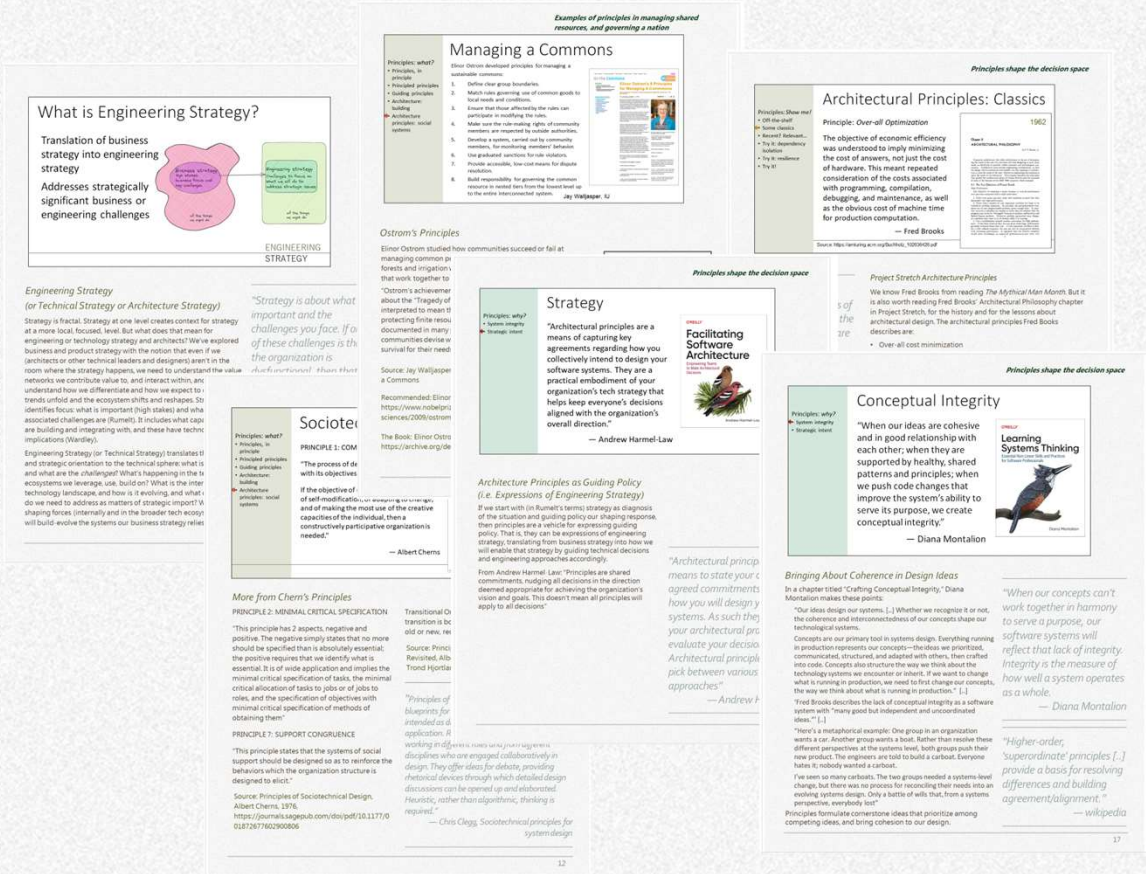
Architecture Principles

Ruth Malan

Bredemeyer Consulting



March 2025

Technical Leadership Workshops

Remote:

- May 8 and 15, 2024, 12pm-3pm Eastern Time (US/Canada).

System Design and Software Architecture Workshops

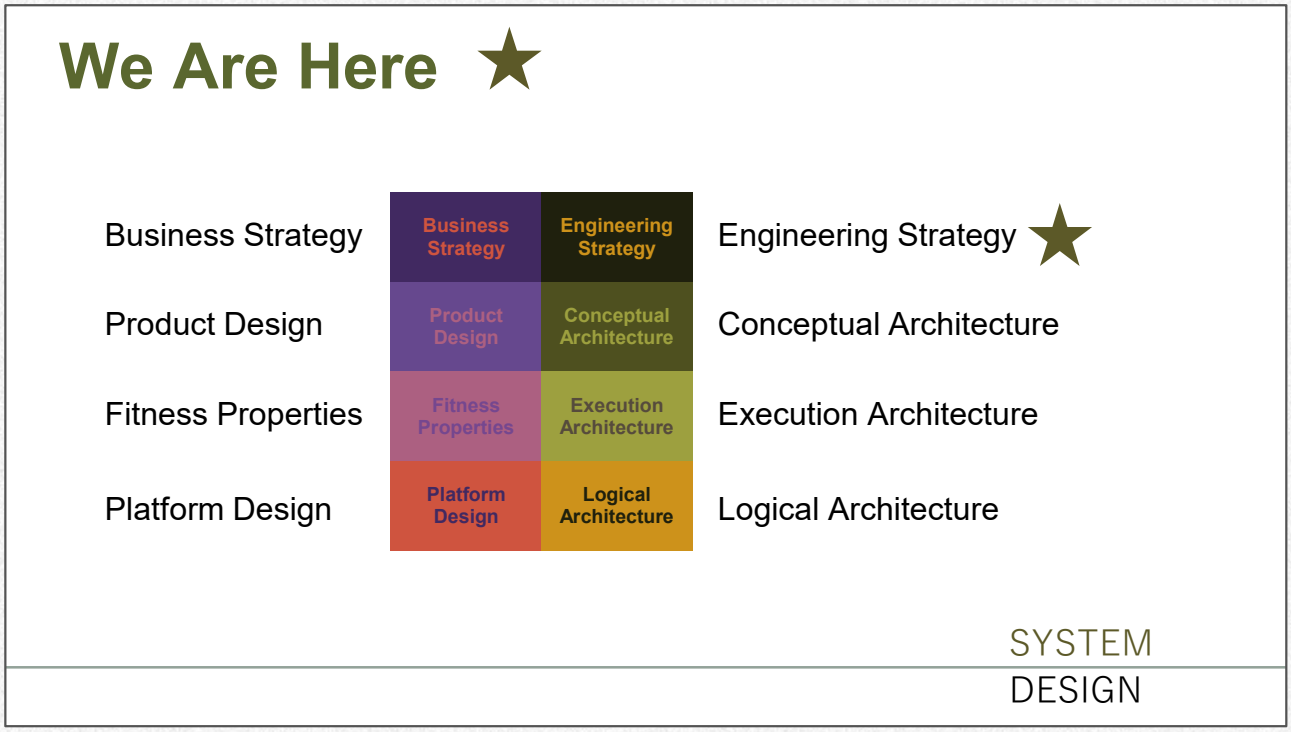
Remote:

- May 12-14 and May 19-20, 11am-3:30pm Eastern Time (US/Canada).

See ruthmalan.com and https://ti.to/bredemeyer/ for schedule and more information.

Attribution — All quotes used in this material, belong to their sources. For original work herein, you must give appropriate credit, provide a link to this material, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. Only noncommercial uses of the work are permitted. Adaptations must be shared under the same terms

discourse (n.): late 14c., "process of understanding, reasoning, thought,"



Technical or Engineering Strategy

Ecosystems or ecologies of (sociotechnical) systems, plays a role in business and product strategy. This strategy determines where we will play or compete in the ecosystem, how we will create unique, differentiating value (within our org, to other partners, and to customers) so we can survive and even thrive. Situation understanding or competitive landscape understanding, therefore, means we need to understand this landscape -- the ecosystem(s) we play in, and how the capabilities (we compose into systems) offer unique and compelling value. etc. These are techno-economic, value exchanging ecologies of interacting socio-technical systems. Add too, political forces and geopolitical factors.

Ecosystems also play a role, as we consider what capabilities we will build internally, to be viable as a complex (open) system, with the capabilities that enable us to play the roles the org is striving to play, in the ecosystems of broader value flows and transformations. For example, microservices rely not only on architectural approaches to decomposing the system and mechanisms for coherence and integrity despite the challenges posed by distributed systems, but also on development infrastructure for automated builds and continuous delivery (like Puppet and Chef), and the infrastructure of containers and orchestration tools (like Kubernetes). Our engineering strategy identifies and addresses the over-arching challenges we face, and the capabilities we need to build.

Strategy asks: what's going on? And what are we going to do?

What is Strategy?

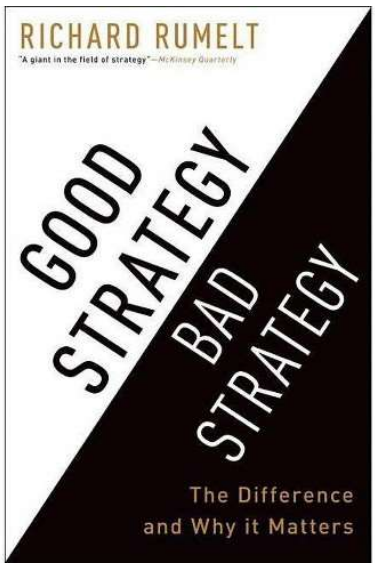
For Rumelt, strategy contains three elements:

1) a diagnosis *situation awareness*

2) a guiding policy *response*

3) coherent action

coherent in good part because we have a basis for coherence



STRATEGY

Strategy: Diagnosis, Guiding Policy, aaand Action!

