Enterprise Architecture Function

Ingo Arnold

Enterprise Architecture Function

A Pattern Language for Planning, Design and Execution

Springer

Ingo Arnold
Riehen, Switzerland

ISBN 978-3-030-84588-9 ISBN 978-3-030-84589-6 (eBook)
https://doi.org/10.1007/978-3-030-84589-6

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

For my family, my loved ones and for my wonderful sons Jonas and Luis who followed their father's path into computer science.

Preface

Architecture Has Become All-Pervasive in the Twenty-First Century

The topics of *architecture* in general and *enterprise architecture* in particular have become ubiquitous in the twenty-first century. The networked universal computing machine (*the world computer*) has directly or indirectly permeated virtually every aspect of our lives, transforming *the way we cooperate, partner, create, and run businesses and plan, implement, and provide goods and services. What were originally distinguished as* business *and* technology *are merging and constituting a common* business model *backbone that is neither distinguishable in form nor in essence.* At the same time, we find that dealing with, if not mastering, complexity has become the central challenge for modern enterprises. Successfully planning, designing, and operating business models today is increasingly synonymous with establishing and evolving their digital means and building materials. In summary, business ecosystems have gained significantly in potential through digital means on the one hand but have also become inherently more complex on the other—both in their breadth (*many touchpoints*) and in their depth (*complex touchpoints*). Architecture has become ever-present because of its enormous relevance for the enterprise-wide planning, construction, and operation of digital material constituting digitalized services and means.[1] Job titles on business cards include *enterprise*

[1] In terms of digital means, the industry delineates two terms that sound very similar but have important differences in their conception: *digitalization* and *digitization*. Gartner, for example, defines digitalization in its IT Glossary as follows: "Digitalization is the use of digital technologies to change a business model and provide new revenue and value-producing opportunities; it is the process of moving to a digital business." Similarly, definitions of digitization have been proposed, with Gartner defining the term in their IT Glossary as follows: "Digitization is the process of changing from analogue to digital form, also known as digital enablement. Said another way, digitization takes an analogue process and changes it to a digital form without any different-in-kind changes to the process itself" (Gartner Glossary 2020).

architecture, *architecture management*, *business*, and *domain architecture*, or *enterprise solution architecture*. Architects participate in steering board meetings, contribute to strategic decision-making, create roadmaps, identify consolidation opportunities, or identify systematic shortcomings from a perspective that goes beyond individual services. Although the term *enterprise architecture* is so commonly used, a closer look reveals that architects, strategists, portfolio managers, project managers, and other stakeholders lack a common understanding of its conception.

At the Same Time Architecture Is Interpreted in Lots of Different Ways

For some, *enterprise architecture* is an architecture super-discipline that encompasses all other disciplines, like domain, solution, security, or data architecture; for others, it is a framework or tool; for still others, it is part of the corporate police department trying to make the good work of others hurt; for many, *enterprise architecture* may be a binder of drawings geared to help a manager convince others that he is in charge and in control and that his yard is in beautiful order. In its practical use, the term *enterprise architecture* covers quite a wide field—that is, it is not uniformly defined or understood.

My Own Architecture Journey

I have spent my entire professional life in the architecture arena. While practically designing and delivering concrete solutions caught my initial attention, over time I broadened my perspective of architecture as a strategic planning and governance discipline. Ten years ago, I wrote a book on *solution architecture* with three friends (Vogel et al. 2011). The goal of the book was to strike a good balance between the aspects of holistic orientation, theoretical substance, and practical guidance—a consolidation of our architecture experiences at that time. When we thought about writing the book, the topic of solution architecture lacked a holistic framework that established a common, unified terminology and described architecture in a differentiated way. We had all been intuitively searching for a framework that covered the essential solution architecture dimensions for a long time. Throughout our professional lives and educational journeys, each of us had developed our architecture understanding from individual insights. With the book we wrote, we finally reached

the point where we reconciled our individual insights and finally merged them into a common architecture orientation framework that became the core of our book.

