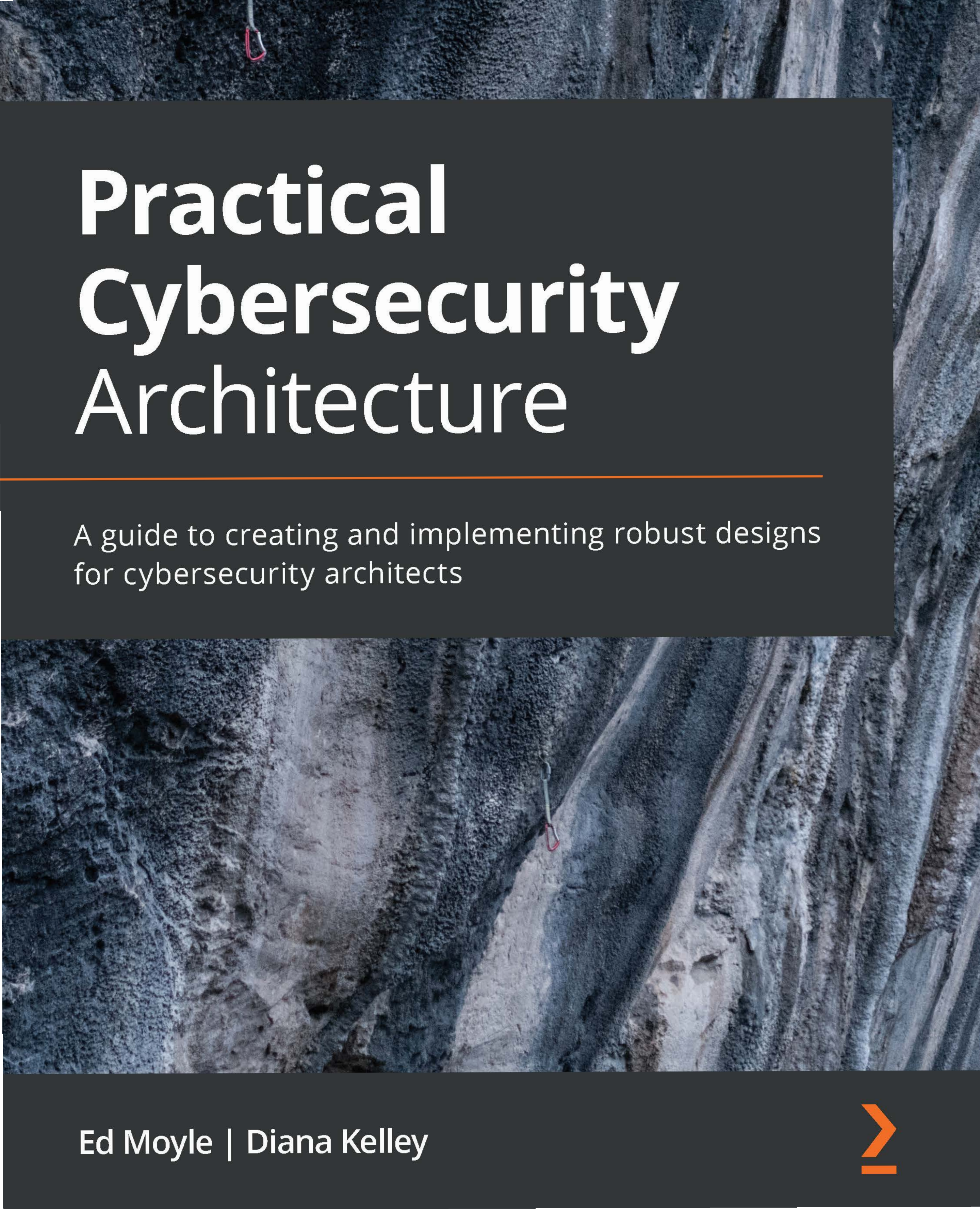

Practical Cybersecurity Architecture

A guide to creating and implementing robust designs for cybersecurity architects

Ed Moyle | Diana Kelley

Practical Cybersecurity Architecture

A guide to creating and implementing robust designs for cybersecurity architects

Ed Moyle

Diana Kelley

Packt

BIRMINGHAM—MUMBAI

Practical Cybersecurity Architecture

Commissioning Editor: Vijin Boricha
Acquisition Editor: Meeta Rajani
Senior Editor: Rahul Dsouza
Content Development Editor: Alokita Amanna
Technical Editor: Soham Amburle
Copy Editor: Safis Editing
Project Coordinator: Neil Dmello
Proofreader: Safis Editing
Indexer: Priyanka Dhadke
Production Designer: Shankar Kalbhor

First published: October 2020

Production reference: 1221020

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-83898-992-7

`www.packt.com`

Contributors

About the authors

Ed Moyle is currently a partner with SecurityCurve. In his 20 years in information security, Ed has held numerous positions, including Director of Thought Leadership and Research for ISACA, Senior Security Strategist with Savvis, Senior Manager with CTG, and Vice President and Information Security Officer for Merrill Lynch Investment Managers. Ed is a co-author of *Cryptographic Libraries for Developers* and a frequent contributor to the information security industry as an author, public speaker, and analyst.

Diana Kelley's security career spans over 30 years. She is co-founder and CTO of SecurityCurve and donates much of her time to volunteer work in the cybersecurity community, including serving on the ACM Ethics and Plagiarism Committee, as CTO and board member at Sightline Security, board member and Inclusion Working Group champion at WiCyS, and RSAC US Program Committee. She was the Cybersecurity Field CTO for Microsoft, Global Executive Security Advisor at IBM Security, GM at Symantec, VP at Burton Group (now Gartner), and a manager at KPMG. She is a sought-after keynote speaker, a co-author of the book *Cryptographic Libraries for Developers*, and one of Cybersecurity Ventures 100 Fascinating Females Fighting Cybercrime.

About the reviewer

G. Colin Campbell has been working in the information technology and cybersecurity fields for 25 years, focused in particular on building and testing secure corporate infrastructure. He has worked as a consultant with clients at all levels, but has focused on financial services companies in recent years. Within companies, Colin has managed business continuity, penetration testing, log management, and other associated disciplines. He is passionate about security inside and outside of corporate organizations and has led basic computer security seminars for users at all levels.

I'd like to thank the Packt Publishing team for the opportunity to review and contribute to this book. I'd also like to thank my wife and son for allowing me to sacrifice some time outside of work hours to let me help with this project.

Packt is searching for authors like you

2
The Core of Solution Building

In the first chapter of this book, we went through what a cybersecurity architect is, what they do (that is, the functions they perform), and the value they – and the architecture process itself – can provide to an organization. In this chapter, we will build on that to begin the process of describing how to develop an architecture.

Over the course of this chapter, we will explore the structures that organizations set up to ensure key outcomes are achieved. It is important for the architect to understand these elements because the whole point of architecture is to support the organizational mission – to do that, we have to understand what that mission is and the strategies that the organization has put in place to achieve it. This will be the backdrop to all the work we do.

We'll walk through how organizations derive their goals (including security goals), and strategies they will use to ensure those goals are being met and that resources are used optimally in support of them. We'll go through the *structures* (that is, the processes, documentation, and technology systems) that organizations use to ensure that these things are happening and that strategies to realize goals are performing as expected. Lastly, we'll walk through a process that you can use to tailor such an organization-specific "backdrop," custom to your organization, which you will use as context and a guide for your architecture efforts.

We'll achieve this by working through the following topics:

- Terminology
- Understanding solution building
- Establishing the context for designs
- Understanding goals
- Structures and documents
- Risk management and compliance
- Establishing a guiding process

Terminology

Before we dive into the meat of this chapter, the first thing we should note is that we have made the conscious decision to use terminology that will be most immediately transparent and accessible to all readers. This, however, will not always be the exact *same* terminology used by many of the security architecture frameworks, standards, and guidance that you may encounter.

For example, one of the most important concepts in this section is understanding the goals of the organization and mapping those goals to security outcomes and principles, as we stated previously. It is quite literally the case that any security effort, whether architecture, operations, incident response, or any other discipline within security, exists solely to help enable the organization to fulfill its mission. In the case of a for-profit entity, such as a company, this might mean being profitable or providing value to shareholders. In the case of a philanthropic organization, it might mean providing value to the communities it serves. In government, this might mean providing public services to citizens and ensuring transparency. In short, it's whatever is important to the organization to be successful at whatever its mission is.

Formally, this process of ensuring that resources are used in support of the organization's goals and mission is what many frameworks and standards mean when they use the word "governance." In fact, many frameworks and standardized approaches (both for technical architecture and otherwise) start from the point of view of "governance" used with this meaning.