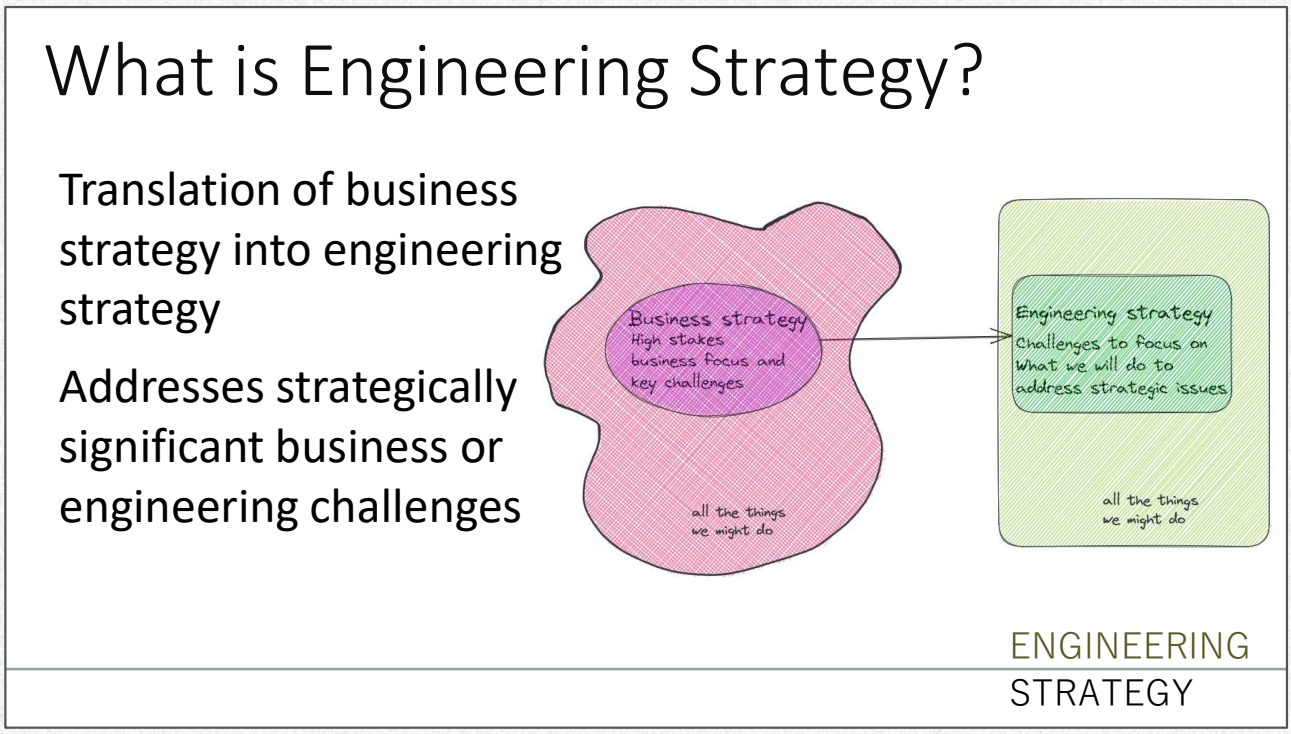
"At a minimum, a diagnosis names or classifies the situation [..] suggesting more attention be paid to some issues and less to others. An especially insightful diagnosis can transform one's view of the situation, bringing a radically different perspective to bear. [..] An explicit diagnosis permits one to evaluate the rest of the strategy. Additionally, making the diagnosis an explicit element of strategy allows the rest of the strategy to be revisited and changed as circumstances change." (Richard Rumelt)

We talked about "chicken and egg" problems as an example of diagnosing a situation (in scaling the startup phases or eXplore→eXpand in Beck's 3X). We might also want to think about how we gain momentum for an architectural shift (say, as we move from scaling to scope and need to recover from technical debt, or shift to a new S-curve). Even when we have senior executive support, that's not enough; organizational change needs new capabilities and practices. Are we enabling those? Understanding of why? Time to learn? A culture that supports learning?

"Drawing the map: [..] observed that great leaders recognize an issue before it becomes an emergency. They consistently map the changing dynamics of the company's environment and create a clear, prioritized vision for where the business should be headed."

— B Tom Hunsaker and Jonathan Knowles

Engineering strategy translates business strategy into technical direction and addresses key engineering challenges



Engineering Strategy (or Technical Strategy or Architecture Strategy)

Strategy is fractal. Strategy at one level creates context for strategy at a more local, focused, level. But what does that mean for engineering or technology strategy and architects? We've explored business and product strategy with the notion that even if we (architects or other technical leaders and designers) aren't in the room where the strategy happens, we need to understand the value networks we contribute value to, and interact within, and understand how we differentiate and how we expect to do so as trends unfold and the ecosystem shifts and reshapes. Strategy identifies focus: what is important (high stakes) and what the associated challenges are (Rumelt). It includes what capabilities we are building and integrating with, and these have technology implications (Wardley).

Engineering Strategy (or Technical Strategy) translates this attention and strategic orientation to the technical sphere: what is *important* and what are the *challenges*? What's happening in the technical ecosystems we leverage, use, build on? What is the internal technology landscape, and how is it evolving, and what challenges do we need to address as matters of strategic import? What are the shaping forces (internally and in the broader tech ecosystem)? How will build-evolve the systems our business strategy relies on?

> *"Strategy is about what is important and the challenges you face. If one of these challenges is that the organization is dysfunctional, then that's strategic. If your managers are not managing properly, if the organization lacks the resilience it needs for the business it's in, that is a strategic issue."*
>
> *— Richard Rumelt*

Source: Why bad strategy is a 'social contagion', Richard Rumelt interviewed by Yuval Atsmon

Interrogatives As Structuring Device Principles:
- What?
- Why?
- How?
- Show me!
- Who?
- How, who, when, where?

Architecture Principles

Principles: *what?*
- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles: building
- Architecture principles: systems
- Governing the system, guiding designers, and design of design

Principles: *why?*
- System integrity
- Strategic intent

Principles: *how?*
- Template
- But seriously, how?

Principles: *show me!*
- Off-the-shelf
- Some classics
- Recent? Relevant…
- Try it: dependency isolation
- Try it: resilience
- Try it!

Principles: *how, who, when, where*
- Strategy: diagnosis and guiding policy
- Integrity, common ground and framing intent
- Workshopping principles

Principles and What's Ahead

Architecture principles are decisions that set direction, rule sets of decisions out, shape our actions, guide our work so we bring experience and coherent action to bear and achieve more the outcomes we intend. Well, that packs in a lot, so over the next pages we'll explore more what we mean and what the implications are.

This work builds on (and replaces) the previous iteration: https://ruthmalan.com/ByTopic/architecture/202102ArchitecturePrinciples.pdf

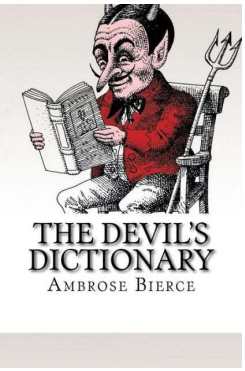
"I love thinking about the word DECISION through the lens of "what am I cutting off?"
It causes me a bit of a pause because in choosing a path of action, I'm not choosing other paths of action (at least at that time)
— Eb Ikonne

For us, a (design) principle guides (design) decisions and actions

Principles: *what?*
- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles
- Architecture principles
- Governing the system, guiding designers, and design of design

Principles: what?

Alas, it's not in the dictionary. Well, not this one:



TheFreeDictionary

prin·ci·ple (prĭn'sə-pəl)

n.

1. A basic truth, law, or assumption: *the principles of democracy.*

2.

a. A rule or standard, especially of good behavior: *a man of principle.*

b. The collectivity of moral or ethical standards or judgments: *a decision based on principle rather than expediency.*

3. A fixed or predetermined policy or mode of action.

4. A basic or essential quality or element determining intrinsic nature or characteristic behavior: *the principle of self-preservation.*

5. A rule or law concerning the functioning of natural phenomena or mechanical processes: *the principle of jet propulsion.*

6. *Chemistry* One of the elements that compose a substance, especially one that gives some special quality or effect.

7. A basic source. See Usage Note at **principal**.

Source: https://https://www.thefreedictionary.com/principle

Principles

Principle has a range of meanings, including:

- law concerning the functioning of natural phenomena or mechanical processes ("the principle of jet propulsion" or "Archimedes principle, relating buoyancy to the weight of displaced water")
- maxim in philosophy or ethics: a moral rule; often pedagogical and motivates or guides specific actions
- guiding principle (of substantive influence when making a decision or considering a matter)
- rule or predetermined policy or mode of action.

Two that stand out (from Cambridge.org):

- a rule or belief that influences your behaviour and which is based on what you think is right
- a basic idea or rule that explains or controls how something happens or works

And then there's *in* principle. "If you agree with something in principle, you agree in general terms to the idea of it, although you do not yet know the details or know if it will be possible."

We'll agree, in principle, that architecture or design principles are useful, even if there are differences in whether they establish *guidance* (informs and influences) or *rules* (musts; official policy) to be followed.

"one of the fundamental tenets or doctrines of a system, a law or truth on which others are founded"

— etymonline.com

Principles as inherent laws
Principles as imposed laws
Principles as guidelines

Various sources including Wikipedia, *Collins Dictionary*, *Cambridge Dictionary* and *American Heritage Dictionary*

"a rule or belief that influences your behaviour and which is based on what you think is right"

Principles: *what?*
- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles

Principled Principles

Principles behind the Agile Manifesto

We follow these principles:

principles

values

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Source: https://agilemanifesto.org/

Principles behind the Agile Manifesto (continued)

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Source: https://agilemanifesto.org/

"the guiding practices that support teams in implementing and executing with agility"

Source: https://www.agilealliance.org/

"Principles unpack the values underlying them more concretely so that the values can be more easily operationalized in policy statements and actions."

