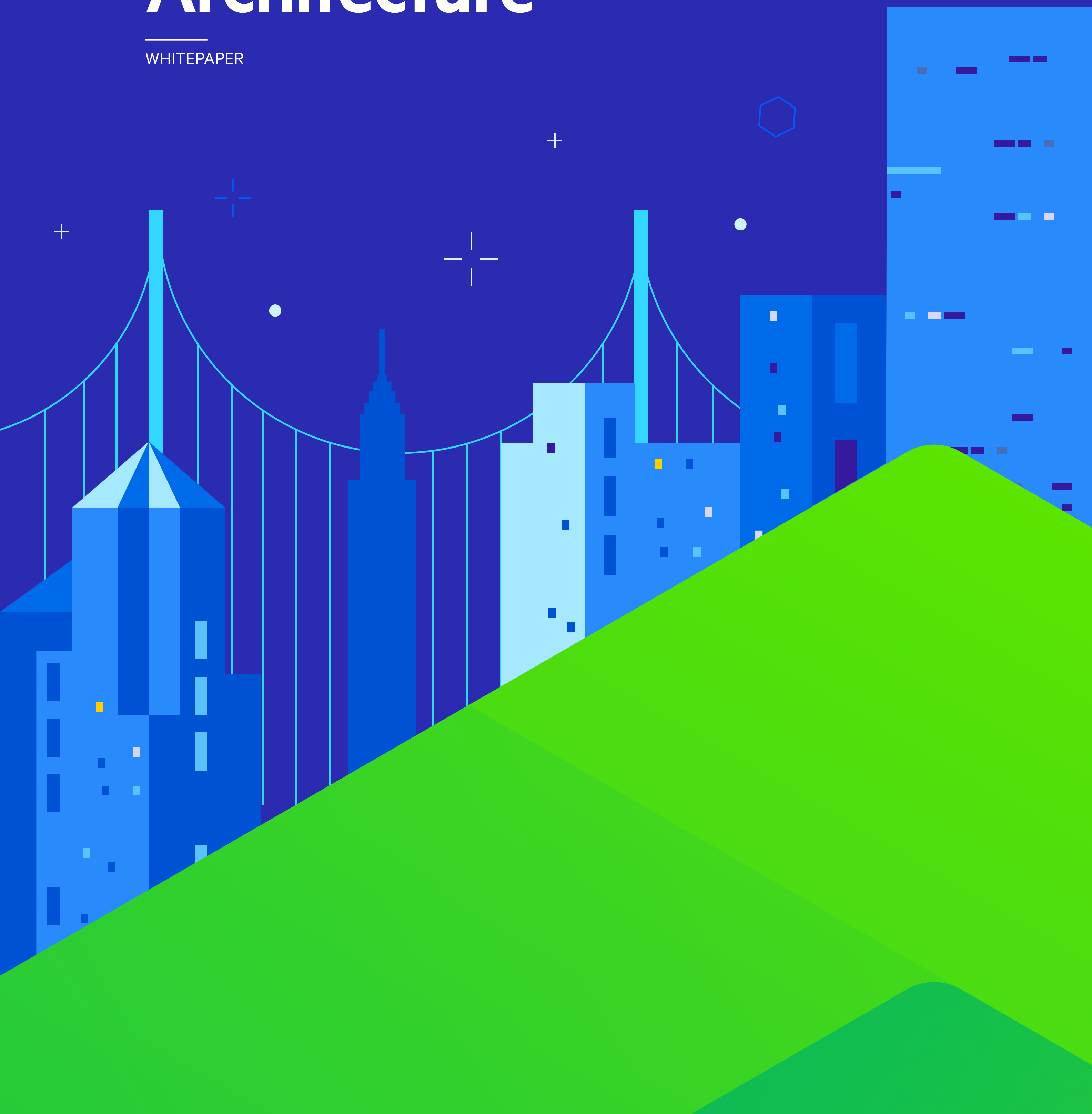
Progress

The Art of Cyberspace Architecture

WHITEPAPER



Abstract

This is a paper about the nature of cyberspace architecture and the principles, capabilities and methods required to develop optimal designs and implementations. It is a subject that does not appear to be defined, taught or applied to the extent that it justifies, given its growing business and technological significance and the increasing business demand for cyberspace architects.

For the purposes of this paper, the term "cyberspace architecture" encompasses frameworks that model systems and processes that operate across cyberspace, whether designed to guide business processes, IT systems or infrastructures, data models, or cybersecurity technologies and controls.

The paper explores and explains a range of principles, techniques and "tricks of the trade" needed to develop high-quality cyberspace architectures, especially at the enterprise level.

Much of the content is original or esoteric, so will be new to many cyberspace architects. I will introduce techniques that are rarely seen in practice, such as methods to enable alternative choices and to create possible past and future worlds, as well as principles for managing complexity, achieving better categorization and joining up architectural components in a seamless and useful way.

I welcome all observations, challenges (however critical) and recommendations.

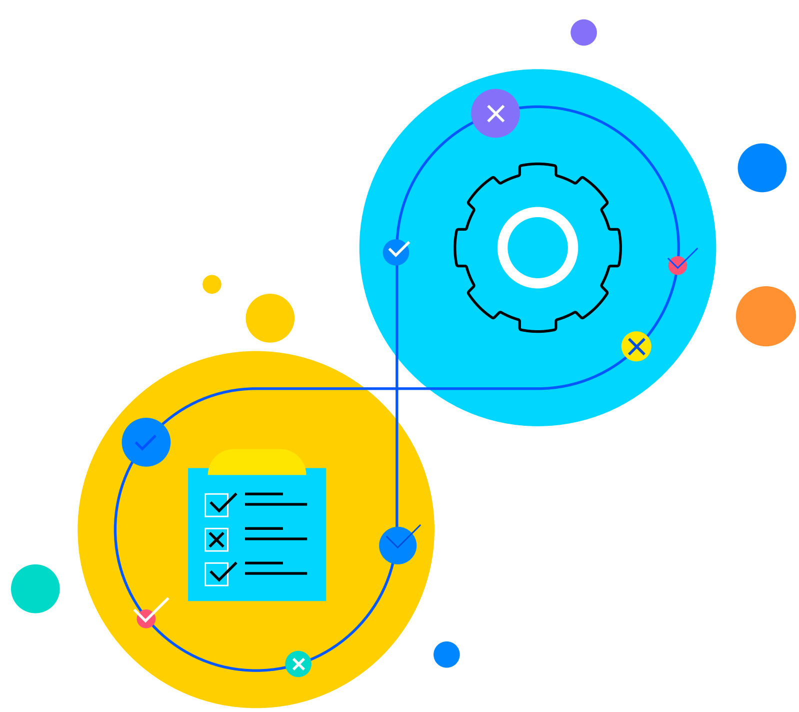
Introduction

The Scope of This Paper

I have called this paper "The Art of Cyberspace Architecture" because it addresses architectures that focus on the type of resources that we might expect to encounter within or between IT systems and the infrastructures that support them. This scope includes business process models, IT enterprise architectures, system architectures, data models and security (or cybersecurity) architectures.