There are a few problems with this, however:

- The first problem is that not all of the frameworks use this same language – or use it in the same way. For example, the concept of understanding the organization's goals is important in all of them (they have to be for the reasons that we outlined), but they don't all use consistent language to refer to it. For example, O-ESA uses the formal sense of the word "governance" (*The Open Group*, *Open Enterprise Security Architecture (O-ESA): A framework and template for policy-driven security*, *Van Haren. Zaltbommel*). OSA uses the word *governance* too, but instead uses it as a specific component of the overall "security architecture landscape" (although note that the source material implies a slightly broader definition of the word than that outlined by O-ESA) (`http://www.opensecurityarchitecture.org/cms/foundations/osa-landscape`). SABSA, by contrast, prefers the language of "business enablement"(*John Sherwood et al*, *Enterprise Security Architecture: A business-driven approach*, *CRC Press*).
- The second problem is that the word "governance" itself is one that can be used differently by different people. Sometimes, though not often, it is used in the formal way we described previously. More commonly, it is used in other ways. For example, one person might use the term to refer to the organizational model of an organization (for example, *the board of directors provides corporate governance*), another in reference to a political body or national government (*the governance of a nation*), yet a third in the formal sense that O-ESA and TOGAF mean it (*technology governance*), and someone else to refer to influence or control (*the king's governance over their realm*).

These factors mean that reasonable people talking to each other can easily speak at cross purposes about this topic in a way that leads to misunderstanding. It is critical to the architecture process that we minimize the likelihood of being misunderstood. In fact, as we go through this book, you will see that communication is probably among the most important (if not *the* most important) skills that an architect can have. We will want to be well understood by others we work with, and we want to make sure that we are well understood by you. It is frankly a hard-enough exercise to get right without using language that engenders or lends itself to miscommunication.

Therefore, we've chosen language that we hope is unambiguous throughout. If we mean *enabling the mission of the organization*, we will say that. We will attempt to avoid terms (such as "governance") that might be potentially ambiguous, even when that same terminology is used differently elsewhere outside of this book.

While this is advantageous to us in conveying the concepts clearly to you here and now, we wanted to be clear from the get-go that we're doing this. Why? Because it can get you in trouble if you look in other sources and see the same concepts being referred to in other ways. Should you decide to look into other architectural materials and methods, keep in mind that the same concepts can be conveyed in different ways and that, depending on which material you are leveraging, some of them might use different terms than those we are employing here.

Understanding solution building

In order to get started developing a cybersecurity architecture, we need to gather a few *raw materials* first. These are the items that will set the context for all the design work that we will undertake in subsequent chapters. Specifically, we need to first obtain a baseline understanding of the organization itself; this helps ensure that the measures we will later incorporate into designs are appropriate, practicable, and in line with the context. This is, in turn, because the nuances and specifics of the organization – everything from its goals, to its culture, to its "mission" and unique needs – will ultimately drive the design. Everything about the design – the scope, security measures, implementation, operational constraints, and functional requirements – must account for the context in which it will operate. That context is an extension of the organization itself.

As an example of what we mean here, consider a situation where you decide to plan a fishing trip. There are a lot of tasks you'd need to complete to do so, and several important decisions that you'd need to make along the way. Specifically, you'd need to do the following:

- Acquire the appropriate equipment (tackle, bait, fishing rods, and so on).
- Acquire any provisions that you'd need (food, beverages, hot or cold weather clothing, and so on).
- Organize secure lodging and travel arrangements.
- Depending on the type of fishing you want to do, you may need to secure a boat, and specialized equipment such as drills (ice fishing), depth finders or radar (deep water fishing), and lights (night fishing).

Each of the decisions you make impacts the experience. Some will impact the final experience more than others, but each individual decision *will* impact the outcome; and, obviously, you need to make decisions in a particular order. These and other factors govern the circumstances under which your decision making and preparation make sense – and without them, you have every likelihood of making a wrong assumption, resulting in wasted planning time, wasted money, and bringing about a sub-optimal experience for all involved.

One way to think about it is by drawing an analogy with the laws of physics that govern how our universe operates. We know that light moves at a constant speed in a vacuum (*c*=299,792,458 m/s). We know that the force between two bodies is governed by the product of their masses over the square of the distance between them multiplied by the gravitational constant (*G*=6.67384(80)×10−11 kg−1 m3 s−2). But what would the universe be like if these values were subtly different?

It'd be totally different – fundamentally different. Even a small change to the underlying laws and parameters of the universe would result in a vastly different universe. If the speed of light were faster, (aside from other more pronounced effects) it would impact magnetism, perhaps leading to an earth without magnetic poles (see `https://worldbuilding.stackexchange.com/questions/10126/what-if-the-speed-of-light-were-100-times-higher` for more details). If the gravitational constant were subtly smaller or larger, the universe as we know it couldn't exist at all. All these values define the context of what we can perceive in the universe.

The point is, even a very small change to the underlying "context" means something totally different – and a minor change at that most basic level means outcomes that are vastly and fundamentally different from each other. Just like the structure of the universe would be vastly different with even a minute tweak to the underlying constants of gravitation and speed of light, so too will designs that are viable – even optimal – in one organization be total non-starters in others based on the fundamental assumptions and drivers upon which the business is built. A minor, subtle difference, misunderstanding, or "tweak" of the most fundamental contextual parameters means a completely different outcome and set of constraints. As architects, therefore, it is up to us to identify, understand, and abide by these fundamental assumptions that govern the world in which we will operate.

There are the "laws" that govern how your cybersecurity architecture "universe" operates that are just like this: contextual information that governs each decision that we'll make. These represent the fundamental assumptions and constraints under which design decisions make sense and are viable. If you've been in an organization for a period of time, many of them are likely to be so ingrained to you as to be relatively invisible. For example, if I've spent my whole career as a software developer working in a shop that deals exclusively in writing *Java* code, it might not occur to me that it might be advantageous to write in Python.

Either way, we need to know what these things are so that we can design around them and ensure that the designs we're proposing make sense for the organization into which they'll be deployed.

Establishing the context for designs

"Bosch (the power tools company) was addressing how to market their products. They started to examine how they market and how they might do it better. They realized in the course of that that they don't provide "drills," they provide the capability to rapidly and reliably make holes of a certain depth, width, and quality. What their customers want is to make holes: the drill is just a tool to get them there. This way of looking at capability carries across to technology: the "why" isn't about the technology – it's instead about the business capability. This is what's most important."

– John Sherwood, chief architect, thought leader, and co-founder of The SABSA Institute

This section is all about identifying, breaking down, and systematically cataloging the fundamental assumptions, design constraints, and other immutable factors that will govern what designs are possible. In other words, it's about laying out the "universal laws" that will govern how designs must operate.

Believe it or not, this is uniquely important to the design process. The Roman architect Marcus Vitruvius Pollio, in his seminal 20-volume work *De architecture* (*Of architecture*), set out three fundamental principles of architecture (see *Hicky Morris Morgan*, *translator. Vitruvius: The Ten Books on Architecture*, *by Marcus Vitruvius Pollio*, *Harvard University Press*):

- Firmatis (durability)
- Utilitas (utility)
- Venustatis (beauty)

These principles are as valid to us today as they were 2,000 years ago when the book was written. They are also almost entirely dictated by the context in which the structure (in our case, the structure of the security mechanisms safeguarding our organization) will be employed. For our purposes, we will be most concerned with the first two of Vitruvius' principles (durability and utility) – both of these are tightly bound to the organization since the "durability" of the solution will vary depending on the context within which the solution will operate, and "utility" will depend on how the solution will be employed and how best it meets the needs of those using it.

This chapter will therefore focus on establishing the context for your designs. To do this, we will start by outlining the three most critical areas that will influence your planning:

- Organizational goals
- Existing structures and documents
- Risk management and compliance

The first item, organizational goals, is really what defines this chapter, meaning if there's one single thing that is most critical to architecture design, it's this: the organization's goals. Understanding the goals is necessary because it is key to understanding what is important to the organization, and this in turn informs us what – and how – to defend to best ensure those goals are met.

If it were practical to do so, we might stop and look solely and exclusively at the goals of the organization as the single, pure, and unerring contextual framework upon which we would build our design. This means they would be the only, sole source of information to guide our design work. We'll explain strategies to do this, but in reality, it is often not practical to do it this way, mostly because the time and energy to do so (and do it well) often eclipses what time we have available. This is because a full mapping of all the enterprise's goals, including extrapolating from them the security requirements that will dictate our architectural planning, can take more time than we usually have available to us. Therefore, we employ a bit of a "shortcut" in our approach to understand the organization's goals.

That shortcut entails looking at places where goals of the organization may already be codified, implied, or otherwise detailed in some written form. For example, "governance structures" such as policy, procedures, guidance, and technical standards provide a roadmap that we can use to understand better the goals of the organization. Because they represent decision points already made by the organization, we can intuit that they are themselves steeped in – and reflective of – the goals of the organization, but exist in a "pre-rendered" format that we don't have to invest time in to discern.

In keeping with the approach that we will follow throughout, we will explain the *why* of these items first. Then, once it is clear why we are examining these things, we will explain how we can use them. By the time we get to the description of *how*, the process will seem, if not entirely self-evident, at least significantly easier to accomplish than would be the case if we merely gave you a prescriptive "recipe" to follow.