—Wikipedia

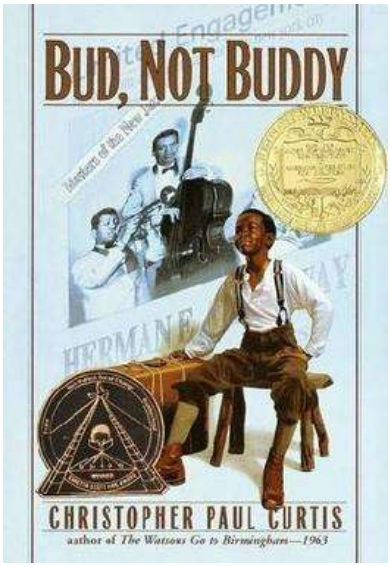
Principles: *what?*

- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles (yes, the other kind)

Guiding Principles

Bud Caldwell's Rules and Things:

"No. 83: If an adult tells you not to worry, and you weren't worried before, better hurry up and start 'cause you're already running late."



Rules, Heuristics, Principles

Bud, Not Buddy is a children's story about a boy who sets out across Depression era America to find his father after his mother has died. The relevance to our discussion here, is that based on his experiences and situations, Bud extracts "rules" to help him through situations down the road. He's looking for what helps him cope, or saves him from harm, including being misled into hope and trust when that's worked out badly for him. So these are his "rules" and they could be rules of thumb (heuristics) or rules he tries to hold himself too. They are propositions that will serve to guide his behavior. He's looking for ways to make his (encounters with) world less erratic, and more navigable.

His Rule #3

> "If you got to tell a lie, make sure it's simple and easy to remember."

has echoes in guidance on naming architecture principles; make them:

> Clear, precise, and easy to remember.

Drawing out that parallel is playful. But it is worth noting the structure of Bud's rules: they give a context, and guidance to his future self, when that situation arises.

Rule #29 is "When You Wake Up and Don't Know for Sure Where You're. At and There's a Bunch of people Standing Around You. It's Best to pretend You're Still Asleep, Until You Can Figure Out What's Going On and What You Should Do."

— Bud's Rules and Things
Christopher Paul Curtis

But··· relevance? To us??

Principles: *what?*
- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles (yes, the other kind)

Gothic Architecture

Gothic cathedrals (late 12th century to the 16th century) are characterized by extraordinary verticality and light.

Purpose gives rise to desired properties, achieved by solving technical problems.

Image: https://upload.wikimedia.org/wikipedia/commons/

Just Enough Design Upfront

This description of Gothic architecture is interesting, in that, by working within a small set of structural principles, the various builders and artisans gained considerable freedom to innovate:

"guilds of masons and builders carried from one diocese to another their constantly increasing stores of constructive knowledge. By a wise division of labor, each man wrought only such parts as he was specially trained to undertake. The master-builder—bishop, abbot, or mason—seems to have planned only the general arrangement and scheme of the building, leaving the precise form of each detail to be determined as the work advanced, according to the skill and fancy of the artisan to whom it was entrusted. Thus was produced that remarkable variety in unity of the Gothic cathedrals; thus, also, those singular irregularities and makeshifts, those discrepancies and alterations in the design, which are found in every great work of medieval architecture. Gothic architecture was constantly changing, attacking new problems or devising new solutions of old ones. In this character of constant flux and development it contrasts strongly with the classic styles, in which the scheme and the principles were easily fixed and remained substantially unchanged for centuries."

"replaced the construction system based on thick load-bearing walls in favour of a structure - called a skeletal system - that freed itself of all superfluous parts by identifying the forces acting on the interior - the thrusts of the vaults and the weight of the roof and walls - so as to direct them along predetermined routes"

Source: https://www.zeepedia.com/read.php?gothic_architecture_structural_principles_ribbed_vaulting_history_of_architecture

Principles: *what?*
- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles (yes, the other kind)

Gothic Architecture: Principles

Key principles enabling distinctive features:

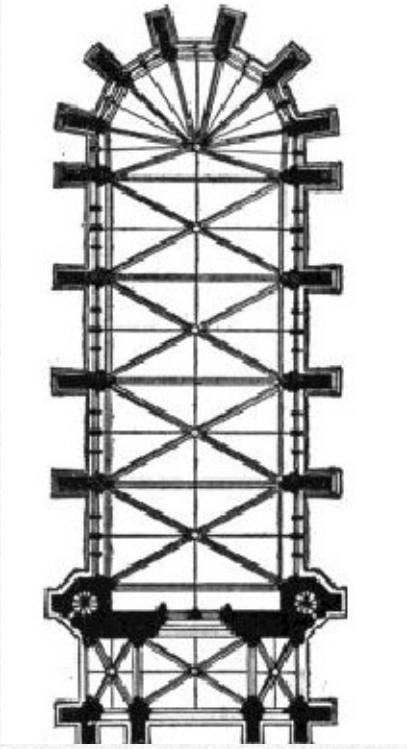
- *concentration of strains* upon isolated points of support
- *balanced thrusts*

FIG. 105.--CONSTRUCTIVE SYSTEM OF GOTHIC CHURCH, ILLUSTRATING PRINCIPLES OF ISOLATED SUPPORTS AND BUTTRESSING.

Structural Principles

"What really distinguished [Gothic architecture] most strikingly was the systematic application of two principles [..]. The first of these was the *concentration of strains* upon isolated points of support, made possible by the substitution of groined [ribbed] for barrel vaults. This led to a corresponding concentration of the masses of masonry at these points; the building was constructed as if upon legs (Fig. 105). The wall became a mere filling-in between the piers or buttresses, and in time was, indeed, practically suppressed, immense windows filled with stained glass taking its place."

Source: https://www.zeepedia.com



"The second distinctive principle of Gothic architecture was that of *balanced thrusts*. In Roman buildings the thrust of the vaulting was resisted wholly by the inertia of mass in the abutments. In Gothic architecture thrusts were as far as possible resisted by counter-thrusts, and the final resultant pressure was transmitted by flying half-arches across the intervening portions of the structure to external buttresses placed at convenient points. This combination of flying half-arches and buttresses is called the flying-buttress."

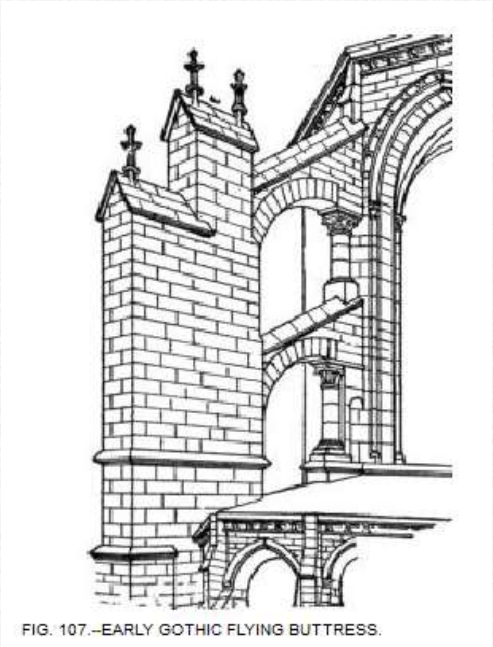


FIG. 107.--EARLY GOTHIC FLYING BUTTRESS.

Principles: *what?*

- Principles, in principle
- Principled principles
- Guiding principles
- Architecture: building
- Architecture principles: social systems

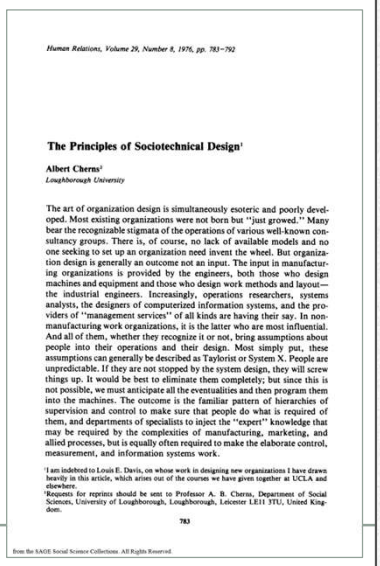
Sociotechnical Design

PRINCIPLE 1: COMPATIBILITY

"The process of design must be compatible with its objectives. [..]

If the objective of design is a system capable of self-modification, of adapting to change, and of making the most use of the creative capacities of the individual, then a constructively participative organization is needed."

— Albert Cherns



Human Relations, Volume 29, Number 8, 1976, pp. 783–792

The Principles of Sociotechnical Design[1]

Albert Cherns[2]
Loughborough University

The art of organization design is simultaneously esoteric and poorly developed. Most existing organizations were not born but "just growed." Many bear the recognizable stigmata of the operations of various well-known consultancy groups. There is, of course, no lack of available models and no one seeking to set up an organization need invent the wheel. But organization design is generally an outcome not an input. The input in manufacturing organizations is provided by the engineers, both those who design machines and equipment and those who design work methods and layout—the industrial engineers. Increasingly, operations researchers, systems analysts, the designers of computerized information systems, and the providers of "management services" of all kinds are having their say. In non-manufacturing work organizations, it is the latter who are most influential. And all of them, whether they recognize it or not, bring assumptions about people into their operations and their design. Most simply put, these assumptions can generally be described as Taylorist or System X. People are unpredictable. If they are not stopped by the system design, they will screw things up. It would be best to eliminate them completely; but since this is not possible, we must anticipate all the eventualities and then program them into the machines. The outcome is the familiar pattern of hierarchies of supervision and control to make sure that people do what is required of them, and departments of specialists to inject the "expert" knowledge that may be required by the complexities of manufacturing, marketing, and allied processes, but is equally often required to make the elaborate control, measurement, and information systems work.

