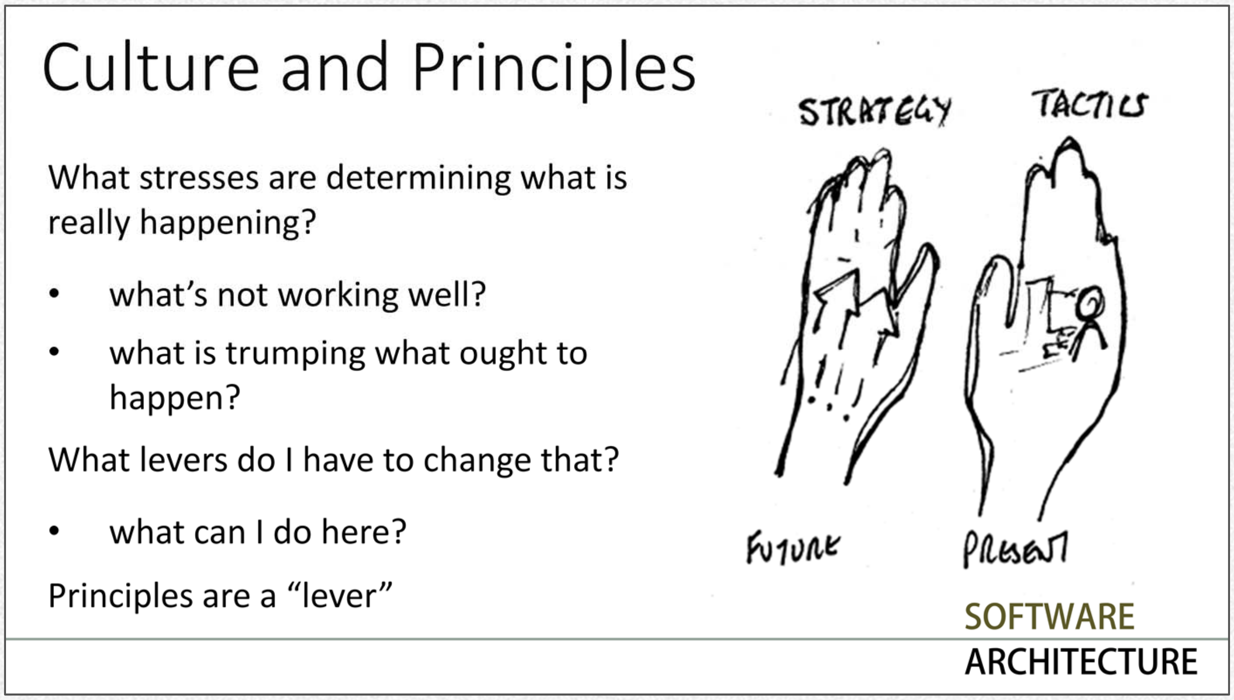
Architecture Principles

By Ruth Malan
Bredemeyer Consulting

The Role of Critique
Posit Structure
Design It! From Programmer to Software Architect
Michael Keeling
Thinking in Systems
Donella H. Meadows
System Integrity
"A system is an interconnected set of elements that is coherently organized in a way that achieves something"
– Donella Meadows
Coherence and Purpose

2021

Principles shape, ruling out what we don't want, and guiding towards what we do



Culture and Principles

What stresses are determining what is really happening?

- what's not working well?
- what is trumping what ought to happen?

What levers do I have to change that?

- what can I do here?

Principles are a "lever"

SOFTWARE
ARCHITECTURE

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.