Ten years have passed since then, and I decided it was the right time to summarize my experience in the field of *enterprise architecture* and write the book you are holding in your hands. However, there is a difference between the situation of solution architecture in 2011 and the topic of enterprise architecture in 2021. In the area of solution architecture, a framework gap existed, and we suffered from lacking an orientation model that was timeless, agnostic to technology, and that provided a sustainable reference beyond temporary architecture trends. In the area of enterprise architecture, the situation today is almost exactly the other way around. Numerous enterprise architecture frameworks claim to cover the topic in whole or in part, and an enormous amount of effort has been and is being expended to not only create the corresponding paperwork but to build an entire industry on it.

Initially I Was Enthusiastic About Enterprise Architecture Frameworks. . .

When I started my journey into enterprise architecture, I found that there were already several frameworks. They included definitions of what architecture means, suggested view models, development processes, roles and responsibilities, artifacts, and nomenclatures for modeling and more. I considered myself fortunate to be able to build on proven practices, most of which were offered in a comprehensive, unique package. These frameworks impressed me a lot—not only but also because of their sheer size, emphasis on inherent structures, and the list of prominent companies that have contributed to them and claim to use them themselves. And I was not alone. Academic and non-academic publications, enterprise architecture experts and consultants, architecture tool and platform vendors, and practitioners in the field—basically everyone I met or everything I read (a few exceptions included)—sang one song in unison: "Enterprise architecture frameworks have everything you need—just do it."

Since an *enterprise architecture function*[2] has full responsibility for the governance and execution of the architecture within an enterprise, corresponding

[2] Enterprise architecture functions establish architecture capabilities organizationally—they are socio-technical systems that realize architecture disciplines by integrating their full breadth and depth into an enterprise-wide operating model.

frameworks are basically blueprints that identify all aspects that a well-designed architecture function generally needs to consider.

When I speak of *enterprise architecture frameworks* here, I am not thinking of Zachman,[3] *The Open Group Architecture Framework*[4] (TOGAF), or ArchiMate[5] in the narrowest possible sense—i.e., not in the sense that their cores are released by respective consortia and then eke out a quiet existence on bookshelves or on their official websites. A complementary (yet essential) addition to *naked* TOGAF or ArchiMate is a vast commercial ecosystem, an ecosystem that includes consulting services; tons of publications, books, tools, and platforms that feed respective tool vendors; and an architecture certification industry. In addition, there are countless architecture practitioners who have gone to the trouble of acquiring certifications—and who want something in return for the investment they have made. Finally, those practitioners whose frustration with using such frameworks increased over time also remain stabilizing factors. While frustrated practitioners, on the one hand, have painfully learned that an architecture framework is not "all you need" and that it is not enough to "just do it," these practitioners, on the other hand, have also learned how useful the authoritative power of dropping framework names can be in the pursuit of their own personal agendas.

. . .Until I Realized a Widely Spread Misconception

When I first started my enterprise architecture journey, I quickly climbed the learning curve thanks to these frameworks, delighted in what my colleagues had to say, and happily joined in the enterprise architecture song myself. I trusted in the wisdom of crowds. However, despite my initial enthusiasm, I painfully realized over the years that the claim of frameworks to be all you need is false. There were simply too many disappointments, frustrations, and merely declared successes (which were in fact failures) to continue to ignore the evidence of a significant gap between the

[3] The Zachman Framework proposes a basic enterprise architecture schema that provides a structured approach to holistically viewing and defining an enterprise's architecture. The basic scheme distinguishes two dimensions that introduce the intersection of two classifications. While the first classification distinguishes the primitive interrogatives what, how, when, who, where, and why, the second classification is derived from the philosophical concept of reification (i.e., from the process of transforming an abstract idea into its instantiation and vice versa).