I had considered using the title "Digital Architecture" but this term is used in the "built environment" to address the design of physical buildings and the creation of online spaces.[1] And it is an important design principle of mine to minimize ambiguity and misinterpretation by avoiding terms, words, abbreviations and acronyms that have alternative meanings.

For simplicity, I refer to the general community of architects working in cyberspace as "IT architects" or simply "architects" whether they are enterprise architects, solution architects, security architects or data architects. There are differences of course in their skills and their preferred methods and structures, but such distinctions are not particularly relevant to the content of this paper and will be pointed out as it unfolds.



A Brief Note on Terminology

This paper is concerned with architecture. There are other terms used to describe structures that help solve business problems, and many are used interchangeably. Does it matter which term we choose? Not really. Our main objective is to design useful structures that solve complex business problems. The terminology is less important. But it is worth noting the nuances associated with these terms. So, here is my take.

A framework is a simple conceptual structure to help address complex problems by breaking them down into components and illustrating their relationships.

A model is an abstraction of reality, a simplified representation of a real life that is used to highlight certain important characteristics for a particular purpose, ignoring other things judged to be less relevant.

An architecture is a framework or model that is used as a basis for constructing something, whether it be a building, a system, an infrastructure or anything else.

These structures are a means to an end. Frameworks break problems down into manageable chunks. Models help us to focus on the most important features of a problem or solution. And architectures provide the foundation for taking us to the next level of detail in solving a problem. Architectures are an essential step in building the sophisticated solutions that we need to address today's complex business problems.

The Aim of This Paper

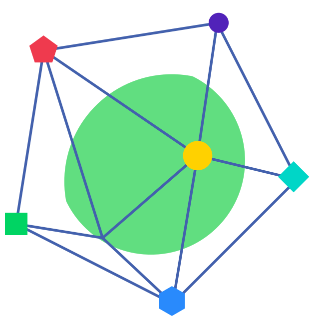
The idea for this paper arose from a series of talks I gave in the first half of 2025 on the subject of Zero Trust architecture. I expected pushback on my radical views from professional architects, but was pleasantly surprised to find encouragement for new concepts.[2] I came away determined to set out my views and ideas in more detail for the stimulation of other IT cyberspace architects.

The paper sets out a range of powerful but little-known design considerations that enable large complex structures to be better designed, managed and used. It focuses on some esoteric "tricks of the trade" to create high-quality structures, rather than the more widely taught skills required to manage the development of an architecture. It also draws on little-used principles from fields as varied as philosophy (metaphysics), grammar and mathematics, as well as the little-known but powerful practice of high-quality data modeling.[3]

These ideas, correctly applied, can deliver long-lasting, fully-flexible designs with minimal technical debt that deliver long-term cost-savings and benefits.[4] But be warned! Quality design takes longer, and therefore costs more in the short term. And upfront costs do not fit comfortably in a business world dominated by agile development methods and minimum viable products.[5] But, for long-lasting, grand schemes and critical designs, it pays to adopt a "slow and sure" approach rather than a "fast and loose" one.

Many ideas and methods presented in this paper are original thoughts or new applications of existing principles. Some are obscure but proven techniques that are rarely taught or practiced. Some are the ideas of brilliant academics that have not been widely adopted by the mainstream community.[6] Some might be difficult to grasp and challenging to implement. But they are not speculative; I have personally applied all of these ideas successfully in practice.

My hope is that the content of this paper will stimulate new thinking for architects and designers, not just in the IT field, but perhaps in other fields concerned with the design of large complex structures.



Is Architecture Art or Science?

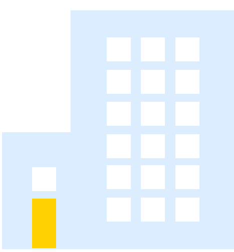
According to Encyclopaedia Britannica, architecture is the art and technique of designing and building, as distinguished from the skills associated with construction. This definition implies that there are different skills and techniques required of an architect than needed for builders. It also implies that architecture is as much an art as a science.

When one considers magnificent buildings such as Gothic cathedrals and Art Deco skyscrapers, it's quite clear that architecture has a strong artistic component. It's also obvious that the knowledge and skills of their designers must have been quite different from those of the stone masons and construction workers who built them. But is this also the case for cyberspace architectures? Is there an artistic component? And are the skills of an IT architect different from those of the programmers who construct the building blocks?

In my view the answer is "yes" to both questions. The development of a successful cyberspace architecture (though there are few examples) demands good skills in disciplines such as aesthetics, ergonomics and communications, as well as a good grounding in computer science and engineering. Artistic skills are less important in software construction, though elegance in algorithmic design is clearly desirable.

Encyclopaedia Britannica mentions design as a component of architecture, which prompts the question of whether there is a difference between architecture and design? Are they distinct in some way? On this issue, the answer is less clear-cut. There are obvious differences, but they are largely concerned with vision and scale. Architecture demands a holistic perspective to conceive and portray the overall structure of a process, system or infrastructure. Design focuses primarily on the form of the individual components and how they fit together. This distinction is important, so we will return to it later in the paper when we consider the importance of wholeness and the perils of fragmentation.

Scale is also a challenge for architects as further management skills are required to manage the delivery of a large, complex architecture in comparison to those needed to shape the design and interfaces of its individual components. Enterprise architects require a good political antennae and strong leadership skills to help navigate the maze of corporate and office politics.

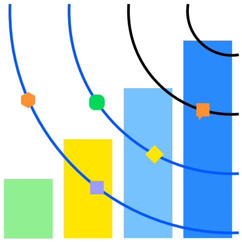


Why We Need Architectures

There are good reasons for developing architectures. All grand designs demand a comprehensive vision and a means of constructing and communicating it. Large, complex systems also need to be developed (or replaced) in manageable parts. Modularity enables the reuse of existing components and builds agility into systems by enabling changes to be easily and quickly accommodated. Many monolithic computer programs have been scrapped for the sole reason that they were unable to accommodate change.

Architectures have many benefits but the most important one is their ability to manage complexity. A good framework helps us to explore, decide and communicate how individual building blocks can best be arranged to form a pleasing, utilitarian and accomplishable whole. Architectures are essential for constructions with numerous interacting components, such as business processes, organizational designs, IT infrastructures, data models, code structures and cybersecurity controls. They also enable the construction work to be divided up, and they help simplify interfaces between components. Their unique perspective also enables business activities to be streamlined. And in addition, they promote standardization, reducing diversity and duplication of effort.

All these things help reduce operational and support costs. The challenge is the difficulty in finding the hard evidence required to support a financial business case for developing an enterprise architecture. Such an investment will usually therefore be a leap of faith, justified by a desire to adopt industry best practices, or the need to support an important strategic business initiative.