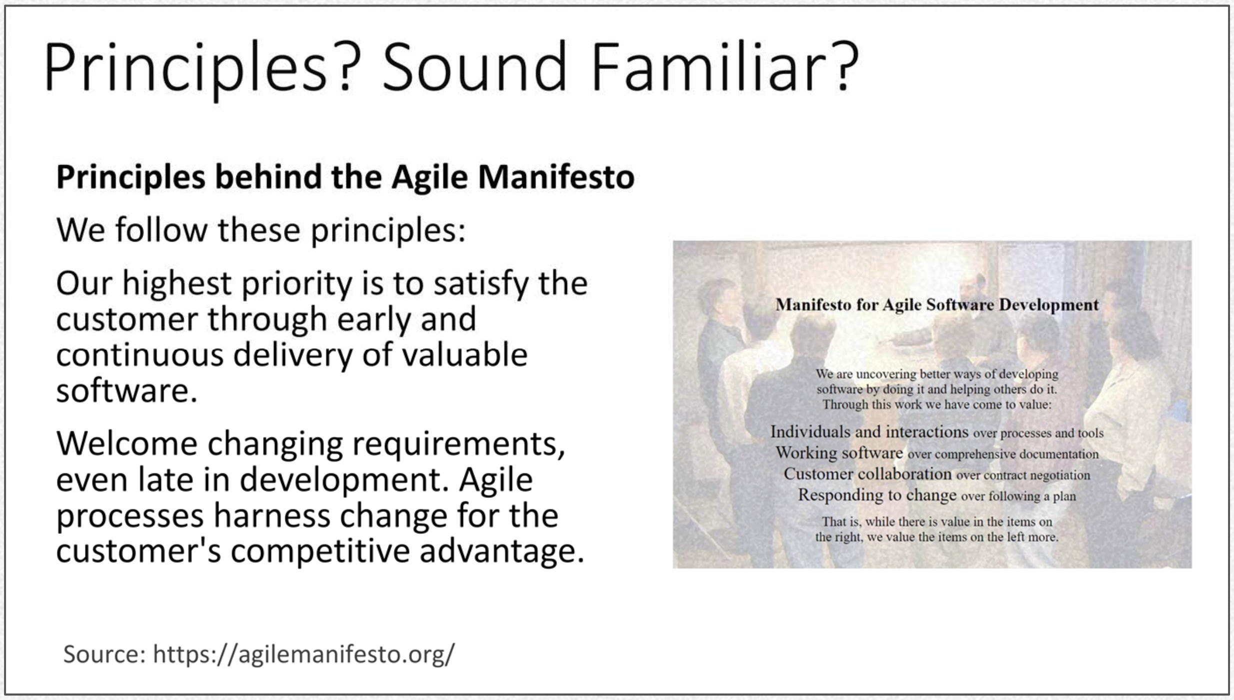
Every principle we add to the architecture, consumes cycles – creating and evolving the principle statement, understanding and following the principle, and governing the principle (to catch misses and help ensure architectural integrity) all costs attention.

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.

What are Principles? It's worth revisiting the Principles in the Agile Manifesto, and considering their role and influence



Principles? Sound Familiar?

Principles behind the Agile Manifesto

We follow these principles:

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.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

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.

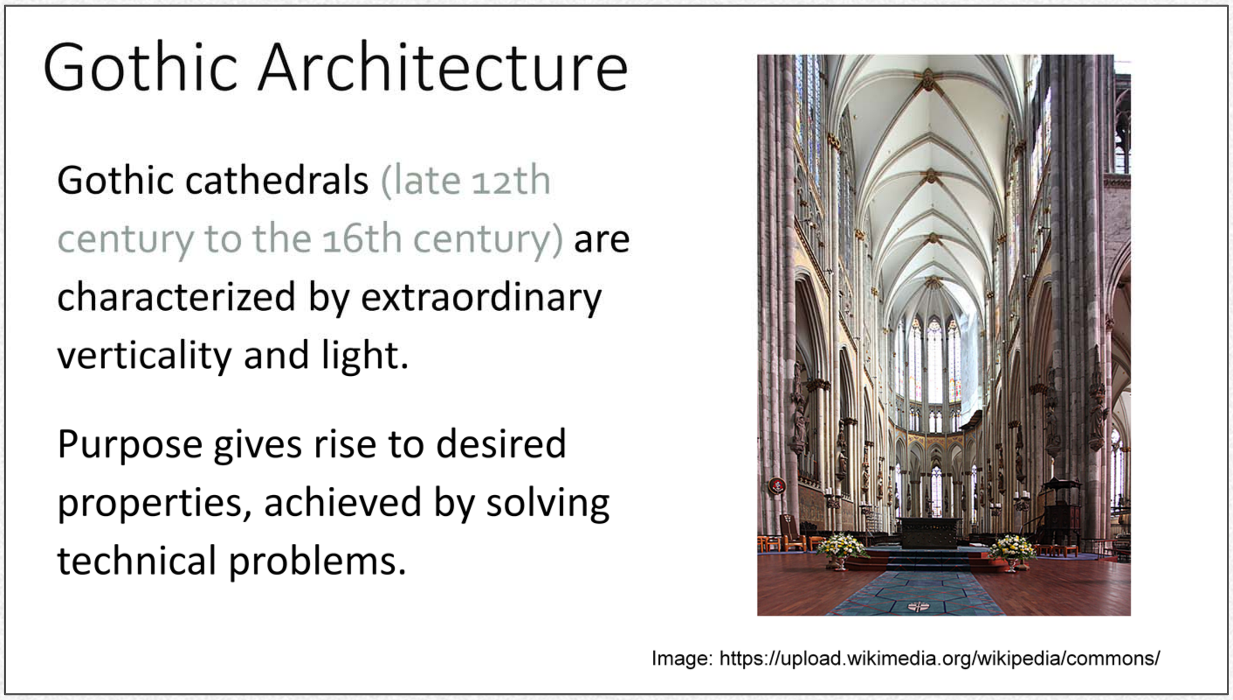
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Source: https://agilemanifesto.org/

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

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

Building architecture? Literally?? Seriously??
Well, what can we learn?