[1] I am indebted to Louis E. Davis, on whose work in designing new organizations I have drawn heavily in this article, which arises out of the courses we have given together at UCLA and elsewhere.
[2] Requests for reprints should be sent to Professor A. B. Cherns, Department of Social Sciences, University of Loughborough, Loughborough, Leicester LE11 3TU, United Kingdom.

from the SAGE Social Science Collections. All Rights Reserved.

More from Chern's Principles

PRINCIPLE 2: MINIMAL CRITICAL SPECIFICATION

"This principle has 2 aspects, negative and positive. The negative simply states that no more should be specified than is absolutely essential; the positive requires that we identify what is essential. It is of wide application and implies the minimal critical specification of tasks, the minimal critical allocation of tasks to jobs or of jobs to roles, and the specification of objectives with minimal critical specification of methods of obtaining them"

PRINCIPLE 7: SUPPORT CONGRUENCE

"This principle states that the systems of social support should be designed so as to reinforce the behaviors which the organization structure is designed to elicit."

Source: Principles of Sociotechnical Design, Albert Cherns, 1976, https://journals.sagepub.com/doi/pdf/10.1177/001872677602900806

Transitional Organization – "an organisation in transition is both different and more complex than old or new, requiring careful planning and design. "

Source: Principles of Sociotechnical Design Revisited, Albert Cherns, 1987 (as conveyed by Trond Hjortland)

"Principles of this kind are not offered as blueprints for strict adherence. They are not intended as design rules for mechanistic application. Rather, they provide inputs to people working in different roles and from different disciplines who are engaged collaboratively in design. They offer ideas for debate, providing rhetorical devices through which detailed design discussions can be opened up and elaborated. Heuristic, rather than algorithmic, thinking is required."

— Chris Clegg, Sociotechnical principles for system design

Principles: *what?*

- Principles, in principle
- Principled principles
- Guiding principles
- Architecture: building
- Architecture principles: social systems

Managing a Commons

Elinor Ostrom developed principles for managing a sustainable commons:

1. Define clear group boundaries.
2. Match rules governing use of common goods to local needs and conditions.
3. Ensure that those affected by the rules can participate in modifying the rules.
4. Make sure the rule-making rights of community members are respected by outside authorities.
5. Develop a system, carried out by community members, for monitoring members' behavior.
6. Use graduated sanctions for rule violators.
7. Provide accessible, low-cost means for dispute resolution.
8. Build responsibility for governing the common resource in nested tiers from the lowest level up to the entire interconnected system.

Jay Walljasper, IU

Ostrom's Principles

Elinor Ostrom studied how communities succeed or fail at managing common pool (finite) resources such as grazing land, forests and irrigation waters. She identified a set of principles that work together to sustainably manage a commons.

"Ostrom's achievement effectively answers popular theories about the "Tragedy of the Commons", which has been interpreted to mean that private property is the only means of protecting finite resources from ruin or depletion. She has documented in many places around the world how communities devise ways to govern the commons to assure its survival for their needs and future generations."

Source: Jay Walljasper, Elinor Ostrom's 8 Principles for Managing a Commons

Recommended: Elinor Ostrom's Nobel acceptance speech, https://www.nobelprize.org/prizes/economic-sciences/2009/ostrom/lecture/

The Book: Elinor Ostrom, Governing the Commons, https://archive.org/details/governingthecommons/mode/2up

Checks and balances: separate branches are empowered to prevent actions by other branches and are induced to share power.

"the Constitution itself is a set of principles for building a very complex dynamic structure that should last for centuries"

—Alan Kay

Principles help guide decisions to achieve more the strategic outcomes we want.

Principles: *what?*

- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles: building
- Architecture principles: systems

Architecture Principles

"a fundamental truth or proposition serving as the foundation for belief or action"

— OED

"When something is causing pain… do something! "

— Architecture Principles, HP OWEN Architecture

Source: Jacob Refstrup, sei.cmu.edu/splc2009

Principles in Principle

From Eoin Woods (in InfoQ): we "can define a software design principle as

> a fundamental truth or proposition serving as the foundation for action with regard to deciding on a software system's workings.

The key point is that a principle is a clear statement of intent that guides our design work."

Jeff Eaton describes them this way:

> "Principles are shared perspectives that affect the decisions you make. Another way of saying it might be, they're lenses you filter your goals through to determine what approaches are reasonable or acceptable. Some of your principles might be assumptions about the way the world works. Others might be beliefs about what matters and what doesn't matter, or moral and ethical lines that you're not willing to cross, no matter what the goal.
>
> A good rule of thumb is that new goals shouldn't change your principles. If they do, you're not really describing principles, just preferences. Consistent principles are one of the key ways Doctrine provides continuity when everything else is changing."

"Architectural principles are a means of capturing key agreements regarding how you collectively intend to design your software systems.

— Andrew Harmel-Law

Doctrine: "a distinct set of beliefs about how the work should be done"
Principles: "fundamental rules and assumptions that guide our thinking"

— Jeff Eaton

Sources: Eoin Woods, https://www.infoq.com/articles/architectural-design-principles/

Jeff Eaton, The Doctrine Gap, https://eaton.fyi/talks/the-doctrine-gap/

Andrew Harmel-Law, *Facilitating Software Architecture*, 2024

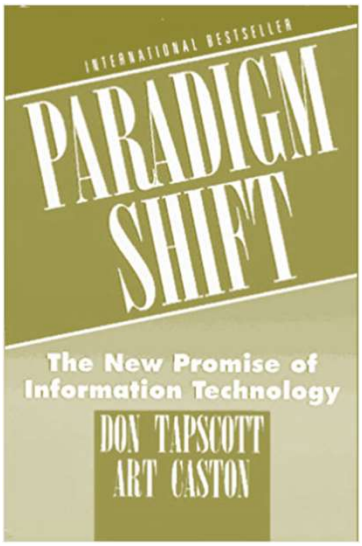
Architecture principles are agreements we intend to follow; they should be countably few and well-formed

Principles: *what?*
- Principles, in principle
- Principled principles
- Guiding principles
- Architecture principles (building)
- Architecture principles (systems)

Architecture Principles

An Architecture Principle orients and aligns decisions and actions to achieve strategic outcomes.

Well-stated principles make it clear when decisions are in line with the principle and when they run counter to the principle.



"Architectural principles ensure that some aspect of design decisions meet some aspect of the requirements. [..] epitomize architecture's function: to clearly define the necessary constraints on a system's design without prescriptively defining all the design details."

— Eoin Woods

Template for Architecture Principles

Principle name: memorable, clear name that indicates intent

Statement: clear statement that will guide decisions; express a technical strategy to achieve the strategic intent, address an architectural challenge, or resolve architectural risk

Driver: make the driver behind the principle clear

Rationale: why we should follow the principle; identifies benefits we get from following the principle (motivating why we have to change what we do); provides traceability to strategy or system/architecture objective the principle will help us meet

Counter forces (counterarguments or alternatives considered): provides a place to say we recognize what factors weigh against the principle and what other approaches we might take (that other people in our environment would argue for) and why we shouldn't do that. One way to think about the counterargument/counter force, is that it illuminates implications/things that need to be done to ensure the principle is followed/viable in the social/business/technical context. It provides a place to say "in the face of dilemmas and tradeoffs and pressures we tend to do this, and this is how it hurts us" and "we could alternatively take this other approach, and this is how that would hurt us."

Consequences: identify (positive and negative) side-effects

Implications: what we have to do to as a result of and to facilitate following the principle

Scope/Applicability/Timeframe (optional, as relevant): identify (and limit) where it is acceptable to deviate from the principle

Source: Based on combination of *Paradigm Shift*, Eoin Woods in https://www.infoq.com/articles/architectural-design-principles/ and our work

Principles: *why?*
- System integrity
- Strategic intent

Principles: Why?

"The universal adoption of several guiding principles helped ensure the conceptual integrity of a plan whose many detailed decisions were made by many contributors."

— Fred P Brooks

1962

Chapter 2
ARCHITECTURAL PHILOSOPHY

by F. P. Brooks, Jr.

Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints. Architecture must include engineering considerations, so that the design will be economical and feasible; but the emphasis in architecture is upon the needs of the user, whereas in engineering the emphasis is upon the needs of the fabricator. This chapter describes the principles that guided the architectural phase of Project Stretch and the rationale of some of the features of the IBM 7030 computer which emerged.

2.1. The Two Objectives of Project Stretch

High Performance

The objective of obtaining a major increase in over-all performance over previous computers had a triple motivation.

1. There were some real-time tasks with deadlines so short that they demanded very high performance.
2. There were a number of very important problems too large to be tackled on existing computers. In principle, any general-purpose computer can do any programmable problem, given enough time. In practice, however, a problem can require so much time for solution that the program may never be "debugged" because of machine malfunctions and limited human patience. Moreover, problem parameters may change, or a problem may cease to be of interest while it is running.
3. Cost considerations formed another motivation for high performance. It has been observed that, for any given technology, performance generally increases faster than cost. A very important corollary is that, for a fully utilized computer, the cost per unit of computation declines with increasing performance. It appeared that the Stretch computer would show accordingly an improved performance-to-cost ratio over

5

Conceptual Integrity

In an earlier section, we explored conceptual integrity. In *Mythical Man Month*, Fred Brooks notes that the lack of conceptual integrity is evident when there are "many good but unco-ordinated ideas." A (small, so cognitively and organizationally tractable) set of architecture principles can help co-ordinate and align design decisions, to bend the system arc more towards conceptual integrity.