Types of Cyberspace Architecture

There are many flavors of cyberspace architecture. Business process (or activity) models are necessary for identifying, negotiating and defining efficient operations and services, especially those that traverse multiple business functions and locations.[7] A business architecture is, in fact, a fundamental requirement for business operations and growth, as it provides a template for day-to-day operations, as well as a vehicle for controlled expansion into new regions and markets, enabling effective, efficient and consistent deployment of resources.[8]

IT architectures come in several forms. There are “solutions” architectures that address specific application systems. There are also technical architectures that focus on technology choices at various layers in the infrastructure. Then there are enterprise architectures that present an ambitious grand design of the key IT resources across the organization. And in addition, there are activity-based models that guide IT processes, as well as lifecycle methodologies for systems development and operations, or increasingly DevOps and DevSecOps.

Data models are essential for structuring data in databases and IT systems because of the need to design and document how the data is related and structured, as well as the necessity to navigate and maintain complex sets of data.[9] Such models are generally developed in an iterative way, progressing from a conceptual to a logical and then a physical perspective. They can be developed in a top-down, bottom-up or random way, though this paper strongly advocates a top-down, ontological approach.

Enterprise data models are necessary to establish a single, consistent view of data across application systems, enabling simpler and more reliable systems integration, data sharing and data analytics, while at the same time preventing unnecessary data duplication, which increases data storage costs and can lead to inconsistencies and errors in retrieving data.

Security (or cybersecurity) architectures are necessary for defining and communicating security policy, technology and controls. “Controls” are surprisingly wide in their scope. From an ontological perspective they are essentially a requirement for one or more objects to possess one or more attributes. These requirements can be precisely expressed in first-order predicate calculus, which is the level at which a relational database operates, so there is a potential for reliable mapping of controls to stored objects.[10]

The unlimited scope of controls means that to achieve their full potential, security architectures must be capable of accommodating all types of objects and attributes within and outside an organization, including people, organizations, physical objects, activities, associations and other things.

Diagrams vs. Words

The "aspirin" of architecture is the two-dimensional diagram.[11] It supports the modeling process from beginning to end. And it can immediately deliver a view of the scope, structure, components and connections of a system, process or infrastructure. A diagram can also serve as a "strawman"—an initial, often knowingly flawed proposal designed to provoke criticism and brainstorming.

"A picture is worth a thousand words" is an ancient pronouncement that has been plagiarized by everyone from Leonardo da Vinci to Napoleon Bonaparte. There is merit in this claim. A picture simultaneously communicates the whole and the parts. It can also reveal relationships between components, though with limited space to explain their nature.

One can of course counter that "a thousand and one words are worth more than a picture." And then there is the insightful claim of Edsger Dijkstra (the famous Dutch computer scientist), who remarked, "A picture may be worth a thousand words (but) a formula is worth a thousand pictures." We should never forget or underestimate the power of algorithms to model and manage complex situations.

Diagrams and words are two sides of the same coin. Both are a primary means of communicating information, though they differ significantly in their speed, precision and indelibility. Each has their own relative strengths and weaknesses. Diagrams are excellent for simplifying the presentation of complex structures and relationships. They stimulate the creative and emotional side of the brain. They are faster to process, more memorable and can help convey fresh insights and present a whole that is greater than the sum of its parts. They also help overcome language barriers and they are better suited to collaborative development.

In contrast, words are superior for conveying detailed, precise descriptions and explanations. Words are processed by the left hemisphere of the brain, which operates more like a dispassionate text processor, controlling writing, arithmetic and comprehension. Pages of text lack the simplicity, impact and immediacy of diagrams, but they are a necessary accompaniment. Diagrams have insufficient room to include enough words to fully describe and explain the objects they illustrate. And words have power. A simple phrase can inspire a revolution. And a single change to a word can transform its meaning and influence.

Real-time diagrams such as dashboards are powerful monitoring aids as they convey views of the status of multiple items at the same time. This is the great advantage of data visualization tools. Graphical representations are valuable where speed of observation and analysis is important, such as within a security operations center (SOC) to help identify cyberattacks. They can also enable executive boards to rapidly assess the real-time, operational status of critical business services, though this is rarely seen in practice.[12]

A purely diagrammatic architecture or a solely text-based description will fail to convey the full impact, intricacies and influences of a complex architecture. Both are essential.

In Praise of Simplicity

Many enterprise architectures are similar in presentation, often inspired by the Zachman framework developed by John Zachman of IBM. This framework is a six-by-six matrix of individual models. Each column of the matrix is based on an element of the "5 Ws and one H": Who, What, Where, When, Why and How, an analysis tool first alluded to by Rudyard Kipling. And each of the rows of the matrix represents a particular stakeholder: Planner, Owner, Designer, Builder, Subcontractor and Functioning Enterprise.

The Rudyard Kipling checklist is a useful brainstorming tool for performing a comprehensive analysis of a complex problem. But it has an unfortunate tendency to proliferate rather than prioritize content. And the most important attributes of a usable enterprise architecture are focus, navigability and simplicity of presentation. Unfortunately, however, the world of enterprise architectures is riddled with abstract ideas and shapes, excessive ambition, unsupported promises and a tendency toward overengineering.

Architectures are extremely useful and are essential for the success of business operations and strategic initiatives. They are especially valuable when applied at the enterprise level as they help control the inevitable complexity, and they assist in standardization, growth and efficient business operations. But that is only if they are well designed by architects who understand the underlying principles and techniques that contribute to good design and communication.

What Should a Good Architecture Look Like?

"Give me a framework and I'm free" is a quote I recall hearing in the 1970s, a time when frameworks were relatively thin on the ground. I can certainly understand how a good framework can help a novice go about a complex task. I can also appreciate how a badly structured one might distract and confuse a practitioner. Personally, I prefer to paraphrase the quote as: "Give me a good framework and I'm free. Give me a bad one and I'm lost."

An architecture is no more than a structure supported by specifications. There are no rules on how they should be assembled. One may choose to follow the detailed and suspiciously complete frameworks promoted by John Zachmann and other authorities. Many organizations have started down that track yet failed to complete the task successfully. And there is further scope for even more complexity, as some architects have begun to adopt cubic shapes, introducing an extra dimension to an already crowded set of representations.

A good architecture should focus on bringing out the most important aspects of whatever is being represented, at the expense, if necessary, of the least critical ones. That's the whole point of models. They are an abstraction of reality to enable the user to focus on the most important elements while filtering out the less relevant ones.

For many years I have stated what might be considered by many IT architects to be a sacrilegious pronouncement:

"A good, forward-looking architecture should be ragged around the edges, full of holes and exist largely in the minds of people."

Such a declaration might come as a shock to many people, at least on first hearing. But as Shakespeare's Polonius put it: "Though this be madness, yet there is method in it."[13] Think carefully about these words. A real-world architecture has shifting boundaries as organizations and infrastructures evolve and externalize their services. It can never be complete, as we rarely have all of the answers to new paradigms and emerging challenges. And if it is to be actually used by people, it needs to be in their minds when they make decisions, rather than sitting unread on a shelf in a cupboard or buried on a server in a lengthy folder of organizational policies and standards.