Now that we've established the basic philosophy with which we will approach this chapter, we'll move on to the most important aspect – understanding the goals of the company/institution and why they are important when creating security goals.

Understanding goals

"The most important piece of architecture is to understand the why: why it is that you are doing what it is that you are doing. Understanding the why leads you to the how. Understand it in the context of the broader business and organization goals context and let that be the guide to when and how you implement security."

– Ted Ipsen, president and COO of Positroniq, LLC

It is a truism that the work of the architect must start and end with enabling the organization to accomplish its goals. Security is not an end in and of itself – it doesn't *operate in a vacuum*. This means it is only useful in the furtherance of some other goal that an organization or individual has.

You can prove this is the case by considering what security controls you'd use if there no threats to defend against. For example, would you use antivirus software if malware didn't exist? Would you hire armed guards to protect an empty room? Of course not. Without a goal – that is, something to protect – security controls have no point.

Therefore, security is only useful to the extent that it is attached to some mission that the organization has: some motivating factor that causes it to be important in the first place. In a business context, the mission (among other things) usually involves the business being competitive, profitable, able to undertake new activities, enabling the workforce, positioning itself the way it wants, and minimizing risk while maximizing opportunity. Other organizations (for example, non-commercial entities) might have different missions that are important to them based on what they do, who they are, their context, and numerous other factors.

Understanding the mission of the organization is therefore the root from which all security measures and activities spring. Most of us don't think about it in this way often, but we can understand and evaluate the mission of an organization systematically. By this, we mean that we can trace each and every security outcome, goal, requirement, and practice back to the organization's *first principles* – the mission of the organization and why it exists in the first place.

There are multiple ways to do this, but one approach is to start with the enterprise goals and examine them for the *success factors* or *enablers* that are required for those goals to be realized. This means we identify one of many high-level goals, we understand what that high-level goal involves (that is, the reason for it), we understand the implementation strategy of the organization for reaching the goal, and we describe the technology, security, and practical or other factors required for the implementation strategy to succeed.

As an example of what we mean by this, say that I have a goal of learning to play the bagpipes. I can't do that unless certain criteria are met. For example, I'd need someone to teach me; I'd need an instrument to practice on; I'd need a location to practice without disturbing others. If I lived on a desert island (without access to an instructor), if I don't have an instrument, or if the only available space to practice is in a busy library, there's fundamentally no path that allows me to achieve the stated goal. In this case, the goal (learning the bagpipes) is supported by multiple success factors that allow it to be possible; those are access to an instructor, an instrument, and a quiet location to practice, in addition to numerous others that we did not list here.

This same principle applies to technology too. Specifically, technology itself is only useful to the extent that it serves a higher business goal. Would a company invest millions of dollars in an application that adds no value? No. Would they buy a multi-million-dollar supercomputer and store it unused in a closet? Also no. Therefore, technology itself is an implementation strategy for making possible a bigger plan that is itself, in turn, a success factor for a larger business goal. You might refer to this as a "technology goal": a part of a larger plan that feeds into and supports a business goal.

For example, consider Facebook. Facebook as an organization might have the business goal of making money. That larger goal is supported by the implementation strategy of selling advertising on a content-sharing platform. These are true at the broadest level, but how (specifically) will the organization put it into practice? Crafting an answer spawns its own new set of goals that are themselves in support of the larger one. For example, one way that the folks at Facebook might conclude that they can achieve the outcome they want is via advertising sales (economic business goal) as realized by them building and providing a website where people can share content with each other (technology implementation strategy).

This website, itself an implementation strategy for the higher-level business, becomes its own goal (technology goal) supported by implementation strategies of its own (that is, the implementation that will allow them to create, deliver, and host that website).

You'll notice two things about this. First, this technology goal (a website) is only one of many possible technology goals that stem from the high-level business goal. There are likely to be others. In the preceding example of Facebook, you'll notice that there are other things they can do to help realize their business goal; they might choose to also have a mobile app, acquire other companies such as Instagram, or implement games or other technology that ties into their overarching business goal. The specific technology goal (again itself an implementation strategy for the business goal) of "providing a website to share content" in turn has its own corresponding success factors and subgoals. These get more specific, such as requiring users to have their own space where they can upload content, being able to access the website from a variety of platforms, and so on.

These technology goals and success factors branch off yet again to be supported by security goals. Specifically, those security goals support technology by describing things that need to be in place to support the usage of that technology securely. Going back once again to the Facebook example, providing a user with space to share content means ensuring that only authorized users can access or edit that content. Therefore, we can see how a security mechanism we're all familiar with (authentication and authorization) supports technology goals directly – and business goals indirectly (that is, by supporting technology goals that are themselves supporting larger business goals):

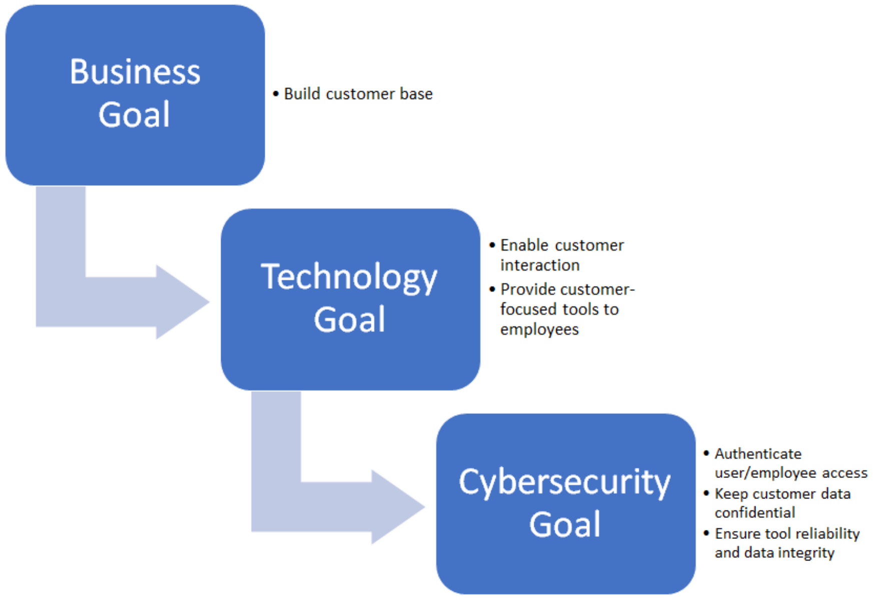
Business Goal
• Build customer base
Technology Goal
• Enable customer interaction
• Provide customer-focused tools to employees
Cybersecurity Goal
• Authenticate user/employee access
• Keep customer data confidential
• Ensure tool reliability and data integrity

Figure 2.1 – Cybersecurity goals derive from business goals

Starting from the top down, you could map out most, if not all, of the security goals and necessary functionality that will represent the "universe" of items that are important to us (security requirements) in our security architecture. By "cascading" each of the organizational goals to the implementation strategies that support it, and using the resulting subgoals to provide input to the next layer in a pyramid-like fashion, this would (should we complete this exercise in its entirety) result in a complete list of all the technology and security goals that our architectures will need to support.

This is not a new methodology, nor is it unique to this book. Those familiar with it might recognize that this is a security-focused view into the COBIT 5 "goals cascade" (*COBIT 5: A Business Framework for the Governance and Management of Enterprise IT. Pages 17-20. ISACA*), which is, in turn, based on research from the University of Antwerp. The goals cascade begins with identifying the high-level business goals of the organization; it derives technology goals from these goals, and then security goals from them. This means that there is a continuous chain from the highest organizational goals down to each and every technology and security outcome desired to support the organization's mission.

To do this systematically, the process implied by COBIT 5 involves (at least theoretically) the following steps:

1. Understand enterprise goals.
2. Derive organizational success factors *(organizational enablers)*.
3. Derive technology goals.
4. Derive technology success factors (*technology enablers*).
5. Derive security goals.
6. Derive security success factors (*security enablers*).

Now, as you can imagine, you could spend quite a bit of time going through this process completely, fully, and thoughtfully for an entire organization. And, in fact, there is benefit in doing so. However, we need to balance completeness and depth of understanding with realistic practicality and the time we can realistically allot to it. This means such an effort – to do it well and completely for the level of understanding we need – would take weeks or even months; for a large organization, it might even take years. This might be a great long-term project if you have infinite time, but we're going to assume that you probably don't.

Instead, we're going to abbreviate this process somewhat. Specifically, any organization you're likely to realistically work with will have documentation already written that codifies – or at least implies – many of these things implicitly or explicitly. This means there will be some documents (usually in the form of policies, procedures, standards, and guidance) that either directly or indirectly state security goals. For example, a policy statement such as "passwords must be a minimum of 10 characters in length" tells us something directly about security goals, while a statement such as "log all access to confidential data" tells us something indirectly. To the extent that these are recorded already, we can leverage them to save ourselves quite a bit of work in building our baseline understanding of what's important.