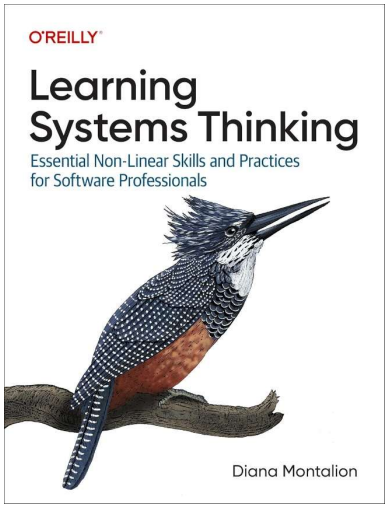
Reminder (harks back to earlier section): Why conceptual integrity and coherence matters

Principles: *why?*
- System integrity
- Strategic intent

Conceptual Integrity

"When our ideas are cohesive and in good relationship with each other; when they are supported by healthy, shared patterns and principles; when we push code changes that improve the system's ability to serve its purpose, we create conceptual integrity."

— Diana Montalion



Bringing About Coherence in Design Ideas

In a chapter titled "Crafting Conceptual Integrity," Diana Montalion makes these points:

> "Our ideas design our systems. [..] Whether we recognize it or not, the coherence and interconnectedness of our concepts shape our technological systems.
>
> Concepts are our primary tool in systems design. Everything running in production represents our concepts—the ideas we prioritized, communicated, structured, and adapted with others, then crafted into code. Concepts also structure the way we think about the technology systems we encounter or inherit. If we want to change what is running in production, we need to first change our concepts, the way we think about what is running in production." [..]
>
> 'Fred Brooks describes the lack of conceptual integrity as a software system with "many good but independent and uncoordinated ideas."' [..]
>
> "Here's a metaphorical example: One group in an organization wants a car. Another group wants a boat. Rather than resolve these different perspectives at the systems level, both groups push their new product. The engineers are told to build a carboat. Everyone hates it; nobody wanted a carboat.
>
> I've seen so many carboats. The two groups needed a systems-level change, but there was no process for reconciling their needs into an evolving systems design. Only a battle of wills that, from a systems perspective, everybody lost"

Principles formulate cornerstone ideas that prioritize among competing ideas, and bring cohesion to our design.

"When our concepts can't work together in harmony to serve a purpose, our software systems will reflect that lack of integrity. Integrity is the measure of how well a system operates as a whole.

— Diana Montalion

"Higher-order, 'superordinate' principles [..] provide a basis for resolving differences and building agreement/alignment."

— wikipedia

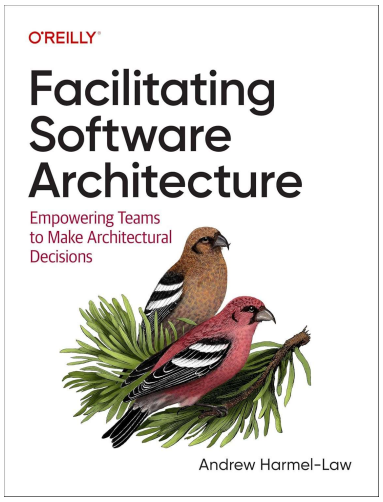
Principles interpret and make some aspect of strategy more actionable

Principles: *why?*
- System integrity
- Strategic intent

Strategy

"Architectural principles are a means of capturing key agreements regarding how you collectively intend to design your software systems. They are a practical embodiment of your organization's tech strategy that helps keep everyone's decisions aligned with the organization's overall direction."

— Andrew Harmel-Law



Architecture Principles as Guiding Policy (i.e. Expressions of Engineering Strategy)

If we start with (in Rumelt's terms) strategy as diagnosis of the situation and guiding policy our shaping response, then principles are a vehicle for expressing guiding policy. That is, they can be expressions of engineering strategy, translating from business strategy into how we will enable that strategy by guiding technical decisions and engineering approaches accordingly.

From Andrew Harmel-Law: "Principles are shared commitments, nudging all decisions in the direction deemed appropriate for achieving the organization's vision and goals. This doesn't mean all principles will apply to all decisions"

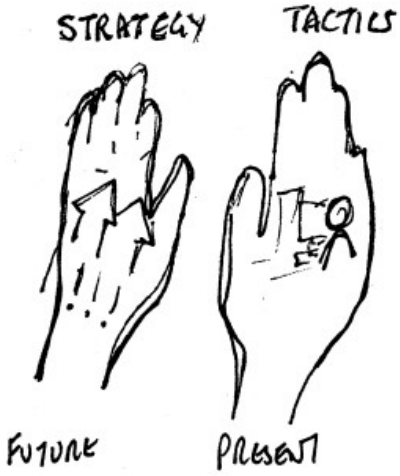
"Architectural principles are a means to state your collectively agreed commitments regarding how you will design your systems. As such they direct your architectural practice and evaluate your decision options. Architectural principles helps pick between various approaches"

— Andrew Harmel-Law

Principles: *why?*
- System integrity
- Strategic intent

Culture and Principles

Architecture Principles are agreements that we reach and share; they're a mechanism to express values in terms of behavioral/decision guidance, and are an important way to influence the (sub)culture of our teams*.



* And a vital way to do that organically, is by co-creating principles together

Principles Impact Design Culture

A principle like "above all do no harm" directs our attention and behavior to avoiding needless injury to others whenever possible. Such principles are normative – they express how things should or ought to be, how to value them, which things are good or bad, which decisions or actions are right or wrong.

Architecture Principles are agreements that we reach and share; they're a mechanism to express values in terms of behavioral/decision guidance, and are an important way to influence the (sub)culture of our teams. They are part of the "left hand" work of guiding and shaping by reducing the decision space, while still preserving degrees of freedom and design discretion.

Inherently we're "giving shape to" the system, influencing the form, aesthetics and properties of the system.

Every principle we add to the architecture, consumes cycles – creating and evolving the principle statement, understanding and following the principle, and governing the principle (to catch misses and help ensure architectural integrity) all costs attention.

We can view Architecture Principles as a lever to help set and maintain direction/course, so long as we don't get overzealous and wash out their impact by having so many that we create cognitive overload and cause them to be ignored.

We will explore principles and Architecture Principles further in this section.

The point is that a principle is worth stating as an Architectural Principle if it makes a difference to decisions/behaviors that have strategic impact.

Some examples of "off-the-shelf" (existing, documented…) principles

Principles: *Show me!*

- Off-the-shelf
- Some classics
- Recent? Relevant…
- Try it: dependency isolation
- Try it: resilience
- Try it!

Architectural Principles: TOGAF

2.6.1 Business Principles

Principle 4: Business Continuity

Statement:

Enterprise operations are maintained in spite of system interruptions.

Rationale:

As system operations become more pervasive, we become more dependent on them; therefore, we must consider the reliability of such systems throughout their design and use. Business premises throughout the enterprise must be provided with the capability to continue operations regardless of external events. Hardware failure, natural disasters, and data corruption should not be allowed to disrupt or stop enterprise activities. The enterprise must be capable of operating on alternative information delivery mechanisms.

2.6.3 Application Principles

Principle 16: Technology Independence

Statement:

Applications are independent of specific technology choices and therefore can operate on a variety of technology platforms.

Rationale:

Independence of applications from the underlying technology allows applications to be developed, upgraded, and operated in the most cost-effective and timely way. Otherwise technology, which is subject to continual obsolescence and vendor dependence, becomes the driver rather than the user requirements themselves.
Realizing that every decision made with respect to IT makes us dependent on that technology, the intent of this principle is to ensure that Application Software is not dependent on specific hardware and operating systems software.

Implications:

- This principle will require standards which support portability
- For Commercial Off-The-Shelf (COTS) and Government Off-The-Shelf (GOTS) applications, there may be limited current choices, as many of these applications are technology and platform-dependent
- Subsystem interfaces will need to be developed to enable legacy applications to interoperate with applications and operating environments developed under the Enterprise Architecture
- Middleware should be used to decouple applications from specific software solutions
- As an example, this principle could lead to use of Java®, and future Java-like protocols, which give a high degree of priority to platform-independence

Source: TOGAF https://pubs.opengroup.org/togaf-standard/adm-techniques/chap02.html

"Principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission."

— TOGAF

Source: *Architecture Principles*, TOGAF, https://pubs.opengroup.org/togaf-standard/adm-techniques/chap02.html#tag_02_06_01_06

Principle 7: Compliance with Law

Statement: Enterprise information management processes comply with all relevant laws, policies, and regulations.

Rationale: Enterprise policy is to abide by laws, policies, and regulations. This will not preclude business process improvements that lead to changes in policies and regulations.

Implications:

- The enterprise must be mindful to comply with laws, regulations, and external policies regarding the collection, retention, and management of data
- Education and access to the rules
- Efficiency, need, and common sense are not the only drivers. Changes in the law and changes in regulations may drive changes in our processes or applications.

Principle 6: Service Orientation

Statement: The architecture is based on a design of services which mirror real-world business activities comprising the enterprise (or inter-enterprise) business processes.

Rationale: Service orientation delivers enterprise agility and Boundaryless Information Flow.

Implications:

- Service representation utilizes business descriptions to provide context (i.e., business process, goal, rule, policy, service interface, and service component) and implements services using service orchestration
- Service orientation places unique requirements on the infrastructure, and implementations should use open standards to realize interoperability and location transparency
- Implementations are environment-specific; they are constrained or enabled by context and must be described within that context
- Strong governance of service representation and implementation is required
- A "Litmus Test", which determines a "good service", is required

Principles: *Show me!*
- Off-the-shelf
- Some classics
- Recent? Relevant…
- Try it: dependency isolation
- Try it: resilience
- Try it!