So be suspicious of architectures that appear complete, flawless and symmetrical. The business world is not like that. It is untidy and evolves with changing priorities and focus, sometimes in leaps and bounds. And it cannot realistically be shoehorned into a neat, two-dimensional view of some desired future state. We need to acknowledge gaps and emerging issues, and to be able to accommodate changes to its boundaries, dimensions and content. A good architecture should also include next-best alternatives and appropriate compensating controls, as not every aspiration can be justified, afforded or implemented in practice.[14] We need realistic, predefined and approved fallback recommendations.

It is worth studying the definitive design principles developed by Dieter Rams of Braun. He asserted that good design is innovative; makes a product useful and understandable; is aesthetic, unobtrusive, honest, long-lasting, thorough down to the last detail, environmentally friendly; and has "as little design as possible." These principles are highly applicable to architecture design.

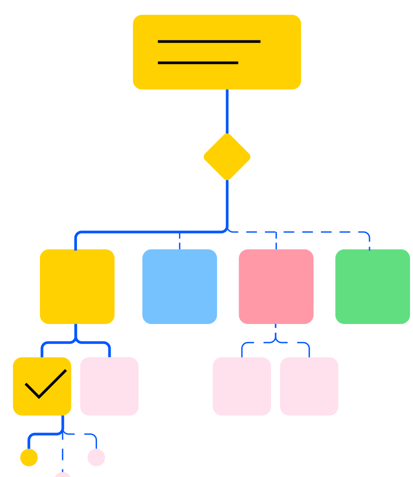
We need to see more original thinking and innovation for the design and presentation of cyberspace architectures. Despite past failures, little seems to have changed over the last half century. In addition to incorporating alternative choices, we also need to address missing dimensions (such as time) in our models, and to adopt better methodologies to improve our categorizations of objects and attributes. Managing complexity is also an important challenge. We can strive to be simple, but as we shall see, complex systems can only be managed by control mechanisms that are equally complex. The remainder of this paper addresses these requirements and how they might be achieved.

Incorporating Alternative Choices

Many years ago, when I was a director in a large enterprise, I commissioned the development of a cybersecurity architecture for a deperimeterized infrastructure. I knew this would be a complex task as it needed to take account of a large estate of legacy systems and infrastructures, many with inadequate security and limited room for remediation. In addition, we were planning the implementation of ambitious new enterprise systems and cutting-edge e-business systems.

The network infrastructure and office systems had many unused security features, but supporting products were of different generations and not sufficiently compatible to enable centralized security management. The new architecture had to address all of these challenging constraints by promoting and enabling a target infrastructure, while recognizing the unavoidable slower migration paths necessitated by existing legacy systems, components and infrastructure.

My researchers built an architecture expressed in pseudocode with many IF ... THEN ... ELSE statements to explain what to do if a recommended architectural component could or could not be implemented, whether for technical, financial or operational reasons. This code was also used to form the basis of a simple piece of software to guide users in determining implementation choices. Interestingly, this approach had a mixed acceptance, embraced by some and disliked by others. But as they say, "that was then, and this is now." Employees today are now much better educated in programming concepts.



Lifecycles and Maintenance

Time is an important consideration in designing architectures. There is in fact more than one consequence to this assertion, which is further discussed in a following section.

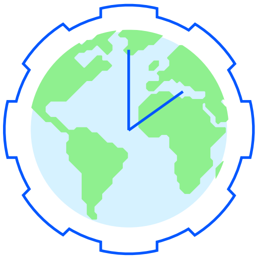
Ease of maintenance over time is a fundamental requirement that runs through all architectures. Lifecycle asset managers will recognize the folly of cutting corners in the short term at longer term expense, though other executives might counter that, as Keynes put it, “In the long run we’re all dead.” Nevertheless, architectures need to be maintainable and the long-term cost of this should not be prohibitive.

I have encountered architectures and manuals that mixed together content with very different update lifecycles, which should never be done as it results in content that is unmaintainable. Some standards, policies and principles rarely change and can be good for as long as five to 10 years. Other items require more frequent revision, perhaps annually, monthly or even daily. It’s important to be conscious of the revision cycles of different sets of content, to be able to isolate them for maintenance purposes. Frequently changing technical or organizational details, for example, should not be expressed within longer-lasting policy statements.

Artifacts should be structured into temporal levels for maintenance purposes, though their content will need to be linked for navigation and presentation purposes. At the highest level we might plan for the overall vision, scope and objectives of the architecture to have a lifetime of 10 years without any major modifications. Such material should be forward-looking and visionary.

At the next level we can position corporate policies and standards that do not require any major revision for five years, which is the length of the cycle for revising international standards. At the third level we might position technical standards and guidance that need to be revised for each major release of software, perhaps every one to two years. At the fourth level, we should locate technical advisory information, for example on recently discovered vulnerabilities, which needs to be addressed in real time.

By separating these layers of documentation, the content will become much easier to maintain.



Catering for Alternative Worlds

If architecture begins with two-dimensional sketches, then it should end in more. The world we are modeling has at least four dimensions, and possibly more. We all recognize time as the fourth dimension, though theories in quantum mechanics suggest that the universe might be composed of 10, 11 or more dimensions.[15] These additional dimensions are not quite the same as physical dimensions of course, but if they are useful to cyberspace architecture, then we should try to understand and apply them.

Subatomic matter clearly has little relevance to cyberspace architecture, but time is an important dimension. Modeling the future is useful, for example for various kinds of planning purpose. And analyzing or postulating past events is essential in cybersecurity investigations. So, let's consider time as a useful dimension that we should attempt to build into our architectures.

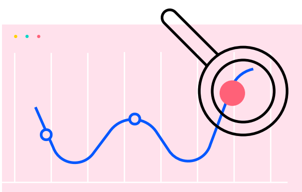
We normally think of time as a single dimension, but there are a few nonconformist theories that claim that there are more. The revolutionary aircraft designer Robert Bartini believed that time had dimensions of length, height and width, where height is the speed of time and width is the number of possible (parallel) worlds. To many, this might seem bizarre. Yet the concept of parallel worlds is surprisingly useful for architectural purposes. And if it is useful, we should pay attention to it.

