from one hype to another? A well-defined vision not only steers our actions but also fuels our determination to achieve our long-term goals.

- Are we using appropriate methods to validate our solutions and address doubts?
- Are we skeptical enough of our own work, or are we being overly critical of innovative ideas?

By answering these questions, IT architects can help teams ensure they make **informed**, **balanced decisions** that sustainably drive organizational development. The goal is to create a harmonious environment where change is driven by curiosity, vision, doubt, and skepticism—leading to more sustainable and impactful outcomes.

8.6: To Probe Further

- The Four Points of the Research Compass[5], by Željko Obrenović, 2013

[5] https://researcher-practitioner.com/research-compass

8.7: Questions to Consider

- *How does curiosity drive innovation in your organization, and how do you prevent it from leading to unsustainable or risky decisions?*
- *In what ways can doubt contribute to the quality and robustness of your work, and how do you ensure that it doesn't stifle innovation?*
- *How do you balance long-term vision and belief with the need for critical evaluation and validation in your decision-making?*
- *How do you cultivate constructive skepticism without allowing it to turn into cynicism that hampers creativity?*
- *When exploring new technologies, how do you ensure you follow industry trends and align them with a clear strategic vision?*
- *How can fostering a culture of curiosity, doubt, vision, and skepticism lead to more resilient and impactful solutions in your organization?*
- *What strategies can you use to balance risk-taking with rigorous validation in the innovation process?*
- *How does your organization support contributions driven by the four motivators: curiosity, doubt, vision, and skepticism?*
- *In what ways can skepticism be used as a tool for improvement without dismissing promising new ideas too early?*
- *How do you foster environments where curiosity and exploration are encouraged but tempered with thoughtful, critical evaluation?*