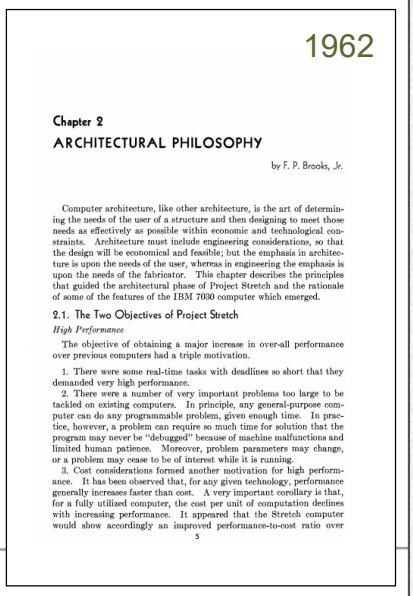
Architectural Principles: Classics

Principle: *Over-all Optimization*

The objective of economic efficiency was understood to imply minimizing the cost of answers, not just the cost of hardware. This meant repeated consideration of the costs associated with programming, compilation, debugging, and maintenance, as well as the obvious cost of machine time for production computation.

— Fred Brooks

Source: https://amturing.acm.org/Buchholz_102636426.pdf



1962

Chapter 2
ARCHITECTURAL PHILOSOPHY

by F. P. Brooks, Jr.

Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints. Architecture must include engineering considerations, so that the design will be economical and feasible; but the emphasis in architecture is upon the needs of the user, whereas in engineering the emphasis is upon the needs of the fabricator. This chapter describes the principles that guided the architectural phase of Project Stretch and the rationale of some of the features of the IBM 7030 computer which emerged.

2.1. The Two Objectives of Project Stretch

High Performance

The objective of obtaining a major increase in over-all performance over previous computers had a triple motivation.

1. There were some real-time tasks with deadlines so short that they demanded very high performance.
2. There were a number of very important problems too large to be tackled on existing computers. In principle, any general-purpose computer can do any programmable problem, given enough time. In practice, however, a problem can require so much time for solution that the program may never be "debugged" because of machine malfunctions and limited human patience. Moreover, problem parameters may change, or a problem may cease to be of interest while it is running.
3. Cost considerations formed another motivation for high performance. It has been observed that, for any given technology, performance generally increases faster than cost. A very important corollary is that, for a fully utilized computer, the cost per unit of computation declines with increasing performance. It appeared that the Stretch computer would show accordingly an improved performance-to-cost ratio over

5

"Although the discussion is in terms of a specific computer, the concepts discussed are quite general. The computer chosen is the IBM 7030"

— Werner Buchholz

Source: *Planning a Computer System: Project Stretch*, Ed Werner Buchholz, 1962; Chapter 2: Architectural Philosophy, by Fred P Brooks

Source: https://amturing.acm.org/Buchholz_102636426.pdf

Project Stretch Architecture Principles

We know Fred Brooks from reading *The Mythical Man Month*. But it is also worth reading Fred Brooks' Architectural Philosophy chapter in Project Stretch, for the history and for the lessons about architectural design. The architectural principles Fred Books describes are:

- Over-all cost minimization
- Power instead of Simplicity
- Generalized Features
- Specialized Equipment for Frequent Tasks
- Systematic Instruction Set
- Precision for New Operating Techniques

Principle: Power Instead of Simplicity

"The user was given power rather than simplicity whenever an equal cost choice had to be made. It was recognized in the first place that the new computer would have many highly sophisticated and experienced users. It would have been presumptuous as well as unwise for the computer designers to "protect" such users from equipment complexities that might be useful for solving complex problems. In the second place, the choice is asymmetric. Powerful features can be ignored by a user who wishes to confine himself [sic] to simple techniques. But if powerful features were not provided, the skillful and motivated user could not wring their power from the computer. "

Principles: *Show me!*
- Off-the-shelf
- Some classics
- Recent? Relevant…
- Try it: dependency isolation
- Try it: resilience
- Try it!

Classics: Unix Principles

1978

(i) Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features.

— Doug McIlroy

McIlroy called these maxims

They are principles as governing design ideas

Source: UNIX Time-Sharing System: Foreword, 1978, https://archive.org/details/bstj57-6-1899/page/n3/mode/2up

Principles Behind Unix Pipes, cont.

(ii) Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.

(iii) Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.

Source: Doug McIlroy (1978), https://archive.org/details/bstj57-6-1899/page/n3/

"This is the Unix philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."

— Doug McIlroy

More of the Unix philosophy was implied not by what these elders said but by what they did and the example Unix itself set. Looking at the whole, we can abstract the following ideas:

1. Rule of Modularity: Write simple parts connected by clean interfaces.
2. Rule of Clarity: Clarity is better than cleverness.
3. Rule of Composition: Design programs to be connected to other programs.
4. Rule of Separation: Separate policy from mechanism; separate interfaces from engines.
5. Rule of Simplicity: Design for simplicity; add complexity only where you must.
6. Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.
7. Rule of Transparency: Design for visibility to make inspection and debuggingeasier.
8. Rule of Robustness: Robustness is the child of transparency and simplicity.
9. Rule of Representation: Fold knowledge into data so program logic can be stupic and robust.
10. Rule of Least Surprise: In interface design, always do the least surprising thing.
11. Rule of Silence: When a program has nothing surprising to say, it should say nothing.
12. Rule of Repair: When you must fail, fail noisily and as soon as possible.
13. Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.
14. Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.
15. Rule of Optimization: Prototype before polishing. Get it working before you optimize it.
16. Rule of Diversity: Distrust all claims for "one true way".
17. Rule of Extensibility: Design for the future, because it will be here sooner than you think.

Eric S. Raymond, Basics of the Unix Philosphy https://cscie2x.dce.harvard.edu/hw/ch01s06.html

Principles: *Show me!*
- Off-the-shelf
- Some classics
- Recent? Relevant…
- Try it: dependency isolation
- Try it: resilience
- Try it!

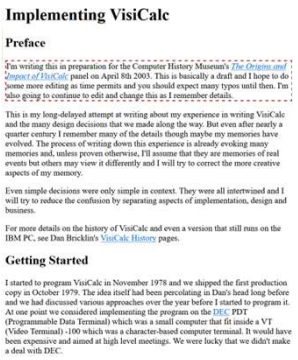
VisiCalc Design Principles

1978

the *Principle of Least Surprise*

"The goal was to give the user a conceptual model which was unsurprising — it was called the principle of least surprise. We were illusionists synthesizing an experience. Our model was the spreadsheet — a simple paper grid that would be laid out on a table. The paper grid provided an organizing metaphor for a working with series of numbers."

— Bob Frankston



Implementing VisiCalc

Preface

I'm writing this in preparation for the Computer History Museum's *The Origins and Impact of VisiCalc* panel on April 8th 2003. This is basically a draft and I hope to do some more editing as time permits and you should expect many typos until then. I'm also going to continue to edit and change this as I remember details.

This is my long-delayed attempt at writing about my experience in writing VisiCalc and the many design decisions that we made along the way. But even after nearly a quarter century I remember many of the details though maybe my memories have evolved. The process of writing down this experience is already evoking many memories and, unless proven otherwise, I'll assume that they are memories of real events but others may view it differently and I will try to correct the more creative aspects of my memory.

Even simple decisions were only simple in context. They were all intertwined and I will try to reduce the confusion by separating aspects of implementation, design and business.

For more details on the history of VisiCalc and even a version that still runs on the IBM PC, see Dan Bricklin's VisiCalc History pages.

Getting Started

I started to program VisiCalc in November 1978 and we shipped the first production copy in October 1979. The idea itself had been percolating in Dan's head long before and we had discussed various approaches over the year before I started to program it. At one point we considered implementing the program on the DEC PDT (Programmable Data Terminal) which was a small computer that fit inside a VT (Video Terminal) -100 which was a character-based computer terminal. It would have been expensive and aimed at high level meetings. We were lucky that we didn't make a deal with DEC.

Source: Implementing VisiCalc, Bob Frankston, https://landley.net/history/mirror/apple2/implementingvisicalc.html

Principle of Least Astonishment

'In user interface design and software design, the principle of least astonishment (POLA), also known as principle of least surprise,[a] proposes that a component of a system should behave in a way that most users will expect it to behave, and therefore not astonish or surprise users. The following is a corollary of the principle: "If a necessary feature has a high astonishment factor, it may be necessary to redesign the feature."

The principle has been in use in relation to computer interaction since at least the 1970s.'

— wikipedia

Source: https://en.wikipedia.org/wiki/Principle_of_least_astonishment

"For those parts of the system which cannot be adjusted to the peculiarities of the user, the designers of a systems programming language should obey the "Law of Least Astonishment." In short, this law states that every construct in the system should behave exactly as its syntax suggests. Widely accepted conventions should be followed whenever possible, and exceptions to previously established rules of the language should be minimal."

— Bergeron et al, "Systems Programming Languages" in Advances in Computers

Principles: *Show me!*
- Off-the-shelf
- Some classics
- Recent
- Try it: dependency isolation
- Try it: resilience
- Try it!

Example Principles

"In my team, we have a bit over a dozen foundational (mostly technical) core principles. We only wrote them down a year ago but they've existed for a decade. I call them the constitution. It's the consensus framework within which we evolve what we build."

— Clemens Vasters

Source: https://twitter.com/clemensv/status/1001129444633870336

SOFTWARE ARCHITECTURE

"With a set of baseline principles, effectively setting guard rails that nobody gets to cross, you can eliminate lots of little conflicts and with a well-thought-out set of boundaries, you can still leave lots of room for innovation."

— Clemens Vasters

Examples of Core Principles

Clemens Vasters is Principal Architect, Messaging and Real-time Intelligence Services, at Microsoft. A few years ago, he shared some of his team's principles, and the reasons for having them, on twitter, including the following:

"They're quite specific to the work we do. Let me share 3; literally:

1: The services implement *open-standards based protocols* as primary form of communication. If a standards-based alternative emerges for a proprietary capability, the proprietary capability is phased out.