The notion of possible worlds goes back more than 500 years to Leibniz but has mostly evolved over the last century. Like many things in mathematics such as imaginary numbers, a possible world doesn't have to be real to be useful. If we design it into a database or architecture, however, it is most certainly real within that context.[16] In fact, high-quality data modeling, of the type expressed in the international standard ISO 15926-2 (a conceptual data model for technical information about process plants) actually includes the ability to model possible worlds. And most people would agree that you can't get more real than a process plant.[17]

High-quality data modeling implements possible worlds through an ontological approach called four-dimensionalism.[18] This defines sets, not in the usual way by defining fixed criteria for membership (as is generally done in mathematics) but by asserting that they are defined by their members, past present and future.[19] This enables us to model objects as extending through time from their origins in the past, through the present and into the future, including alternative past or future states where paths may diverge or cross at any time. This may sound challenging, but it is surprisingly easy to implement in a data model by the simple insertion of a supertype (parent entity) above the object that represents all particular states, real or theoretical, of the object in question.[20]

This type of set definition makes perfect sense for everyday groups of objects such as a "network" or "organization," which generally do not have defined membership criteria (as anyone can be invited to connect or join). And by allowing us to model alternative past and future worlds, we can better support investigations into past events and alternative future plans. Further, it also enables all data to be retained regardless of updates and modifications, as all changes to an object's attributes create new states, while retaining data relating to their previous states.

So now we have a richer, more versatile means of modeling objects and their attributes which can be implemented in a relational database, accommodating possible past and future worlds and enabling alternative views of the architecture to be presented to its users.



Managing Complexity

The primary reason for an architecture of any kind is to help manage the design, development and operation of a complex process, system or infrastructure. The business world is becoming increasingly complex with the growing use, networking and externalization of technology. And architecture is the tool of choice to define, present and navigate complexity.

We can start with simple, two-dimensional diagrams, but when it comes to implementing real processes, systems or infrastructure, we need to dive deep into the lowest levels of detail. And this is where architectures can fail, by using oversimplistic or nonintuitive structures to navigate fine-grained content. Or perhaps by incorporating unnecessary, distracting detail to pad out incomplete matrixes.

There are many principles and techniques to make complex models easier to navigate. Simplicity is a fine aspiration. But an equivalent degree of complexity is necessary to model complex systems and structures. What needs to be avoided, however, is unnecessary complexity. The art of managing complexity is a broad, deep subject that is rarely taught but I will attempt to provide a summary of how to go about this necessary task.

Computers and networks amplify complexity. This fact has been known since the 1950s when cybernetics, artificial intelligence and systems theory were fashionable sciences. There is even a formal mathematical law (Ashby's Law of Requisite Variety) that states that if we wish to control a system or situation, then we will need to have the same number of states (termed "variety") in our controlling mechanisms as there exist in the system we are trying to control.[21] Otherwise the system will be unstable and out of control.

The consequence of Ashby's law is that a simple control mechanism cannot control a complex situation. So how is it that we manage to control them in everyday practice? Two ways. Firstly, there are techniques that enable us to reduce the number of allowable states in the system we wish to control. And secondly, we can exploit technology to amplify the number of states in our control mechanisms. These techniques are not usually taught to systems designers, but many managers will over time discover them, at least enough to verify that systems are not out of control.

The number of states in a system can be substantially reduced by techniques such as standardization, centralization and categorization, or by applying limits or filters, and resisting unnecessary changes. We can also scale up the number of states in our controlling mechanisms by using technology, which is a recognized variety amplifier. Computers are excellent at monitoring the individual states of systems, infrastructures and environments. And networks can enable a single endpoint to harness the power of other devices connected to the network.

The management and use of a complex architecture can be considerably enhanced by basing the content on a comprehensive high-quality data model, preferably one that is ontologically sound and has a structure that reflects the underlying nature of objects and the natural relationships between them. These techniques are the foundation for efficient categorization, which is the key to building architectures that are manageable, flexible, extendible and easy to navigate.

The Elusive Art of Categorization

If smart categorization is the key to building excellent architectures, then how do we achieve it? Well, the first point to note is that it is not sufficient to rely on common sense or to copy and go with the crowd, because both you and the crowd will almost certainly be wrong.

Good categorization is not taught and is not instinctively obvious. It can only be achieved by an understanding of metaphysics and grammar, and by standing on the shoulders of the small number of professionals who devoted their careers to mastering the art of high-quality data modeling. “Experto credite,” as they say in classical circles: “Trust an expert.” And preferably one who knows how to do it properly.

Why is good categorization so hard to achieve in practice? There are several reasons for this, such as our tendency to structure taxonomies based on the words that we use to describe things, rather than the nature of the objects themselves. Words are dangerous, as virtually all of them have alternative meanings.[22] And this is not helped by the fashion for creating abbreviations and acronyms that usually have several alternative interpretations.

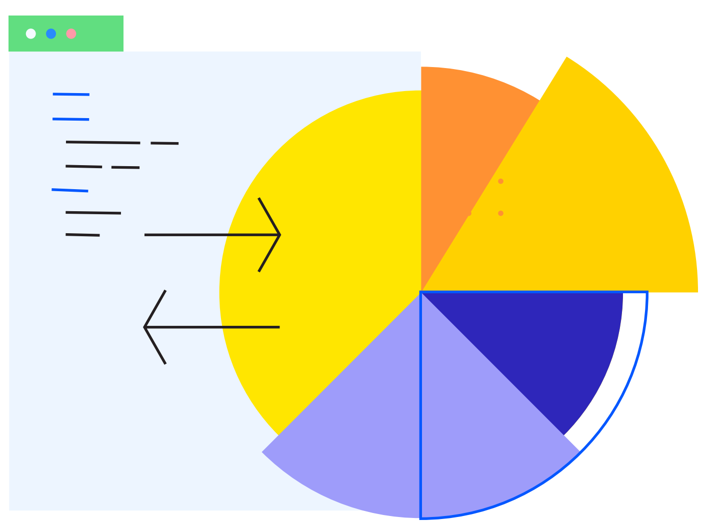
Ignorance, impatience and oversimplification are the enemies of good categorization. The components of an enterprise architecture are an interrelated combination of heterogeneous items: processes, organizations, physical objects, software systems, security controls and information, as well as their individual attributes. To make this kaleidoscope of objects and words fit together in an organized and navigable way demands insight, experience and know-how. In particular, it demands an appreciation of metaphysics and high-quality data modeling.

High-quality data modeling promotes an ontological approach, which is that structures should be designed from the top downward and take account of the nature of the items being categorized. Objects need to be grouped according to what they actually are (such as a person, a physical object, an activity or an association) which allows the supertypes and subtypes (parent and child relationships) of entities to share common attributes and relationships. The structure of the data will also be easier to navigate, as it will be more obvious where to find things and these items are less likely to fall into more than one category. To paraphrase an idiom, we should aim for a locatable place for everything and everything in its correct place.

Categorization starts at the very top with a clear division between spatiotemporal objects (things with beginnings and endings such as real-world people and physical objects) and abstract objects, (infinite things such as colors, properties and other types of attributes). Spatiotemporal objects are generally referred to as “instances.” And collections of abstract objects are generally referred to as "reference data." It is important not to mix instances of things and reference data in the same category because the former things come and go, and the latter last forever. Consider "information" for example. It is both a physical object (a paper or electronic document) and a pattern of symbols. The physical object can be easily replicated or destroyed, but the latter has always existed and is indestructible.[23]

It is important also to differentiate between objects that share similar labels but which are quite different in nature. We may have an entry called "intrusion detection" for example, but this could be a physical device, a piece of software or an activity. And these are different types of instances, so they should not be grouped together. They may be linked in some way but they should not occupy the same path within a structure. Such labels should be qualified, for example by calling them "intrusion detection appliance" or "intrusion detection activity."