[4] The Open Group Architecture Framework (TOGAF). TOGAF proposes an approach to design, plan, implement, and govern enterprise architecture. It describes a generic method for developing architectures. TOGAF suggests a common vocabulary, a generic information model, an adaptable role model, general architecture artifacts, and tooling.

[5] ArchiMate is an open and independent modeling standard for enterprise architectures that supports the description, analysis, and presentation of architecture within and between architectures. ArchiMate is an open group standard and is based on fundamental architectural concepts defined by the IEEE (i.e., IEEE 1471).

frameworks themselves and their successful operationalization in a specific business context.

Do not get me wrong. Enterprise architecture frameworks are extremely useful in many ways. They do a good job of aggregating architecture best practices, generic knowledge, and reference material and making them compactly available to the architecture community. The fact that frameworks exist and are widely accepted sends a silent, albeit clear, message to all: do not reinvent the wheel where proven practices exist. While it is naïve to recklessly ignore best practices, believing that an architecture practice or framework is in itself a *solution* that is fit for purpose in a concrete environment without adaptation is similarly naïve.

When I reached the peak of my frustration curve, I took stock of *what* I saw going wrong frequently when using architecture frameworks. Next, I analyzed *why* this was happening. At a very high level, the *what* can be traced to one, surprisingly common, misconception: the extremely unfounded expectation that a generic solution will completely satisfy the requirements of a specific problem. Generic solutions such as architecture frameworks propose generic best practices, techniques, and other responses to appropriately generic problems. However, concrete problems must be addressed by concrete solutions. Where concrete problems differ, concrete solutions differ as well.

When I wanted to understand *why* this is such a widespread misconception, I found it due to a paradox: The impressive amount of material that frameworks contain, their apparent completeness, supposed comprehensiveness, as well as their structuredness, loudly confirmed by a huge commercial ecosystem, give the strong impression that this is *all you ever need* and that you *do not need to do or adapt anything else*. I have seen architects with years of experience naively (i.e., 1:1) adopt generic framework practices for their concrete solutions. The real paradox is that the richer the framework, the worse it is applied. It is the heightened version of the well-known "a fool with a tool is still a fool" aphorism: "a fool with a tool is an armed fool"—armed in the sense that the fool has successfully immunized himself against the insight of being a fool in the first place.

Problems in Operationalizing Enterprise Architecture Frameworks

Obviously, the naive use of generic reference models almost inevitably leads to suboptimal concrete solutions. To better understand this phenomenon, I have held generic frameworks against concrete architecture function designs over the past 10 years and analyzed frequently observed, or experienced, shortcomings or even complete failures. Here, it did not matter whether it was an initial function design or an evolution to improve its fitness. Based on the analysis, I narrowed down those aspects that disproportionately determine success versus failure. It is imperative that

these be designed correctly and therefore deserve our undivided attention—so let me give you specifics on some of the misconceptions.

For example, when adopting frameworks, architecture functions do not consider the complete service lifecycle as equally relevant. Enterprise architecture frameworks tend to overemphasize planning aspects (*plan*) while at the same time underemphasizing their contribution to continuous service improvement (*build* and *run*). Another faulty notion to which architecture functions succumb under overwhelming framework impression is to focus too much on the internal makeup of a function (*white-box perspective*) without considering the expectations of their actual stakeholders (*black-box perspective*). Frameworks also largely conceal their concrete establishment in the form of an organizational capability. As a result, they lack references to *evolutionary fitness* as an important design maxim for architecture functions. This in turn leads to regular organization restarts, which undermines trust and acceptance. To give another example, frameworks seem taxonomically complete at first glance. Accordingly, architecture functions often expend little or no effort in creating a company-specific, stringent, and unambiguous conceptual as well as terminological foundation. This inevitably creates a relevant gap in understanding since the central conceptions of each business are simultaneously subsets and supersets of a respective framework glossary. Architecture functions pay the price for this bitterly and yet at the same time almost unnoticeably: in the form of inefficient, ineffective, and mostly never-ending debates, which in turn lead to suboptimal decisions and their protracted consequences. Furthermore, the frameworks often lack references to differentiations that are important in practice. For example, many frameworks do not give any indication of the *architecture levels*[6] to be differentiated in a company or do not adequately separate the *perform mandates* of an architecture organization from its *governance*[7] *mandates*. This, in turn, leads to imprecise organizational boundaries. As a final example of the inadequate adoption of frameworks by architecture functions, consider the inconsistent and disconnected definition of practices, like *architecture methods*, *view models*, *patterns*, *principles*, and *roadmaps*. These are often implemented incompletely, vaguely, and thus inconsistently and are isolated from each other, which leaves their potentials insufficiently exploited.