2: *Contract is honored*. Protocol enhancements or additions do not break existing functionality. Breaking protocol changes at any level are announced with one year lead time before eventual retirement and removal from the system.

3: All servicing, update, and load balancing activities are performed while keeping SLA

That's the level of base consensus that we have defined. It sets baseline rules. There are also some rules that clearly state what we will not do."

Source: https://twitter.com/clemensv/status/1001133005593858048

Principles: *Show me!*
- Off-the-shelf
- Some classics
- Recent
- Try it: dependency isolation
- Try it: resilience
- Try it!

Rudder API Principles

4 principles

- Assess failure modes
- You are responsible to keep promises made
- Give agency to your users
- And don’t forget any of them

— François Armand

Source: https://speakerdeck.com/fanf42/devoxxfr-2021-systematic-error-management-in-application?slide=15

"You aren't gonna need it" (YAGNI) is a principle which arose from extreme programming (XP) that states a programmer should not add functionality until deemed necessary (Wikipedia)

Do the simplest thing that could possibly work (Ward Cunningham): start with the most straightforward solution to a problem, rather than overcomplicating it

More Examples of Principles

"The *robustness principle* is a design guideline for software that states: "be conservative in what you do, be liberal in what you accept from others". It is often reworded as: "be conservative in what you send, be liberal in what you accept". The principle is also known as Postel's law, after Jon Postel, who used the wording in an early specification of TCP." (Wikipedia) (We might generalize it to a *Play Well With Others* Principle.)

Couterargument: it can make problem identification harder when implementations tolerate deviations from a protocol specification, but then some years later a less tolerant implementation rejects the message. (Marshall Rose, via Wikipedia) Rose therefore recommended "explicit consistency checks in a protocol ... even if they impose implementation overhead" (Wikipedia).

"In 2023, Martin Thomson and David Schinazi argued that Postel's *robustness principle* actually leads to a lack of robustness, including security:

> A flaw can become entrenched as a de facto standard. Any implementation of the protocol is required to replicate the aberrant behavior, or it is not interoperable. This is both a consequence of tolerating the unexpected and a product of a natural reluctance to avoid fatal error conditions. Ensuring interoperability in this environment is often referred to as aiming to be "bug-for-bug compatible". (Wikipedia)

And example with a strong counterexample (see slide and notes)

Example: Architecture Principles

Common Clinical Context Manager Principles

Principle Name	*Technical Neutrality*
Description	The architectural approach should work equally well with any one of a candidate set of relevant technologies.
Rationale / Benefits	Increase acceptance. Selecting a single language or component technology would require some application developers to change their approach in order to use the facility, potentially limiting acceptance and use.
Implications	We will have to carefully select the technologies to support. Increased development time and support effort for us.
Counter argument	Selecting single dominant technologies to support will reduce effort and simplify development and support.

Based on: Common Clinical Context Architecture Specification

"Good architecture principles are constructive, reasoned, well-articulated, testable and significant."

— Eoin Woods

Technology Independence: Counter-Argument

Eberhard Wolff argues against a technology independence principle, and in doing so, gives us a nice example of a counter-argument:

> "The goal of technology independence usually leads to abstractions and indirections being built to be independent of a concrete implementation. These are often limited because they have to implement the lowest common denominator of all possible implementations. In addition, the concrete implementation might leak through the abstraction, so that real independence is not achieved. And finally, technology independence only pays off, when a new technology is actually needed. Until then, you have to "pay" with increased complexity. This is not necessarily the easier way. It is often better to implement technology-dependent and make full use of the technology. Only when a different technology needs to be used, the system is migrated and the complexity cost is paid."

Source: https://www.innoq.com/en/blog/no-principles-software-architecture/

Principles: Guidelines

The principle should help us achieve something strategic or key to the purpose and integrity of the system

- properties of the system
- challenges we face

ENGINEERING
STRATEGY

"Principles constitute Strategy for achieving Quality goals in Architecture.

If you ask the question of Why to a Principle you (should) get a set of Quality Attributes back."

— Ersin Er

A principle is worth stating as an Architectural Principle, if it makes a difference to decisions/behaviors that have strategic impact. We're taking a stand and setting direction, given some area that makes a difference to system outcomes (system identity, key properties and strategic goals).

Guidelines

Each architecture principle should

- clearly state a chosen direction
  - a technical strategy to achieve the strategic intent or architecturally significant system property, address an architectural challenge, or resolve significant risk
  - can we state a principle that will guide subsequent architecture/design decisions to achieve the strategic intent/goal or challenge?
- be simply stated and understandable
- be stated so that you will know if the architecture has the characteristics expressed by the principle
- have a counterargument
- should not be platitudes or general features that are desirable regardless of the system
- be rationalized, stating why the principle is preferred, drawing on business-related factors where possible
- And implications of adopting the principle should be identified
- Be based on experience (if not our own, then that of someone with relevant, trusted expertise and experience)

Source: adapted from Tapscott and Caston

Netflix: System Challenges

Challenges

(Context: 2018)

Scale
Netflix services 130 million subscribers from 190+ countries. The streaming platform processes trillions of events and petabytes worth of data per day to support day to day business needs. This platform is expected to scale out as subscribers continues to grow.

Elasticity
Although majority of the streams have fixed traffic pattern, we have to design the system to prepare for sudden changes (i.e. spikes due to a popular show coming online or unexpected failure scenarios), and be able to adapt and react to them in an automated fashion.

Fit to context and to purpose

Source: https://netflixtechblog.com/keystone-real-time-stream-processing-platform-a3ee651812a

More Properties Driving Netflix Design Decisions

More challenges arising from Netflix system properties (source: Netflix):

2. Diverse Use-cases

Keystone Routing Service: this service is responsible for routing any events to managed sink per user configuration. Each delivery route is realized by an embarrassingly parallel stream processing job. Users may define optional filter and/or projection aggregations. Events are eventually delivered to a storage sink for further batch/stream processing with at-least-once delivery semantics. Users may choose different latency and duplicate tradeoffs.

Stream Processing as a Service: SPaaS platform has only been in production for about a year, yet we have seen tremendous engineering interests, as well as a wide variety of requirements. Below is a summary of some common asks and tradeoffs.

Job State: ranging from complete stateless parallel processing to jobs requiring 10s of TB large local states.

Job Complexity: ranging from embarrassingly parallel jobs with all operators chained together to very complex job DAG with multiple shuffling stages and complex sessionization logic.

Windows/Sessions: window size ranging from within a few second (i.e. to capture transaction start/end event) to hours long custom user behavior session windows.

Traffic pattern: traffic pattern varies significantly depending on each use case. Traffic can be bursty or consistent at GB/sec level.

Failure recovery: some use cases require low failure recovery latency at seconds level, this becomes much more challenging when job both holds large state and involves shuffling.

Backfill & rewind: some jobs require replay of data either from a batch source or rewind from a previous checkpoint.

More at: https://netflixtechblog.com/keystone-real-time-stream-processing-platform-a3ee651812a

Netflix: Architecture Principle

Guiding policy

Principle: **Failure as a First Class Citizen**

Description: Failure is a norm in any large scale distributed system, especially in the cloud environment. Any properly designed cloud-native system should treat failures as a first class citizen.

Rationale: Assume unreliable network

Implications:
- Trust underlying runtime infrastructure, but design automatic healing capabilities
- Enforce job level isolation for multi-tenants support
- Reduce blast radius when failure arise
- Design for automatic reconciliation if any components drifts from desired state or even if disaster failure occurs
- Handle & propagate back pressure correctly

Source: https://netflixtechblog.com/keystone-real-time-stream-processing-platform-a3ee651812a

Netflix Principles

The Failure as a First Class Citizen Principle has more implications than listed there. What else comes to mind (especially when we view the system as a sociotechnical system)?

CounterArgument

The Netflix team doesn't state a counterargument, but often the counter is the usual or de facto way things are done. In this case, counterposition could be to instead focus on preventing failures, rather than assuming that failures will happen and investing in our responses to them.

Principle Examples: Market Facing

Principle Name	Preserve the Dignity of our Customers
Description	Ensure that customers do not experience shame in food insecurity and that they feel supported
Rationale / Benefits	Builds community? Broadens the pool of users. Creates safety Encourages use because there is no negative connotation
Implications	May need to manage perceptions of 'handouts' or those who would take advantage. Ensure that privacy and security, and data privacy is addressed.
Counter argument	Put more energy into encouraging safe participation than into 'policing' participation

From Strategy Masterclass: Ksenia, Webs, Paula, Louise, Barb

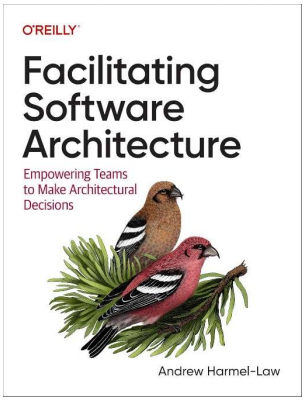
Principle Name	**Prefer accessibility over technology**	**Trust the person in need**
Description	Open the platform the widest user base	We exist to serve people in need and prefer helping over bureaucracy
Rationale / Benefits	Rationale/Benefits: We want to limit the barriers to entry and use of our platform	Rationale/Benefits: Forces us to automate fraud and eligibility
Implications	Counter argument: Accessibility lags	Counter argument: Fraud
Counter argument	Implications:Rely heavily on standardization	