The underpinning structure of an architecture need not be the same as the views presented to its users. In fact, they will be quite different as they serve different purposes. The backbone of the architecture should be a high-quality data model designed to be complete, extendible and ontologically sound. The presentation of the architecture should be designed to be selective, relevant and understandable to its users, expressed in their language and presented in a way that reflects the politics of the organization.



Choosing the Right Words

If the first law of categorization is to aim to get the structure right, then the second should be to choose the best possible terminology for both category headings and their contents. This is just a simple matter of selecting words that are specific and unambiguous. But simple isn't always easy. Virtually all words have alternative meanings, and there are no clear-cut boundaries between word classes such as verbs, nouns and adjectives. Grammar is far from precise, perhaps reflecting its human-driven nature.

The first step in choosing terminology is to avoid words that have a large number of alternative meanings. Some have many more than we might expect. The current record in the English language for example, is held by the word "run" which has 645 different use cases for the verb form alone.[24]

We should also be careful to avoid grouping different types of objects just because they share a common attribute. For example, the word "site" might represent a website, a building, a campus, a region, a strip of land, a city or a space station. These are all geographic locations but have nothing else in common and should not be regarded as things of the same type. A building might be a location, but a location is certainly not a type of building.

Further misinterpretation can be avoided if we avoid abbreviations and acronyms, as they are rarely unique. Is a "CSO" a Chief Scientific Officer or a Chief Security Officer? And does "IP" stand for Internet Protocol or intellectual Property? We need to be as specific as possible, for example by adopting qualifying adjectives or nouns as we did for "intrusion detection."[25] And in addition, some words (e.g., "document") can be either a noun or a verb, and we really do not want users to mistake a physical object for an activity.

Such are the limitations of the English (and every other) language. These quirks make it hard to create a perfect categorization of objects though we can go a long way toward this goal by being as specific as possible in our category names, by choosing the least ambiguous words rather than the fashionable ones of the day, by avoiding abbreviations and acronyms, and by focusing on the underlying nature of the objects themselves, rather than selecting the first word that springs to mind.

Achieving Wholeness

I once asked a Head of Enterprise Architecture what he did. He replied, "We create artifacts." I had little idea what he meant, as an artifact is any object made by a human being. This term has caught on with enterprise architects, but it introduces several presentation problems. Firstly, it is obscure (many people simply don't know what it means). Secondly, it is not specific (it can be absolutely anything). Thirdly, it does not reflect the business rational for the work. And fourthly, it encourages a fragmented approach rather than a holistic one. Architecture should be seen as a single, joined-up whole rather a collection of individual lists and diagrams.

I have written elsewhere about the perils of fragmentation in managing complexity. It is a major challenge for architecture design and maintenance. We need to break large, complex systems down into smaller, more digestible pieces. But we must never lose sight of the bigger picture. Excessive focus on the individual components of an architecture will result in a disjointed, distracting and disruptive approach to its development. We should always strive to verify that the components fit precisely together and that the whole is more than the sum of its parts.

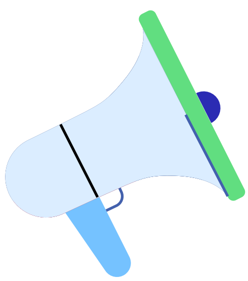
The brilliant American theoretic physicist David Bohm was concerned that most of society, business and academia had a damaging tendency to focus on individual fragments (whether countries, companies, religions, academic subjects or physical objects) rather than verifying that they work together as a whole. He advocated "wholeness" as an important, overriding objective.[26] Bohm's philosophy was shaped by his understanding of how the world appears at the subatomic level where everything is a single, endless sea of interacting particles rather than a collection of distinct objects.

Unfortunately, fragmentation always seems to grow, a bit like entropy does in the second law of thermodynamics.[27] But fortunately for architects there is a practical approach that enables better categorization of objects and also enables them to be seamlessly and usefully joined up. Because Bohm's ideas were influenced by those of Alfred North Whitehead, who believed that process was the only reality and that physical objects only exist when participating in activities.[28]

And whether or not you believe this to be true, it is certainly the case that all of the components of an architecture must have been created by, are involved in, or have been removed by some forms of activity. If we consider business management, IT management or cybersecurity management, it is clear that they are all driven by activities.[29] Chief Information Security Officers talk of management systems. But this misses the very important point that management only works when activities are in motion. Cybersecurity is only effective when its participants are operational. A switched-off or unmonitored security appliance delivers no security.[30] Cybersecurity is a collection of activities, not a system that can be switched off.

Many things in life are activities. Risks, for example, are possible incidents in the future, and incidents are simply a collection of causal activities, effects and responses, such as an attack, followed by a damaging event, followed by series of corrective actions. How do we model an activity? From an ontological and data modeling perspective, an activity is simply a spatiotemporal extent composed of a set of physical objects participating in specific roles that create, reference, modify or destroy other objects and activities.[31]

Focusing on activities is a powerful means of joining up the individual components of an architecture. If we start with processes and activities and map out the various physical objects, roles, things referenced and changes brought about, then we are on our way to a near-perfect, or at least a usable and maintainable, architecture.



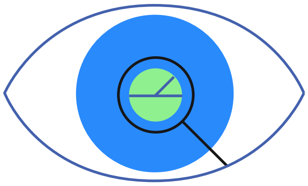
Presentation and Politics

I have mentioned the usefulness of separating the underlying structure of an architecture from its presentation to users. There are three reasons for this. Firstly, we may wish to hide technical or philosophical jargon from users. And secondly, creating tailored views for specific audiences is helpful to them. And thirdly, it allows the terminology of the architecture to not be constrained by the jargon, phraseology and acronyms favored by particular users.

I have advocated a top-down approach to defining the structure of the architecture. This should not start from simple sketches designed for users or management. Because, firstly, it will encourage the use of less precise terminology, which is likely to be cascaded down into the more detailed reaches of the architecture. And secondly, it will encourage excessive brevity to allow the content of each box on a chart to be squeezed into the limited space available.

Politics is also an important consideration in presenting an architecture to executives. In particular, the order and relative size of the boxes on a chart or display can imply the relative importance of each process, system or organizational unit. The core structure of the architecture should be independent of politics, but its presentation will need to be designed with politics clearly in mind.