Secondly, there are assumptions that we can make based on how things normally work that can also save us some time, meaning we can assume things about the technology landscape that are likely to be true even if we haven't specifically mapped out each and every enterprise goal that they in turn support. For example, I haven't ever been to Montana, but I can assume that people there drive on the same side of the road as the rest of the US. I can further assume that the roads there are made of tarmac and that cars in Montana have tires made of rubber. So, while I haven't done the legwork of mapping out the specific mechanics of *why* these things are true, I can (correctly) conclude that they are based on what I know to be true in other regions.

It's important, though, in doing this that we recognize these things for the shortcut that they are. This is true for two reasons. First, there can be times where drawing upon policy documentation can lead us astray. For example, we all know that there are times when security policy is either not followed in practice, doesn't reflect the actuality of what the organization actually does, or is silent on topics that are nevertheless critical to the organization. Second, sometimes there can be things that can be almost universally true for most organizations but where "unicorn" organizations exist where that thing isn't true. Maybe there's some unique factor in that organization where what's usually true isn't for them. So, if we're going to shortcut the work for timing and practicality reasons, we need to do so consciously so that we can be alert for situations where our doing so can lead us astray.

Identifying business goals

For those completely new to an architecture role, it is useful to provide a bit of context around how to identify the business goals of an organization in the first place and how to recognize them for what they are when we come across them in written materials. There are techniques that we can apply to do both.

Essentially, at the highest level, business goals are those goals of the organization that are definitional – that is, the reasons why the organization exists in the first place. For a commercial entity, these might include profitability, shareholder value, or return on financial investment. More tactical goals extend from these as the organization makes decisions about what kind of organization it will be, and how it will realize its core objectives.

The first method that we'll describe to identify what those are begins with a method that is attributed to Toyota as part of quality improvements made decades ago: the *seven whys*. The *seven whys* technique is essentially a strategy for *root cause* analysis – that is, for determining the root cause of something observable. It operates by asking the question *why* until the wellspring or source of an observation is identified.

For example, say your organization has a policy requiring that personnel be subject to background screening on hire. When we encounter a statement like this one, we can use the *seven whys* technique to arrive at the core, most-fundamental organizational goal that causes it to exist in the first place. Take the following example:

- **Observation**: *The organization requires criminal background screening for new employees.*
- **Why (do we do this)?** So that we know that the staff we hire are free from felonies or other criminal history.
- **Why (do we care)?** So that we can prove to our customers that our employees are trustworthy.
- **Why (does that matter)?** So that our customers trust that they can do business with us safely.
- **Why (does this help us)?** So that we can remove barriers that would prevent customers from hiring/using our service.
- **Why (do we need these barriers removed)?** So that we can better acquire and retain customers.
- **Why (is this valuable to us)?** So that we are profitable.
- **Why (be profitable)?** So that the organization can provide value back to shareholders.

You'll notice, in doing this, that we've identified the root business goal (shareholder value) as well as a host of other subgoals along the way. For example, the organization wants to be appealing to customers (a business goal supporting profitability) and the organization wants to demonstrate its diligence and commitment to customers, and one part of the implementation strategy to do that involves verifying the criminal history of new associates.

The more you do this, the more you will start to realize that you will come back almost invariably to a relatively small number of core things that the organization cares about. They are the kinds of things that are often found in an organizational mission statement. They are usually very high level and speak to why the organization exists in the first place.

Dimensions of success

"I'm not going to say that technical skills don't matter, but the skill that will enable all other skills to come out is retrospection. I don't like to use the term "post-mortem," because in reality nobody has to die for us to learn important lessons. When I teach people to threat model, I tell them to start out with as much time for retrospectives as they spend in their analysis. If you start out cooking and the first time you chop an onion the pieces all come out all different sizes, you can learn and practice until you get better. But often, this might not be the most important skill – a more important skill might be being able to tell if the bottom of the pan is burning for example. Sometimes the skills that are most important are the ones that take the least time to learn."

– Adam Shostack, president of Shostack & Associates

Understanding what the goals are is an important step (arguably the most important step), but it's important to note that it's not the only step in the legwork that we need to do in order to gather the "raw materials" required to understand the architectural universe. Specifically, we also need to understand how the organization measures itself against those goals. With a goal that is concrete and specific, this is straightforward. For example, a high-level financial goal such as *increase profitability* or *provide shareholder value* is easily measured financially – that is, by looking at revenue, expenses, operating costs, and so on. Other goals (for example, social responsibility, providing non-tangible value) can be harder to measure.

Security goals are often particularly difficult to measure against as many of them are probabilistic in nature. For example, a goal such as decreasing the likelihood of something undesirable coming to pass (a breach, for example) is based on the probability that the outcome will occur, rather than something directly measurable in and of itself. This matters because, as we discussed earlier, there are often several different implementation strategies to achieve the same outcome. These strategies will not always be equal in how they do so or how they impact other goals.

As an example of what I mean, consider a software vendor – that is, a company whose business is to develop and sell application software. They might set a goal that all source code is reviewed for security vulnerabilities as they conclude that doing so, in turn, supports business goals such as competitiveness and long-term profitability. One way they might choose to implement this is by using a source code scanning tool (for example, *lint* if they're working in C); another strategy might be hiring a team of experienced C developers to manually audit each and every line of code for the product. These two approaches accomplish the same thing (vet source code for errors), but perform very differently with respect to cost, efficacy, time investment, and so on. Understanding how each goal is measured informs what strategies are most advantageous to achieving the outcome we want.

There are multiple different dimensions along which we can evaluate a given approach to implementation. In fact, it's arguable that there are a near-infinite number of dimensions along which any given strategy can be evaluated and that each organization might have a different set. However, as a practical matter, there are four that we will consider here:

- Effectiveness
- Maturity
- Efficiency
- Alignment

We'll explain what we mean by each one in detail in the following subsections. Again, we are trying to understand here how the *organization* measures itself and less so how we might measure in isolation. Thus, keep in mind, as we move through this, that we want to understand these dimensions generically, but also in terms of how an organization might employ them as a method of self-measurement.

Effectiveness

The first evaluation criteria that we'll look at is how well the implementation strategy performs at doing what it is designed to do. For business goals, this can be straightforward. If the strategy is designed to make money, how well does it do that? How much money does the organization generate? If a strategy is used to reduce development time, how much time does it remove from the process?

Security goals likewise can be looked at, evaluated, and measured through the lens of effectiveness to the same degree that other types of goals can – that is, how well does the security measure we implement perform at achieving the goal?

With security controls, this dimension is particularly important as controls are not equivalent. In fact, even when security measures are designed with identical outcomes in mind, individual implementations can impact how well they perform. As an example, consider the difference between **Wired Equivalent Privacy** (**WEP**) and **Wi-Fi Protected Access II** (**WPA2**). Both are designed to do the same thing (more or less) from a very high-level point of view: namely, to provide confidentiality of data transmitted over wireless networks. However, they have vastly different characteristics with regard to their utility and security.

Those who are familiar with the history of WEP will know that there are serious security vulnerabilities in WEP. These issues are serious enough that they allow an attacker to passively monitor the network and break the encryption used in a matter of minutes or hours (*Fluher, Scott et al., Weaknesses in the Key Scheduling Algorithm of RC4*). By contrast, WPA2 is the current generally accepted optimal protocol for providing robust confidentiality on a wireless network. Therefore, while they both serve the same underlying security goal, they differ vastly in terms of how well they do it. They are not equally effective.

This, in a nutshell, is effectiveness: the efficacy security measures have at satisfying the goal/requirement – or, their success at delivering what is intended. Any organization that has conducted a security program review that examined what controls they have in place, and provided feedback on those that were not implemented, has likely been measured along this axis.

Maturity

The second dimension to be aware of is the maturity of implementation. This is particularly true when looking at security measures that have a necessary underlying procedural component.

Note that by *maturity* here, we don't just mean how long something has existed (that is, how old it is chronologically) or its acceptance in the industry (that is, the *maturity* of a technology). These things can be important, too, in some cases, but instead, we're referring to something else: the reproducibility and reliability of the processes that support the implementation. This is the same sense of maturity that is used by frameworks such as **Capability Maturity Model Integration** (**CMMI**), developed by Carnegie Mellon (now stewarded by the CMMI Institute) for understanding software development process maturity (*CMMI® for Development, Version 1.3. Software Engineering Institute, Carnegie Mellon University*).

Two security measures, both themselves designed to fulfill a particular niche and achieve a particular security goal, can have very different maturity characteristics. For example, consider the respective incident response processes at two hypothetical firms. In the first organization, there is no written process, no case management or other support software, no automation, and no metrics collected about performance. In the second, they have a well-documented and highly automated process where metrics about performance are collected and improvements to the process are made over time based on those metrics.