From Strategy Masterclass: Al, André, Burkhard, Hibri, Peter, Ralph

Principle Name	Design for Justice
Description	The products and services we design should be accessible to all and not discriminate
Rationale / Benefits	Ensures equality and does not perpetuate oppression. Supports the entirety of the community. Increases support from all stakeholders
Implications	We will have to vet most of the products and services against some justice criteria to ensure they fit what we have in mind. Need to have people well-versed in ethics who can help judge and assess whether something is "just"
Counter argument	May take more time to release something to market. The definition of "justice" may mean different things to different people. Too much focus for justice may detract from the focus towards equality

From Strategy Masterclass: Erica, Erica, NH, Peter, Sean, Daniel

Principle Examples



Principle: Value independence of teams most highly	**Theme(s): Product Leadership**
Subtitle: Split solutions along team lines	
Rationale: The strength of our approach to building and running our products relies fundamentally on the independence of our teams. The downsides to this are acknowledged, but the upsides are felt to outweigh it, especially when the difficulty of predicting future needs is taken into consideration	
Implications: • Duplication of both function, and data, will inevitably arise. Rather then fight this, we embrace it, acknowledging the need, in certain circumstances, for noticeable eventual consistency and data replication • The combined licencing, runtime and support costs of multiple third-party solutions may be higher than the costs of a single, shared, cross-product-team solution • Solutions can be designed for the needs of the team which owns and runs them. They need not concern themselves the needs of other teams • Both systems and the third-party services / solutions they are build on will tend to be smaller, and more specific-task-focussed • Teams who go their own way need to self-support any third-party services / solutions which they adopt independently	
Background: This principle was adopted at the Architectural Principles Workshop held on the [DATE]. At the time of adoption, the following concerns were raised: • [DRAWBACK/CONCERN] - raised by [NAME]/[anonymous]	

Guidelines

In Chapter 10 of *Facilitating Software Architecture*, Andrew Harmel-Law offers guidance on a collective workshop approach to creating principles. He positions principles as follows:

"Architectural principles are a means of capturing key agreements regarding how you collectively intend to design your software systems. They are a practical embodiment of your organization's tech strategy that helps keep everyone's decisions aligned with the organization's overall direction. They are another means to articulate your minimum viable agreement described in the previous chapter.

A set of principles transforms the organization's abstract vision and goals into a form that can be used to direct software development and make concrete decisions. Targeted and explicit architectural principles are an incredibly powerful tool, but they are (in my experience at least) rarely conceived and deployed effectively.

Failure to take advantage of Architectural Principles is problematic in any approach, but in a decentralized feedback-centric world like ours having effective principles becomes essential. This is because they are superb at aligning decisions effectively without the need for imposing any form of hierarchical or outside control. Poorly conceived principles don't just fail to offer this collective commitment to alignment, they actually get in the way, undermining the ability of the advice process to build trust, enable learning, and deliver high-quality decentralized decisions at a pace that factors in feedback.

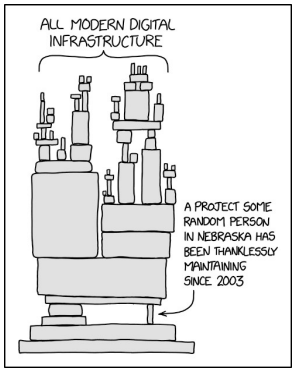
The best way to capture architectural principles and ensure that they align with your organization's goals and vision is to source them from a broad range of people including all teams."

> *"Architectural principles are a means to state your collectively agreed commitments regarding how you will design your systems."*
> *— Andrew Harmel-Law*

Principles: *Show me!*
- Off-the-shelf
- Some classics
- Recent
- Try it: dependency isolation
- Try it: resilience
- Try it!

Principles: Dependencies

If we have identified dependency management as a key strategic challenge (see our Wardley Map), what principles might we explore?



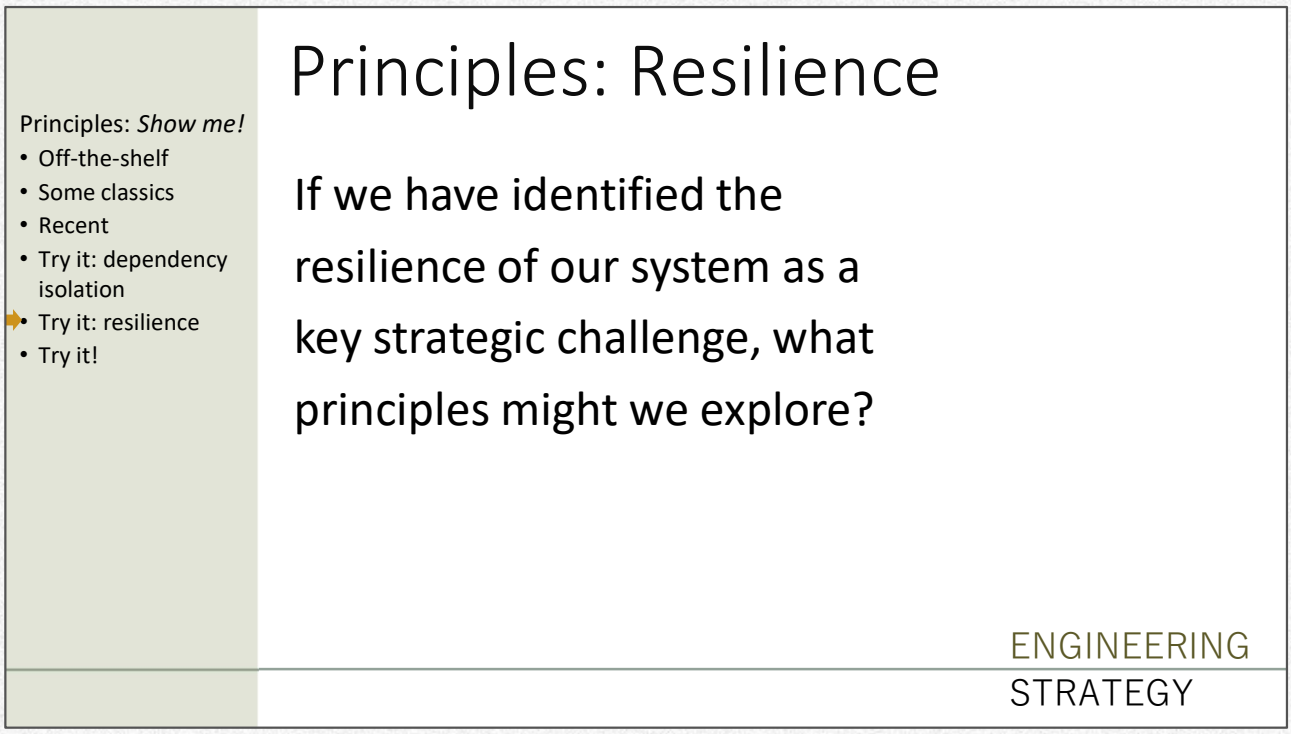
https://xkcd.com/2347/

ENGINEERING STRATEGY

Dependency Management Principles

- Tie your horse to a post, not another horse (The Stable Dependencies Principle: modules may not depend on less stable ones)
- Isolate dependencies. Which dependencies? Identify supply chain risks.

Source: The Stable Dependencies Principle How stability can reduce the cost of change, Christiaan Fouché https://chrisfouche.com/the-stable-dependencies-principle



Resilience Principles

Maintain diversity and redundancy

Manage connectivity

Foster adaptive capacity

Encourage learning

Cluster of Resilience Principles

Looking back at, for example, Ostrom's Principles for Managing a Commons, we might note that small clusters of well-chosen principles can (and need to) work together to provide direction towards the outcome we seek. What resilience principles might work together, to impact the system design, and our approach to system design (including code), operations and engineering practice, and engineering management?

Principles: *Show me!*
- Off-the-shelf
- Some classics
- Recent
- Try it: dependency isolation
- Try it: resilience
- Try it!

Principles: Team Exercise

1. list 4 or 5 challenges that we will face building this system (consider past experience, look at desired qualities indicated on stakeholder profiles, challenges on context map, …)
2. list 4 or 5 ideas for principles (that will help us address challenges)
3. pick one of the ideas for principles, and express the principle (on template)

ENGINEERING
STRATEGY

Principles Exercise

Reminder: it can be helpful to work individually for the first few minutes, then share ideas, then work together as a team on a principle.

Step Back and Reflect

Reflect and share insights and experiences creating and using principles. What's been useful? Hard? What helped?

Attribution

The format for these notes is adapted from a template from Nancy Duarte and team.

For more:

https://www.duarte.com/slidedocs/

Duarte Slidedocs®

We recommend the Duarte material on slidedocs® in addition to the template; much that is valuable there.

Quotes and Photos

We have consciously brought various pioneers and contemporaries visibly into our materials for two reasons:

i. to acknowledge and celebrate the extent to which we are because of others (Abeba Birhane). It is a small way to bring into the room, so to speak, with us people whose insights and work has influenced us, and integrated with our experiences, other reading and conversations, and more, to build what we understand and can share.

ii. to recommend to you wonderful work you may want follow up on, and also to draw in our contemporaries who are sharing insights that you too may find useful, and want to follow them on twitter, etc.

"Act always so as to increase the number of choices.'
— Heinz von Foerster

Stay in Touch

Ruth Malan: find me on LinkedIn, Bluesky and Mastodon

Web: ruthmalan.com

Masterclasses and Workshops

- **System Design and Architecture,** May 12-14 and 19-21, 2025
- **Technical Leadership,** May 8 and 15, 2025

"What we care about is the productive life, and the first test of the productive power of the collective life is its nourishment of the individual. The second test is whether the contributions of individuals can be fruitfully united"
— Mary Parker Follett

Attribution — Please give appropriate credit if you quote from this book. You may do so in any reasonable manner, to a reasonable extent, respecting the work it takes to create something like this.