Aim of This Book

The blurred line between critical and semi-critical success factors in frameworks showed that we do not need larger or more frameworks but that the central gap appears where the concrete design of enterprise architecture functions is concerned. For this reason, I have placed the architecture function at the center of consideration.

[6] See the *Architecture Level* pattern in this book's pattern catalog (Chap. 6).

[7] See the *Architecture Governance* pattern in this book's pattern catalog.

In other words, I make it a first-class citizen and view the topics of architecture and frameworks through the ocular of their organizational incarnation.

Specifically, my book proposes how to design an architecture function so that it ultimately meets the expectations of the business for effectively developing, delivering, as well as operating digitalized services. My book will further outline what roles, services and processes, elaboration and reference methods, disciplines, or artifacts an architecture function is responsible for. It will make tangible to you the overall value proposition of an architecture function, as well as introduce an enterprise operating model into which architecture integrates as an organizational capability. In doing so, I do not reduce the term *architecture function* to the realm of IT but remain largely agnostic in the objective as well as normative portions of my book—agnostic regarding the nature of the contributions (e.g., business versus IT), the process models and attitudes employed (e.g., agile, waterfall, DevOps), the size of the enterprise, and other determining factors.

When an enterprise architecture framework is a blueprint for the subject of enterprise architecture, then my book is a blueprint for an architecture function enabling a digitalized business in the twenty-first century.

In any book, beyond *what* is to be described, the way in which (*how*) it presents its content and offers guidance and orientation accordingly is significant. For my book, the challenge was, on the one hand, to capture the full breadth and depth of the topic of architecture function design and, on the other hand, not to reduce the multifaceted nature of the theme to an oversimplified formula or a monodimensional set of function building blocks. In addition to typical framework content, establishing organizational capabilities is not only about the methods, roles, activities, artifacts, and responsibilities of an architecture discipline but also about its capacity, funding, mandate, engagement model, or adequate organizational embedding in a surrounding operating model, to name just a few examples. Because my book, unlike frameworks, claims to literally generate an architecture function design tailored to its context, I decided to use a pattern language regarding the *how* of my book. I deliberately chose a pattern language and patterns because they have a high degree of familiarity among architects, scale well, but also have the flexibility to accommodate design-determining factors of very different kinds in a single design proposal.

I encourage all readers to follow the navigational structures of my book closely. These structures allow you to fully grasp an architecture function in its holism and plasticity and thus to plan, design, and ultimately operate it accordingly. You will further round out and condense your understanding of what is presented in a pattern by following its references to associated patterns. In this way, you iteratively increase your understanding of architecture functions in general and develop a fully customized design for your own architecture function in particular.

Overall, this book is an expression of my desire for a work that meaningfully structures the design of an enterprise architecture function while providing hands-on guidance for practitioners. In particular, the book is independent of any particular enterprise architecture framework, mindset, tool, or platform—thus timeless in that regard. It belongs to that group of foundational works that provide you with a stable

and future-proof reference that transcends current or future trends in enterprise architecture schools. Writing this book demanded an intensive and in-depth examination of the subject of architecture beyond the usually isolated considerations of individual aspects. During the time I planned, designed, and wrote this book, I learned a great deal and steadily broadened my own perspective and experience. On the one hand, I drew on my own experience; on the other hand, I discussed with many enterprise architects and with colleagues in my network who held and still hold senior architecture positions in a variety of multinational companies and across many industries. I also had challenging conversations with students and people freshly entering the field of architecture. All these exchanges helped me to look at the subject from many different, new, and fresh angles. As a result of these fruitful debates, I gained valuable knowledge and a deeper understanding regarding the design and operation of an enterprise architecture function.