In both cases, the function is the same: incident response. The two processes might even (on the whole) be equally effective (on average) at servicing the incident response needs of the organization. However, one organization employs an *immature* process to satisfy these needs: it is *ad hoc*, relatively unmanaged, and reactive. Using a scale such as the one contained in the CMMI, you might call their process *initial* (level 1) on the maturity spectrum. The second organization has a much more "mature" process: it's reproducible, consistent, and managed. Depending on the degree of ongoing improvement and optimization, it might fall into the *quantitatively managed* or *optimizing* maturity levels (level 4 or 5, respectively, in the CMMI model).

Again, even if the goal is the same – for example, the intent and function are the same and even the efficacy and performance are the same – maturity of implementation can vary. There are advantages to using processes that have higher maturity. For example, the more mature a process is, the more likely it is to have consistency in how it is performed each time, the more easily the process can recover from interruptions such as the attrition of key personnel, and the easier it can be to measure and optimize. There are, though, as you might assume, potential budget and time investments that may be required to bring a process from a lower state of maturity to a higher one. As a consequence, the maturity of implementation might be valuable for the organization to target in its processes.

Efficiency

Another dimension that matters for control implementation is the efficiency of operation. Earlier in this chapter, we used the example of two ways to implement application source code analysis: software testing versus manual code review. We alluded to the fact that these two approaches each have different dynamics and characteristics. One of the places where they diverge significantly is in overall cost and efficiency – both in terms of dollars spent as well as in the time it takes staff to perform the tasks involved.

This can be true of any security implementation. Measures might perform similarly – or even equivalently – in terms of effectiveness and/or maturity, but still have very different financial cost or time investment requirements.

To make this clearer, consider spam monitoring as an example of how this can be true. Say that, to maximize staff time and prevent phishing attempts, we want to filter out suspicious emails, spam, or other potentially unsolicited or undesirable emails. One approach to do this might be to employ automated filtering; another approach might be to hire a team of people to read all incoming emails looking for spam. Assuming for a minute that each approach was equally likely to catch inbound spam, obviously, there are advantages to the automated approach. Putting aside the obvious privacy implications of a team of strangers reading your emails, such an approach would also cost significantly more, both in time and dollars.

This is what we mean by efficiency in this context. You might alternatively refer to this as "cost-effectiveness" for organizations such as commercial companies, where staff time and dollars spent are functions of each other. Since this is not always the case (for example, in educational institutions), we thought "efficiency" was a more descriptive word choice.

This is a particularly important metric for the security architect when it comes to implementing security measures. Why? Because any security measure we put in place comes with an opportunity cost. Assuming resources are constrained (that is, that you don't have an infinite budget or unlimited staff), every measure you put in place comes at the cost of something else you didn't do – that is, what you could have done instead but didn't because you went down the specific implementation path you did.

For example, if you dedicate your entire staff to one security measure (having them manually filter inbound email for spam, for example), there are other security measures you can't implement because those resources are engaged elsewhere. Since staff can't do two things at once, the opportunity cost for the path you chose is whatever those resources would be doing instead if their time wasn't completely occupied. Likewise, if you use your whole budget on one very expensive control, you won't have budget left over for other controls that could also be useful.

Alignment

The last dimension that we'll look at here is what you might call **alignment** with organizational culture and skills. There are times where what you do – or how you do it – will be influenced by other factors. As an example, say that I wanted to watch a movie that isn't available for streaming and is only available on DVD format. If I don't own a DVD player, it really doesn't matter how good the movie is – or how much I want to see it. Until I either buy a DVD player or the movie is released in a format that I have access to, I won't be able to watch it.

This matters with security measures too. For example, if I don't have access to staff with the right skillset to maintain or operate a security measure, or if, for some other reason (such as culture), the measure would be intolerable or untenable to the organization, it becomes a less compelling choice.

An example that will be familiar to many security practitioners is the forensic examination of compromised systems. As we all know, there is quite a bit of specialized expertise that goes into ensuring courtroom admissibility of evidence gathered during an investigation. We need to preserve the chain of custody, collect evidence in a way that doesn't corrupt the crime scene, be careful to prevent the possibility of writing to source media, and so on. Most organizations, unless they specialize in forensics, don't have the staff to do this themselves. It's not that they couldn't acquire those staff (in fact, some larger organizations do), train them, or keep their skills current. Rather, they choose to seek support from outside specialists in the event that such capability is needed because maintaining that skill base can be expensive relative to the amount of time that they will be directly needed.

In that example, a security measure that specifically requires specialized forensics skills to operate would not be a great fit for an organization that has chosen to outsource that specialization. It's not that either approach is right or wrong; it's just a question of whether it aligns with other choices the organization has made. In the same way that purchasing software that runs solely on OS X is a non-optimal choice for an environment that is Windows-only, this security measure requires something that the organization doesn't have access to.

With this new footing in understanding goals and understanding the dimensions of success, we can embark on a quick journey through the policies, procedures, and standards that will help ease the identification of organization goals.

Structures and documents

The dimensions described previously relate both to the goals of the organization (what is important to measure in how it achieves those goals) and also to how well a security measure works within the context of enterprise goals. This is what makes understanding the goals so important. But assuming that going through a full goal-mapping exercise to correlate every organizational goal to technology goals – and map, in turn, security goals to technology ones – represents a time investment that not every architect can afford to make, how can we get to a rapid understanding quickly so that our design work can proceed?

As we implied earlier, one way to do this is by looking at policy, procedure, standards, and guidance documentation that may already exist in the organization. This is because they themselves are the codification of already-made decisions by the organization, which are, in turn, driven by the goals. Sometimes, these items are even referred to as "governance structures" – meaning they are the metaphorical scaffolding upon which governance for the organization rests.

Policies, procedures, and standards

Whatever you call them, let's walk through what these documents are and why they represent a useful shortcut for helping us to define the landscape:

- Policy
- Procedures
- Standards
- Guidance

We'll go through each one, discuss why and how each one is valuable to us, the architect, and discuss ways that you can employ them to help distill down management intent to organizational goals.

Policy

"A mistake people make is conflating policies, standards, procedures, and guidelines. These are all different instruments. Policies are an articulation of the strategic goals of management; they should be high level, and strategic in nature. Technical standards are more closely tied to technologies that you have – or processes that you run in the organization. These may change more often since the underlying technology or processes could change. Procedures are specific actions – step by step instructions – that you take to accomplish a particular goal in achieving a policy goal; these may change even more than standards. Having each of these documents in place at the right level that cover the right things is important for the organization."

– Ted Ipsen, president and COO of Positroniq, LLC

The first area we'll look at is organizational policy: both security policy, but also policy more generally. Sometimes described as "statements of management intent" (*Wood, Charles Cresson, Information Security Roles and Responsibilities Made Easy. Page 63, PentaSafe*), these are documents created by the organization to document and codify the expectations of management in reference to a particular topic.

Policies are typically *high level* in nature, specifically approved by management, and (typically) periodically reviewed to ensure accuracy, completeness, and to ensure they stay current. In practice though, these things aren't always a given. From a level of specificity standpoint, they can run the gamut between the more and the less technical, and from more to less specific. For example, one organization might have a policy that states "information should remain protected when transmitted over corporate networks"; another organization's policy might stipulate that "data on the internal network must be transmitted using TLS 1.2 or higher." Both speak to the same underlying goal – that is, keeping information protected from prying eyes when traversing the network – but they do so with varying levels of prescriptiveness and technical specificity.

While practitioners can, and sometimes do, feel strongly about what the "optimal" level of detail or technical specificity to include in a policy document like this is, the truth is that some are more specific, and some are less so. Either way, because policies are approved by management and are highly visible within the organization (thereby engendering input from relevant stakeholders), they represent a data source summarizing views by management and other stakeholders about the topic the policy addresses.

This means that policy represents a particularly fruitful area for understanding the goals of the enterprise. Why? Because everything contained in that policy is a codification of management intent. For the management of the organization to put a "stake in the ground" about a topic means that, at some level, something about that particular topic relates to something the organization wants to do. In other words, there's no reason to write a policy about something if the topic is of no consequence.

So, how do we derive the organizational goals from policy? One method is to use the *seven whys* technique discussed earlier, meaning, trace the root cause of the policy back to the overarching business goal it supports. Another way is to understand the "intent and rigor" of a policy, meaning what is intended, how much is required, and why.

Procedures

Unlike policy, procedures describe the process that will be used to support a given policy objective. They may or may not require explicit approval by executive leadership, but they are typically very technically specific: at least specific enough to allow constituents within the organization (for example, employees or contractors) to perform the process the same way each time it is executed.

Procedures are useful to the security architect in helping us understand the lay of the land, but arguably less valuable in doing this than policy documentation. Why? There are two reasons. First, procedures can be very specific about details, but the area of coverage can be fairly narrow. For example, a procedure about system access log monitoring might outline in great detail the process by which security team members acquire and review system logs, but it probably won't go into much detail about anything else outside this fairly narrow purview. The scope might target logging for a particular subset of hosts, but not logging more generally (that is, on other systems), other types of monitoring beyond logs, or how the logs will be employed or stored outside the narrow scope of the procedure.