I have pointed out the dangers of symmetrical matrixes and cubes, but one shape that is to be avoided at all costs although it remains surprisingly fashionable is the circle, which results in text being displayed at different angles and in different size fonts as designers struggle to shoehorn words into the increasingly tiny slots toward its center. Such a shape is also far more difficult to extend as the insertion of a new entry generally requires a complete reconstruction of the shape.[32]



Selecting Architectural Components

One of the keys to good architectural component selection is standardization. Standardization is an essential objective to minimise unnecessary complexity. We should always aim to minimize the number of processes, technologies and controls deployed in an organization as diversity not only complicates architectures, but it also increases development, support and maintenance costs.

Standards, however, can be a double-edged sword. Carefully selected, they are a powerful cost-cutter and variety reducer. Badly chosen, they are an expensive and restrictive burden. Good standards, whether for activities, technologies or controls, need to be forward-looking, widely supported, compliant with regulatory requirements and an enabler for externalization.

Forward-looking products and standards are those that are maintainable, scalable, adaptable and extendible. We should avoid unsupported ones and those that are likely "cash cows" with little or no incentive for a vendor to add new features.[33] We should also steer clear of any components that are likely to introduce significant technical debt (the hidden cost of future change). And products and standards should also be capable of adapting to emerging trends, both to meet future business needs and to empower their own survival.

Another important criterion is the level of global and multisector support not only from the vendor or authority but also by third parties, as this is a reliable indicator that they will survive in a competitive marketplace. It also facilitates expansion into new regions or markets. And it means that our chosen product or standard has a surplus, rather than a shortage, of available support. Speed of response to incidents is also a very important consideration. If you have a ransomware attack on a Friday night or Saturday morning, will the vendor respond quickly? Will engineers need to fly in from a different country?

Compliance with regulatory standards is increasingly becoming a mandatory requirement as both customers and authorities become more demanding. Persuading an existing supplier to demonstrate compliance with a specified standard is always very difficult and expensive. Demonstrated compliance with all relevant standards by accredited certification or a regular independent assurance from a trustworthy party should be a priority selection criterion.

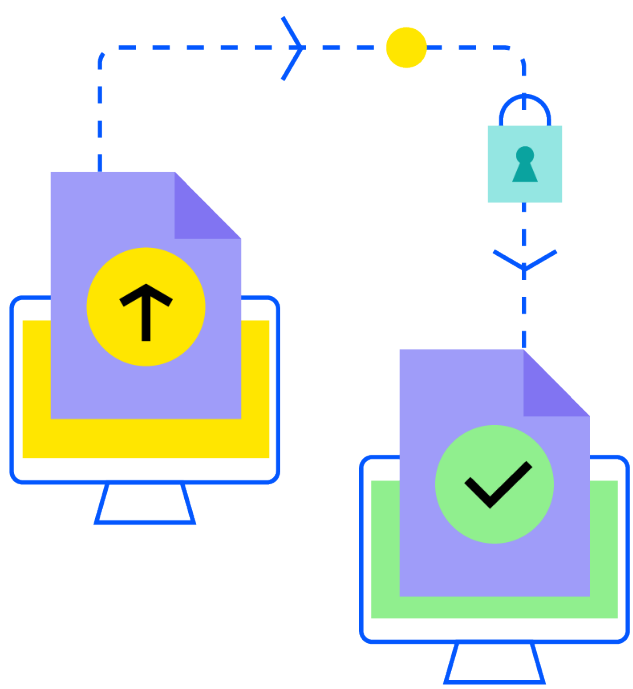
Cybersecurity is a requirement whose scope goes beyond regulatory compliance demands. The emerging and widely accepted doctrine is to adopt a "Zero Trust" architecture based on not assuming trust for users or devices inside a network. What this means in practice is that business applications and their supporting systems will need to migrate from reliance on a strong corporate security perimeter to operating within an insecure, potentially compromised network. These demands require solutions based on strong authentication, end-to-end encryption and threat monitoring.

Support for externalization is also an increasingly important criterion for product or standard selection. We need to find components that can support business processes and systems both within the organization and externally across corporate boundaries. When agreeing on a product or standard for data sharing between organizations, it is clearly better to persuade your partners to adopt a system of your choice rather than to have to adapt to those of your partners. That means choosing a product that is both widely used and easy to introduce.

Finally, you should also check contracts carefully for hidden charges for changes in the use of a product or service. There is a long history of software vendors exploiting customers through aggressive practices that charge unreasonably high fees for small changes, such as a change of location of an extension of a product or service to a third party. Choose a reputable supplier and take advice from an experienced software licensing expert.

Managed File Transfer

A good example of the type of product or service to consider is managed file transfer (MFT). This technology promotes standardization of data exchange processes and technology, facilitating secure externalization. Progress® MOVEit® software and the Progress® Automate MFT platform, for example, support secure, automated, scalable, extendible and compliance-ready workflows. Both products are forward-looking designed to be easy to implement, globally supported and flexible, with compatibility across major platforms and protocols. And it is forward-looking: continuously evolving, introducing new features and leveraging emerging technologies. Both MOVEit software and the Automate MFT platform align with the Zero Trust paradigm by focusing on security at the application and data levels, supporting layered protection strategies beyond perimeter-based approaches.



Summary of Recommendations

In this paper, I have advocated and presented multiple ideas, principles and techniques, many of them new, for designing better cyberspace architectures. It's worth summarizing them.

Architecture demands a holistic perspective to portray the overall structure of a process, system or infrastructure. Design focuses primarily on individual architectural components and how they fit together. Architectures have many benefits, but the most important one is their ability to manage complexity.

A good architecture should focus on bringing out the most important aspects of the process, system or infrastructure being represented, at the expense, if necessary, of the least critical ones. It should also address emerging issues and accommodate changes to its boundaries, dimensions and content. And it should include next-best alternatives and appropriate compensating controls.

Ease of maintenance over time is a fundamental requirement that runs through all architectures. Fast-changing technical and organizational details should not be expressed within slower-changing policy statements. Content should be structured into temporal levels for the purposes of maintenance.

Time is an important consideration in designing architectures. The world we are attempting to model has at least four dimensions. Modeling the future is useful for various kinds of planning purpose and for analyzing or postulating past events. Four-dimensionalism is a powerful but simple and proven technique for incorporating time in data models.

Computers and networks amplify complexity. This has consequences for both the problem space and the solutions space. A simple control mechanism cannot control a complex situation. There are techniques available to enable us to enhance our capability to control systems and to reduce the complexity of the system we wish to control.

Smart categorization is the key to building architectures that are manageable, flexible, extendible and easy to navigate. The components of an architecture are an integrated combination of heterogeneous components. To piece them together in an organized and navigable way demands insight, experience and know-how.

The underpinning structure of an architecture need not be the same as the views presented to users. The backbone of the architecture is ideally a high-quality data model, designed to be complete, extendible and ontologically sound. The presentation of the architecture should be selective, relevant and understandable to users and expressed in their language.

Aim to be as precise in the selection of words for category headings and contents. Grammar is not precise. Avoid words that have a wide range of alternative meaning and steer clear of abbreviations and acronyms. Focus on the underlying nature of the entities and attributes themselves, rather than selecting the first word that springs to mind.