What you hold in your hands is my approach of organizing and explaining architecture from the perspective of an architecture function, putting it on a solid conceptual foundation and merging it into a pattern language. I sincerely hope that this book will help you build and evolve your own specific architecture function, customized to the needs of your business. You are most welcome to share your experiences, successes, as well as failures, or just questions, with me. Please let me know where my book has been of great help to you. But please also be sure to let me know where you have unanswered design questions even after reading the book. I am extremely curious to hear from you about how you fared on your own personal architecture function journey.

Riehen, Switzerland Ingo Arnold

References

Vogel, Oliver, Ingo Arnold, Arif Chugtai, Timo Kehrer, *Software Architecture: A Comprehensive Framework and Guide for Practitioners*, Springer-Verlag, Berlin, 2011

Gartner, *Gartner Glossary*, https://www.gartner.com/en/information-technology/glossary, 2020

Contents

About the Author

Ingo Arnold shaped the *enterprise architecture function* of a global Fortune 50s in the pharmaceutical industry and held a variety of architecture management positions over the course of 25 years. In addition to his roles in industry, Ingo is an assistant professor at universities in Switzerland and formerly in Germany, sharing his insights and experiences with several generations of computer science students. As the author of books on solution architecture, Ingo covers virtually the entire architecture spectrum of modern enterprises. Finally, Ingo is a well-known speaker at conferences, where he gives talks to international audiences on topics such as architecture management, governance, enterprise, solution, and security architecture.

List of Abbreviations

AADL	Architecture Analysis and Design Language
ABB	Architecture Building Block
ACID	Atomicity, Concurrency, Isolation, Durability
ADL	Architecture Description Language
ATAM	Architecture Trade-off Analysis Method
ATM	Automatic Teller Machine
BLOB	Binary Large Object
BPMN	Business Process Model and Notation
CAPEX	Capital Expenditure
CMM	Capability Maturity Model
CMS/CMDB	Configuration Management System or Database
COBIT	Control Objectives for Information and Related Technologies
CoP	Community of Practice
CSI	Continual Service Improvement
DoD	Definition of Done
DSL	Domain-Specific Language
EAM	Enterprise Architecture Management
ERP	Enterprise Resource Planning
GIOP	General Inter-ORB Protocol
I18N	Internationalization
IAM	Identity and Access Management
IEEE	Institute of Electrical and Electronics Engineers
IoC	Inversion of Control
IoT	Internet of Things
IT4IT	IT for IT
ITIL	Information Technology Infrastructure Library
KPI	Key Performance Indicator
MDA	Model-Driven Architecture
MIB	Management Information Base
ML	Machine Learning
NFR	Non-functional Requirement

OODBMS	Object-Oriented Database Management Systems
Open CA	Open Certified Architect
OPEX	Operational Expenditure
ORM	Object-Relational Mapping
PaaS	Platform as a Service
QAS	Quality Attribute Scenarios
RACI	Responsible, Accountable, Consulted, Informed
RDBMS	Relational Database Management System
RMI	Remote Method Invocation
ROI	Return on Invest
RPC	Remote Procedure Call
SaaS	Software as a Service
SEI	Software Engineering Institute
SLA	Service-Level Agreement
SPOC	Single Point of Contact
SWOT	Strengths, Weaknesses, Opportunities, Threats
TCO	Total Cost of Ownership
TOGAF	The Open Group Architecture Framework
UML	Unified Modeling Language