Secondly, procedures often offer less insight into management expectations and intent. For example, consider an incident response process. The process might outline who is responsible for what during an incident, how the incident should be handled given various sets of circumstances, what reporting should occur (for example, frequency, format, and to what audience), when the response team will meet, how they communicate, and so on. But the written procedure might not speak to *why* these things are being done. It is possible (though by no means certain) that it could allude to a subset of goals, but it cannot be relied upon that the entirety of the goals surrounding incident response will be addressed within the procedure itself.

Procedures, therefore, while a useful supplement to data gathered through other means, are perhaps the least direct instrument available to the architect in terms of understanding the universe within which our designs will operate. This is not to imply that we can remain ignorant of them; on the contrary, we need to be aware of them to the extent that the designs we implement will need to work with them, enhance them, and not contravene them. In some circumstances, we might be able to intuit broader security goals from procedures, but in general, policy is better equipped to do this.

Standards

Standards, particularly technical ones, help to specify the manner in which certain goals are to be met. For example, you might have a technical standard that outlines how desktops should be configured, how firewalls are to be deployed, and how cryptography is to be used. Standards are usually narrow in scope/focus and are prescriptive.

Normally, the purpose of a standard is to document the measures required to adhere to a given policy. For example, an organization might have a higher-level policy such as "encryption should be used to protect data when transmitted." But what does that mean? How, specifically, can those to whom the policy applies adhere to it? One mechanism that organizations can employ to provide additional specificity and to assist in overall compliance with the policy is to provide details about *how* to accomplish it.

In the preceding example, an organization might have a few supporting technical standards to support the "encrypt data in transit" requirement such as, for example, the following:

- SSH server and client configuration standards outlining the standard for how to employ secure shell (SSH) and how to ensure that both clients and servers of SSH connections are configured in an appropriate, secure manner
- Web server TLS configuration standard outlining how to enable and configure TLS capability for web servers
- VPN configuration standard outlining what VPN products are to be used and how to configure them

These are only a subset of the possible standards that could support a policy requirement such as the one outlined. You could also have additional standards governing everything from database connections, to proprietary or non-standard communication channels, to secure Wi-Fi, and countless other examples.

As was the case with procedures, when trying to understand the "universe" within which our designs will operate, standards have some utility but provide less direct value to us than policies do. The primary reason is that they also don't always provide new information about management intent.

There are two reasons why this is so. First, as you can imagine, standards are only very seldom written by management directly. Instead, they are normally authored by those subject matter experts who are knowledgeable about the topic under consideration. In the preceding example, can you imagine someone on the executive team, the CFO, say, authoring something called the "TLS Configuration Standard"? No. Instead, you'd expect a document such as this one to refer to specific cipher suites (that is, what algorithms should be used), methods of key management, specific software or hardware configuration parameters, and so on. Something this technically specific isn't – and shouldn't be – the primary focus of someone in the executive suite.

The second reason they're not the best sources of management intent is that they already support an existing policy item, meaning a well-written standard will directly support existing policy where the management intent is already spelled out. Generally, standards actualize an intent; they don't specify new intentions not already referred to in the policy.

Guidance

The final type of document we might encounter is guidance documentation. This type of document represents additional, non-prescriptive information provided to those in the organization about policy, procedures, or standards.

Going back to the earlier example about encrypting data in transit, an organization might have a technical standard in place to help support that, such as configuration guides, or a list of approved cryptographic products or standards. They might also provide guidance to be used in certain situations – for example, if they have in-house developers who need to implement cryptographic functionality themselves. In that case, there might not be a direct configuration standard that is applicable, but they still wish to pass along advice that could be useful in achieving the policy.

As with standards, guidance documents usually exist in service to already existing policy. Therefore, they can be helpful in providing additional details about a policy, but are less directly useful in ferreting out additional, new information about management expectations that would not already be available in the governing policy.

Applying to architectural frameworks

The very first thing that an architect will need to do is establish what the governing philosophies, needs, goals, and other immutable factors of the organization are that will inform their designs.

In an ideal world, this would come through reasoned, systematic, and comprehensive analysis of the enterprise's goals, meaning by deconstructing each and every goal that the organization has, tracing out how they intend to achieve these objectives, and then building out the implicit security goals that support their current or proposed use of technology. In reality, a complete goal-mapping exercise consumes more time than most practitioners are able to invest before beginning a security design. Therefore, we instead begin by looking at what is already codified to begin the process.

This means we begin with the standards, procedures, policy, and guidance that the organization has already written to establish the *rules* for how our security designs must behave, the optimal character of what they should contain, and that make the most sense in light of how the organization already functions.

The most straightforward way to do this is to start by reading the documentation (assuming you have not already done so). By this, I don't necessarily mean that you need to read everything the organization has ever written. An organization (particularly a large one) might have lots of policies and procedures; these might cover everything from time off and attendance to how they handle whistleblowers and ethics violations. However, at a minimum, you'll want to read the security-relevant policy, procedure, guidance, and standard documentation that exist and use them to derive the information about the organizational goals that will inform your work going forward.

This advice isn't exactly rocket science, but following it provides a few valuable things to the architect. First, it lets you get comfortable with the general approach to security. Is this the kind of organization that leaves room for flexibility and creativity on the part of individual employees? Are they risk-tolerant or risk-averse? Do they tend to be more prescriptive in terms of how to accomplish particular goals or do they leave room for individual employees or groups to innovate?

The second thing reading the policy will do is tip you off to the main things that your designs will need to accomplish. It can be helpful to make a list of these things as you encounter them. If the organization requires encryption of data at rest, for example, your designs will need to implement this, so remembering what these things are is an absolute must. Since it is the case that there is often extraneous information in the policy documents themselves, it can be helpful to create a list of "facts" outside of the policy documentation that designs will need to address.

How will you know when you've identified organizational goals? You'll know you've reached one when you get down to a clear, actionable statement that answers the question, "Why does this organization exist in the first place?" Examples of foundational organizational goals might be the following:

- **Shareholder or owner value**: A commercial enterprise may exist for the purpose of providing an economic return to the owners or shareholders.
- **Provide value to a given constituency**: A socially conscious organization might exist to provide benefits or value to a given community.
- **Impact the world in some way**: Some organizations might exist to bring about a concrete change to the world or society – for example, curing a disease, researching a given topic, contributing to the betterment of humankind, or other specific outcomes.

In addition to foundational goals, you might also encounter business goals that themselves spring from these foundational goals, as follows:

- **Maximize efficiency**: An organization might seek to streamline operations. This goal itself is not an *end state* in the foundational sense such as the previous ones. Instead, this goal helps them do something else better. For example, a commercial enterprise might seek to increase efficiency so as to maximize profitability, or a non-profit might seek to maximize efficiency to have more resources to apply to its core mission. In both cases, the higher-level goal is supported by being more efficient.
- **Reduce greenhouse gas emissions**: The organization might strive to be green in its approach to impact on the climate. Note that this supports a broader goal, such as "impacting the world" instead of being an end goal in itself.
- **Expand market share**: Again, this is not an end state in the same way as the foundational goals are. After all, why increase the organization's market if there is no broader goal that is supported?
- **Enhance customer experience**: An organization might seek to improve how it interacts with its customers, either by providing better customer service, a better overall experience with the customer, or numerous other ways. The motivation might be to make customers "stickier" (that is, less likely to switch to a competitor), to improve market share, or for other reasons.

These are only a few examples of many possibilities. In fact, the farther you go down the chain of an organization's goals, the more subgoals you will find. You'll notice, as we outlined earlier in this chapter, that the supporting goals are actually implementation strategies for how to achieve the broader-level goals. In the preceding examples, the goal of improving customer experience is one possible implementation strategy for increasing market share (that is, by improving the customer experience, fewer customers are lost to attrition), which itself is a possible implementation strategy for the foundational goal of increasing profitability.

Now that we've become familiar with the documentation and how it informs our planning, we are ready to take our work one step higher. The next section will take us through risk management and compliance, in which we will try to plug all the holes in our security planning thus far.

Risk management and compliance

If you limit yourself only to looking at what is explicitly documented in the organization already in the form of policies, procedures, and supporting documentation, you'll find that you have a good picture, but not a complete one. At this stage, the picture is incomplete in two ways. First, it's incomplete because there may be other security objectives that are themselves either presupposed (and therefore not mentioned in those documents explicitly) or that are unrealized by the authors of that documentation themselves.

Therefore, as you work to enhance your knowledge of the organization, you can round out your understanding based on your own direct experiences, institutional knowledge gained through familiarity, and other documentation that may speak to the organization's goals. You can also, where they exist, look to other sources – in particular, two critical sources – to help round out your understanding. These sources are the following:

- Risk appetite
- Compliance requirements

We'll explain in more detail what each of these factors is, why they matter, and how to evaluate them.