Excessive focus on the components of an architecture will result in a disjointed, distracting and disruptive approach to its development. Strive for a whole that is bigger than the sum of its parts. Focusing on activities is a powerful means of joining up the individual components of an architecture.

Politics is an important consideration in presenting an architecture to users. The order and relative size of the boxes will imply the relative importance of each process, system or organizational unit. The core structure of the architecture should be independent of politics, but its presentation should be designed with politics in mind.

Architectural components should be carefully selected to support standardization and externalization of processes and technology. They should be secure, flexible, scalable, extendible, widely supported and compliant with regulatory requirements. A forward-looking product or service should be continuously evolving, adding new features and exploiting the latest technologies.

When choosing solutions check contracts carefully for potential high charges for changes in the use of a product or service. Choose a reputable supplier and take advice from an experienced software licensing expert.

Finally, architectural components must fit the emerging Zero Trust paradigm by focusing security at the application and data levels, using strong authentication systems and end-to-end encryption rather than relying exclusively on perimeter security measures.

About

David Lacey is a highly experienced CISO and Cybersecurity thought leader. He founded and led functions for the UK Foreign & Commonwealth Office, Shell International and the British Royal Mail Group. During this time, he developed the set of controls that became the basis of the International Standard ISO/IEC 27001/2 and achieved the world's first and largest accredited certification. He also founded the Jericho Forum (part of the Open Group) to develop the principles and architecture for a De-Perimiterised network environment, the concept for which was subsequently adapted by Forrester Research as Zero Trust. David has written five books and numerous papers on Cybersecurity and is a keen futurist, having been the first CISO to recognise and articulate the impact of the Information Age on Cybersecurity. He is also a long-standing member of the Infosecurity Europe Hall of Fame.

Footnotes

1 The "built environment" refers to the human-made surroundings that provide the setting for human activity. It encompasses everything from individual buildings and infrastructure to public spaces and green areas within a city or town.

2 Perhaps this reflects a profession in which little has changed in best practice in the last 50 years.

3 By "high-quality data modelling," I refer to the type expressed in ISO 15926-2:2003, an international standard for data integration, sharing and exchange. This subject is further explained in Dr. Matthew West's excellent book "Designing High Quality Data Modelling," published by Morgan Kaufmann in 2011.

4 Technical debt is the cost of additional work in the future resulting from the choice of an expedient solution rather than a more robust one.

5 A minimum viable product (MVP) has just enough core features to effectively deploy the product.

6 Brilliant scientists whose ideas I have drawn on include the outstanding mathematician and philosopher Alfred North Whitehead (Bertram Russell's tutor) and the eminent theoretical physicist David Bohr.

7 From an ontological perspective, processes and activities are identical. Both can have subactivities and can be subactivities of other activities. The word "process," however, generally indicates a higher-level aggregation of activities.

8 A business cannot operate without an architecture. It must exist in practice though it may not be fully documented or collected into a single framework.

9 Object-oriented programming (OOP) combines both code and data, but the optimum structure of data models and object models will be different, as they are subject to different considerations and uses.

10 In first-order predicate calculus, we can declare that there must exist a particular thing with a specific attribute, or we might demand that all objects of a certain type must have a particular attribute.

11 If the two-dimensional diagram is the aspirin of architecture, then we should ask "What are the vitamins?" Hopefully, this will become clearer throughout the paper.

12 Imagine a future executive boardroom with screens showing the real time status of all critical business operations and supporting services. Technology exists to enable this, but it is rarely deployed at present.

13 Hamlet Act 2, Scene 2.

14 A compensating control is an alternative measure adopted to compensate for a specified requirement that is deemed to be too difficult, expensive or impractical to implement, at least at the present time.

15 Some quantum scientists claim that a proton particle can have up to 37 dimensions of quantum state.

16 Are alternative worlds real or theoretical? It's a matter of opinion. Personally, I believe them to be real, as I subscribe to the many-worlds interpretation (MWI) of quantum mechanics. But if you're a supporter of the Copenhagen interpretation, then you might not.

17 Personally, I would argue that activities are more real than physical objects. (More on this later.)

18 See https://tedsider.org/papers/4d.pdf for an academic analysis of the difference between three-dimensionalism and four-dimensionalism.

19 This is called an extensional set definition, as opposed to an intensional set definition.

20 One might expect states to be a subtype (child entity) of the object in question, but in fact the whole-life of the object is simply one of the many different states of an object.

21 In the context of this paper, complexity is measured by the number of states a system can be in. The measure is referred to as variety.

22 Interestingly, the only univocal words I can find are words such as unambiguous and specific.

23 For the literary minded, this latter point is nicely expressed in Mikhail Bulgakov's classic book "The Master and Margarita" by the insightful and powerful quotation: "Manuscripts don't burn."

24 The Oxford English Dictionary's entry for "run" contains 75 columns of type and is reported to have taken a professional lexicographer nine months of research to complete.

25 Note that long field names do not necessarily result in large database records or inefficient processing, as the text can be placed in a look-up table.

26 See "Wholeness and the Implicate Order" by David Bohm (ISBN-13: 978-0415289795).

27 The Second Law of Thermodynamics states that the total entropy (disorder) of an isolated system can only increase over time (but that of course is in the absence of interventions).

28 Alfred North Whitehead was an English mathematician and philosopher who created the philosophical school known as process philosophy. In his early career, Whitehead wrote "Principia Mathematica" with his former student Bertrand Russell, which is considered one of the 20th century's most important works in mathematics.

29 Activities include computer processing and other forms of automation.

30 Physical security is slightly different as a static steel door can provide security, but only if installed and locked by some form of operational activity.

31 The recommended data model for an activity can be found in the international standard ISO 15926-2 referenced earlier in this paper.

32 One is reminded of the widely criticized circular design of Paris's Charles de Gaulle Airport's Terminal 1, whose labyrinthine layout and long walking distances were highly unpopular with passengers and were extremely difficult and expensive to expand, even though the architect set out to create a futuristic and aesthetically pleasing structure.

33 A cash cow is a product that generates a steady profit with little ongoing investment or support required.

Would you like to discuss these insights with an expert? **Contact our team.**

About Progress Software

Progress Software (Nasdaq: PRGS) empowers organizations to achieve transformational success in the face of disruptive change. Our software enables our customers to develop, deploy and manage responsible AI-powered applications and digital experiences with agility and ease. Customers get a trusted provider in Progress, with the products, expertise and vision they need to succeed. Over 4 million developers and technologists at hundreds of thousands of enterprises depend on Progress. Learn more at www.progress.com

Worldwide Headquarters

Progress Software Corporation
15 Wayside Rd, Suite 400, Burlington, MA 01803, USA
Tel: +1-800-477-6473

facebook.com/progresssw
twitter.com/progresssw
youtube.com/progresssw
linkedin.com/company/progress-software
progress_sw_

Progress