Risk management and appetite

"Ultimately, architecture is a risk management process. It tells you what to do based on the priorities of risks that are in your sphere of influence. You use those risks to come up with the right set of controls. These controls go into your architecture all the way down: they drive technology standards, process standards, vendor selection, all the way down from the business to the lowest level. Your architecture then becomes an enabler of more effective risk management so that you're maintaining your "architecture vitality" – the property that, as the business is changing and adapting, you are too."

– Andrew S. Townley, CEO at Archistry Incorporated

Not every organization views risk the same way – just like individuals don't always view risk the same way. As an example, consider your retirement savings. If you're young, you might be fairly accepting of some risk in your retirement portfolio. Why? Because you have time to make up losses down the road if things go south. Since you're young, it might be worth it to take on a little bit more risk now since you know that, over the long term, a potential downtick in the economy will likely be erased by the long-term gains that you will realize from a more aggressive posture. In this case, you are fairly risk-tolerant.

If, however, you are closer to retirement age, it would have a correspondingly greater impact on you if you were to lose a substantial portion of your retirement savings in the very short term. In this case, you might not have many working years left, so you have limited time during which to recoup any short-term losses. Therefore, you are more risk-averse.

Neither approach is right or wrong. However, the circumstances dictate what is right or wrong *for you*. This means, depending on those circumstances, that you are more or less tolerant of incurring risks.

Organizations, just like individuals, can have varying risk tolerance too. A conservative financial institution, for example, might be willing to accept less risk than a four-person start-up. Just as with individuals and their relative career timeline, this too is driven by circumstance. In the case of the bank, they might have spent years building a brand image of stability and reliability. They might have invested countless dollars and person hours over the years in making sure that that image is reflected in everything they do. The start-up probably has less invested in their public image. Likewise, a small start-up might be nimbler than a large organization such as a bank or brokerage and be able to rapidly reposition in the event that they incur a financial loss or impact on their image. In this case, as you'd expect, the start-up is likely more risk-tolerant than the bank.

It is very important to have a solid understanding of the risk tolerance of the organization within which you will be designing your security architectures. Why? Because the amount of risk that the organization is willing to assume will dictate what your designs need to accomplish in terms of risk reduction, which in turn drives what measures you will put in place, how you will implement them, and the amount of money the organization is likely to be willing to spend in so doing.

How can you know what the organization's risk tolerance is? In an organization that has a process for formal, systematic risk management, this will be straightforward. Usually, a statement of risk tolerance is created as input into the risk management process, meaning it will be documented clearly and unambiguously.

This happens because risk tolerance is key to what risk management is conceptually. Risk management is the formalized process of "optimizing" risk. In practical terms, this usually means lowering it – that is, reducing the risk to a level that is within the defined organizational risk tolerances. Therefore, to understand what the tolerances are, the organization will need to have a fairly clear understanding of what level of risk is acceptable to them so that they know whether they have achieved their risk optimization goals.

To accomplish this, risk management techniques usually employ several steps. For example, these are the steps as outlined in ISO 31000:2018 (the international standard outlining guidelines for risk management) (*Risk Management – Guidelines (ISO Standard No. 31000). Retrieved from* `https://www.iso.org/obp/ui#iso:std:iso:31000:ed-2:v1:en`):

- **Establish context**: Identify and outline factors to be taken into consideration during the risk management process.
- **Risk identification**: Identify potential risk sources.
- **Risk analysis**: Analyze the risk, including developing an understanding of consequences, likelihood, and other factors.
- **Risk evaluation**: Triage, prioritize, and assign priority to mitigation or other treatment.
- **Risk treatment**: Address the risk through mitigation (remediation), acceptance, transference, avoidance, or other measures.
- **Monitoring and review**: Monitor the risk over time to ensure that it stays within acceptable parameters.

These steps can be either quantitative (that is, where numeric values are used to explicitly quantify impact and likelihood) or qualitative (where more *fuzzy* values such as low/medium/high are used). Either way, though, part of establishing the context must involve understanding the risk tolerances (risk *appetite*) that are acceptable to the organization.

Those new to the security profession often question why we don't always just seek to reduce risk to its minimum possible value in every situation. There are a few reasons not to do that.

First, it's not always the case that lower risk is always good and higher risk is always bad. In some situations, risk can actually be a good thing. Consider again your retirement portfolio. What would happen if you always put your retirement savings in the absolute lowest risk investment (for example, precious metals)? You'd find that while your retirement funds are certainly safe, they wouldn't grow very quickly. By undertaking risk carefully, with due consideration, and in an informed, controlled, calculated way, you can potentially realize a higher return. This same issue is true in business as well. Sometimes, endeavors are inherently risky: the acquisition of another company, the decision to release a new product, or a provocative marketing campaign. Instead of trying to completely remove all risk, instead, business is about taking careful, studied, calculated risks.

Likewise, there can be risks in not taking action just as there are in taking action. For example, consider what would happen if a business failed to adopt some key technology – email, perhaps – because of the risks associated with spam and phishing? Now consider what the competitiveness of that firm is likely to be when that firm decides against email but their competitors all adopt it? They would be competing at a handicap relative to their peers.

Second, there is always a cost. Even the most efficient and cost-effective security measures incur some cost. In many cases, the costs involved far outweigh what the organization is willing to spend to achieve a risk reduction goal. Most of us probably wouldn't say no to someone offering us a million dollars. But what if, in order to obtain that money, we would need to spend the rest of our life in a sensory deprivation tank with no ability to spend it? Most people would conclude that the price associated with earning that money is way too high. Therefore, trying to achieve "zero risk at any cost" is untenable.

Really, then, the goal of risk management is to understand risk so that the decisions you are making are informed ones.

For organizations that do employ formalized risk management processes, architects will typically interact with that process in at least two ways. First, they will use the information coming from the risk management process – information such as risk appetite, as we highlighted previously – to inform their work. Secondly, though, their designs will themselves be iterative with risk management activities. By this, we mean that they will provide input into the risk management process as they track and manage risk areas that result from their designs (for example, residual risks that may remain in their designs) and also that their designs may themselves be risk mitigation steps for other things resulting from the risk management process.

For organizations that do not employ a formalized risk management process, it is still important that the architect understands the risk tolerances of the organization. In this case, though, it may take some further investigation and analysis to arrive at since it may not have been systematically considered as part of a broader risk planning exercise. In this case, you will want to do some information gathering to attempt to get an understanding of it prior to beginning any design work.

In saying this, we should note that it is almost always better to have executive-approved, management-socialized, and documented risk tolerance guidance upon which to base our architectural decision optimism bias". Current Biology, 6 making. Why? Because it's defensible. There are some people (including some in positions of executive leadership) who will represent themselves as being very risk-tolerant so long as negative potentialities don't come to pass. If (when) they do, you will find that they are much less accepting. This means that even if we examine, analyze, and document a risk tolerance in absence of one already existing, we should still work to get it sanctioned and approved.

To help do so, it can be useful to do two things: first, err on the side of being more risk-averse rather than more risk-tolerant. There is a known, well-studied perception bias in people where they tend to view negative outcomes as less likely than positive ones ("optimism bias") (*Sharot, Tali. "The optimism bias". Current Biology, December 6, 2011. Elsevier Ltd. Retrieved from*: `https://www.sciencedirect.com/science/article/pii/S0960982211011912`). Unless organizations are analyzing risk both systematically and quantitatively, this bias can, and does, tend to cause people to view undesirable, negative outcomes as less likely than positive ones. To arrive at something realistic, therefore, it can be helpful to deliberately compensate for the bias in our planning. Secondly, it is helpful, even if the organization is not doing any formalized risk management, to document the risk tolerances that we will use in our architectural planning. Even better is if we can get those socialized and approved (that is, signed off on) by executive management. Again, this is both for defensibility purposes as well as having a documented artifact to draw on for future reference.

Compliance

"Security architecture in the most general sense is about creating a framework. That framework represents a cohesive model that covers the circumference of the organization's needs to ensure ample coverage for security and compliance concerns. There are obviously numerous dimensions to this, but in essence it's about developing a model that captures what is in the best interests of the organization, aligning organizational discipline to that model, and giving you back something that is measurable, manageable, and provides a solid understanding of the organization's risk posture."

– Dr. Richard Perez, vCISO

The other factor that we will need to consider is compliance. By this, we're referring to any legal, contractual, or other requirements that the organization must adhere to by virtue of the business that it is in, the work it does, or the type of data it processes. Many of these mandates will require specific action on the part of the architect to ensure that they are adhered to.

Let's say, for the sake of argument, that we are working in the healthcare sector for a large, multi-institution health system. Perhaps we have been chartered with developing security architecture for a subset of the health system's ecosystem (for example, a particular hospital in the network) or we are working on security architecture for a healthcare application that will support patients.

In the US, an organization like the one described would be considered a *covered entity* under the **Health Insurance Portability and Accountability Act** (**HIPAA**). This US law outlines a set of requirements that pertain, among other things, to the security and privacy of health information that is used during the course of providing treatment to a patient. The regulatory rules, in this case, contain specific provisions for the encryption of data when transmitted over an electronic communications network (§164.312(e)(2)(ii) – "Implement a mechanism to encrypt electronic protected health information...") and when stored (§164.312(a)(2)(iv) – "Implement a mechanism to encrypt and decrypt electronic protected health information").

These two requirements, therefore – encrypting data at rest and in transit – are non-optional from a design point of view. Ideally, there will be an organizational policy that stipulates that these controls will be used. In fact, for most organizations, there will be. But we cannot assume this. This means we cannot rely on there being a one-to-one mapping between line-item requirements as outlined and the policy of the organization. As such, it is incumbent on us to understand, remain aware of, and incorporate compliance goals into our designs.

It is not possible in a targeted book such as this one to outline every possible set of compliance requirements for every type of organization in every geographic region. There are literally hundreds and thousands of potential regulatory requirements that may apply. Therefore, some data gathering will be required at this stage into the type of operations conducted, and the location where they are conducted.

For example, an organization that retains information about citizens of the EU will fall under the requirements of the **General Data Protection Regulation** (**GDPR**) – an EU law governing data protection and privacy. As in the preceding example, a healthcare provider or insurer in the US will fall under HIPAA – US legislation governing requirements for security and privacy in a healthcare context. A merchant accepting payment cards would need to adhere to the **Payment Card Industry Data Security Standard** (**PCI DSS**) – a set of rules required by card brands for the protection of payment card information by merchants.

These are only a few examples of the many possible requirements, legal or contractual, that might apply to a given organization. Likewise, it is possible that multiple regulations could apply for any given single organization. For example, on the surface, it might seem like a US-based hospital would be primarily concerned with the requirements of HIPAA. They are, of course, including other governing regulations (for example, the HITECH act of 2009) and other potentially applicable standards, such as the Joint Commission's Environment of Care standards, rules from the **College of American Pathologists** (**CAP**) standard governing laboratory environments, and numerous others.

But now, consider other factors that might apply as well outside of this; for example, how many hospitals also accept credit payments at the gift shop, cafeteria, or for parking? In this case, they would also be beholden to the PCI DSS just the same way that a merchant such as Amazon is (though with a correspondingly smaller scope of applicability). Likewise, consider a publicly traded healthcare insurance provider in the US, a merchant in the EU processing credit card payments, and so on.

Since multiple regulations might apply, it is always a good idea to begin by obtaining a thorough understanding of the various regulatory requirements that are in play, again in a systematic way to ensure completeness. In a large organization or one in a heavily regulated industry, one way to do this is by engaging those individuals responsible for compliance oversight and obtaining a list of focus areas. Chances are, in this environment, they will have put significant thought into what requirements are applicable. In a smaller environment, this can prove more challenging and you may find that you (at least initially) need to keep track of this yourself.

If that is the case, one way to approach this is to start with what you know based on what the organization's primary business is, but expect that you will need to, over time, add to the list of requirements that are in scope as you learn more. This means begin with what you know and document the various requirements that are likely to impact your architecture, but do so in a way that allows you the flexibility to add to the list over time.

Establishing a guiding process

"There are "top down" and "bottom up" approaches to architecture. Many people start with the highest level of the organization and try to work down; this seems logical at first, but what can happen is you lose sight of context as you move down the chain toward more specific usage. What I think is a better approach is to start bottom up: understand the threats, understand the context, understand the risks, and build architectures for the different environments piecemeal while keeping an awareness of the "macro" – the high-level view. This lets you reduce redundancy and normalize, but also create lightweight, modular frameworks that can be reused where needed or improved upon and reworked when needed. Anyone can build an architecture that looks great on paper; but an overly structured approach can be so rigid that it fails when it gets to implementation. You need concert between top down and bottom up; the best strategies do both at the same time."

– Steve Orrin, federal CTO at Intel Corporation

Now that we've covered in some detail *what* has the potential to influence your design landscape, the next thing we'll cover is a process you can follow that will result in you having a clear, concise, consolidated view of the important factors that will be used as input into your design process.

Hopefully, having come this far with us in this chapter, you have a pretty good idea of *why* this process is important and what things you will want to look at. Once you understand the why, the "recipe" is probably fairly straightforward. Also, understanding the why means that you'll be able to react in the event that something doesn't go as planned.

To gather, then, the information that you will need to set the contextual framework for design, you can do the following:

- Understand the business high-level goals.
- Understand the technology goals.
- Draw implied goals from existing documentation.

- Capture (or define) risk tolerances.
- Account for compliance requirements.

Understanding the business high-level goals

"There are two meanings – or perhaps two "dimensions" – to the concept of risk. One is threat versus opportunity. Remember that security is a business enabler: this means enabling the organization by mitigating potential unwanted impacts (the way we often see risk described), but it also means enabling the organization to seize and avail itself of desirable outcomes. These can be in the form of increased competitive advantage, better ability to fulfill the organizational mission, or better positioning in the marketplace – the specifics depend on the organization's context. Remember that the whole point is business use of technology for positive added value: that means minimizing the likelihood and impact of negative outcomes, but it also means maximizing the likelihood and impact of positive outcomes in equal measure."

– John Sherwood, chief architect, thought leader, and co-founder of The SABSA Institute

The very first thing you'll want to do is make sure you understand the high-level goals of the business. What does the business do? What is important to them? What is the mission of the organization? What is the strategy of the organization in pursuit of achieving its ends? The better you can understand this, the better and more complete your designs will be and the less time you will need to spend validating that the approaches make sense later on down the road.

While not required, it can be helpful to document these if they are not already documented somewhere else. Many organizations will have at least some (if not most or all) of the high-level business goals documented somewhere. If this documentation exists, read it, use it, and incorporate it.

Understanding the technology goals

The next thing to understand is the technology goals of the organization. A larger organization might have completed something similar to the COBIT 5 goals cascade exercise outlined earlier in this chapter, or they themselves might have architecture professionals chartered with planning the design for the overall technology footprint. To the extent that you can do so, meet with these folks. Explain to them what you are looking to do, and capture and read any documentation that they have available to help you understand their high-level plan.

As a practical matter, it doesn't really matter that much *how* you capture these goals; you might create a spreadsheet, write a document, or use existing documentation that is already available, assuming it is in a format that is concise and useful. Personally, we have found it helpful to document them ourselves in a spreadsheet with references to existing documentation, where it exists.

Regardless of how you do it, though, you want to understand any technology goals that are in the scope of the areas where your designs will operate. This may not be the full enterprise – in a large organization, it almost certainly will not be. Remember, you are not trying to boil the ocean here. You are trying to start small and build a "playbook" that you can refine over time. There is plenty of time built into the architectural process later to validate your designs in case you miss something at this stage.

Drawing implied goals from existing documentation

Read the policy, procedure, guidance, and standards of the organization. Some organizations might have multiple, differing bodies of documentation that vary depending on the region or business unit. Again, you are not trying to boil the ocean here. The goal is just to collect any implied or explicit constraints applicable to the area that you will be designing for (see the section in *Chapter 3*, *Building an Architecture – Scope and Requirements*, on scoping).

Capturing (or defining) risk tolerances

Understand the risk tolerance of the organization. Where a documented position on risk exists, read it. Where one does not, but there is a formalized risk management process (for example, if there is a chief risk officer or risk function), meet with them to get a feel for the risk tolerance. If there is no risk tolerance defined anywhere, write one yourself and socialize it to obtain buy-in.

Accounting for compliance requirements

Understand, and explicitly list, any compliance requirements that may be applicable. If there are governing regulations for a specific type of business (for example, financial services, payments, or healthcare), read them and understand what is required. If there is a compliance office, meet with them to understand what is required and where.

Summary

Throughout this chapter, we've tried to outline the key elements that you will need to account for in any subsequent design work that you perform. Our philosophy is that once you understand why something is important, the specific steps to accomplishing it become intuitive and self-evident. Additionally, since there are any number of permutations in how an organization might be organized and structured, laying it out in this way allows you to adopt methods that will work best in your environment. However, we have attempted to provide a "recipe" that will work in most situations assuming you understand why each step matters and why it is important.

In this chapter, we've looked at the key dimensions that will inform your planning context: goals, existing organizational structures, and risk management/compliance. These are items that, though not all that you will need data-wise to begin planning, nevertheless inform and direct how your design will proceed. These items are important because they establish the borders within which your design makes sense; using the earlier analogy, they represent the "rules of the universe." From there, we've outlined a "quickfire" process that will get you to a systematic understanding of these items. Note that this is not the only process that you can use to derive this information; for organizations that conduct broader architectural efforts in a systematic way, there might be other approaches that your organization prefers you to employ. However, understanding these items is critical as it tees up the next steps in the process.

Looking ahead, in the next chapter, we will progress to the next phase of design: defining the scope and gathering requirements. This will lay out what your design is intended to accomplish, and where it will operate. Each of these is itself imperative to a solid design and getting them right will ensure that your design operates as effectively and efficiently as possible.