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

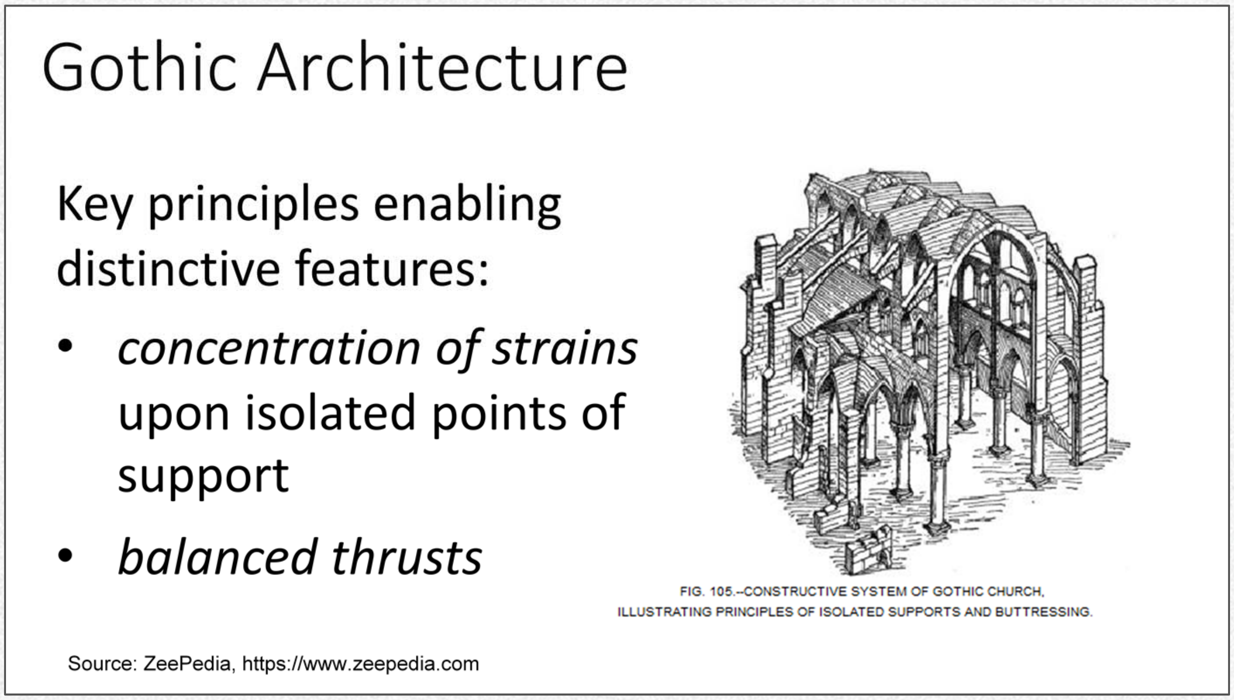
This description of Gothic architecture is interesting, in that, by working within a small set of structural principles, the various builders and artisans gained considerable freedom to innovate:

"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

Gothic architecture has distinctive features, enabled by defining structural principles



Gothic Architecture

Key principles enabling distinctive features:

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

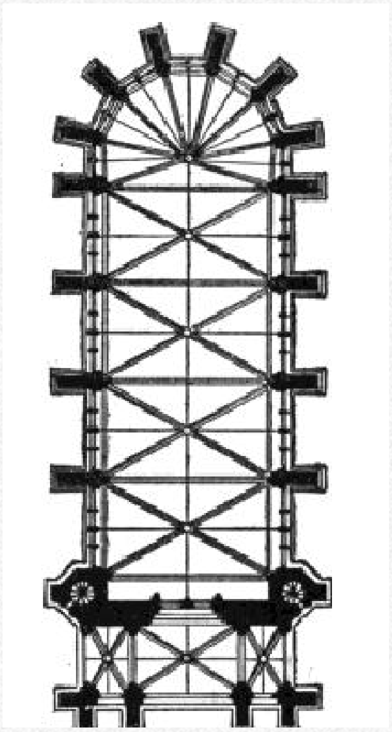
FIG. 105.--CONSTRUCTIVE SYSTEM OF GOTHIC CHURCH, ILLUSTRATING PRINCIPLES OF ISOLATED SUPPORTS AND BUTTRESSING.

Source: ZeePedia, https://www.zeepedia.com

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."

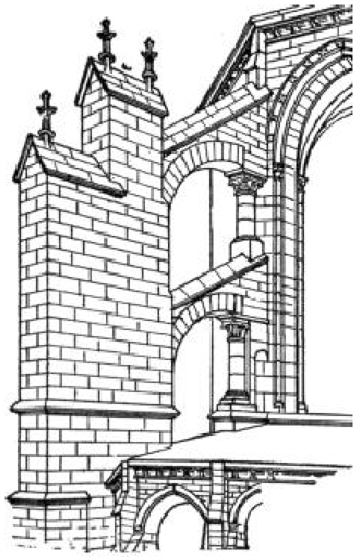
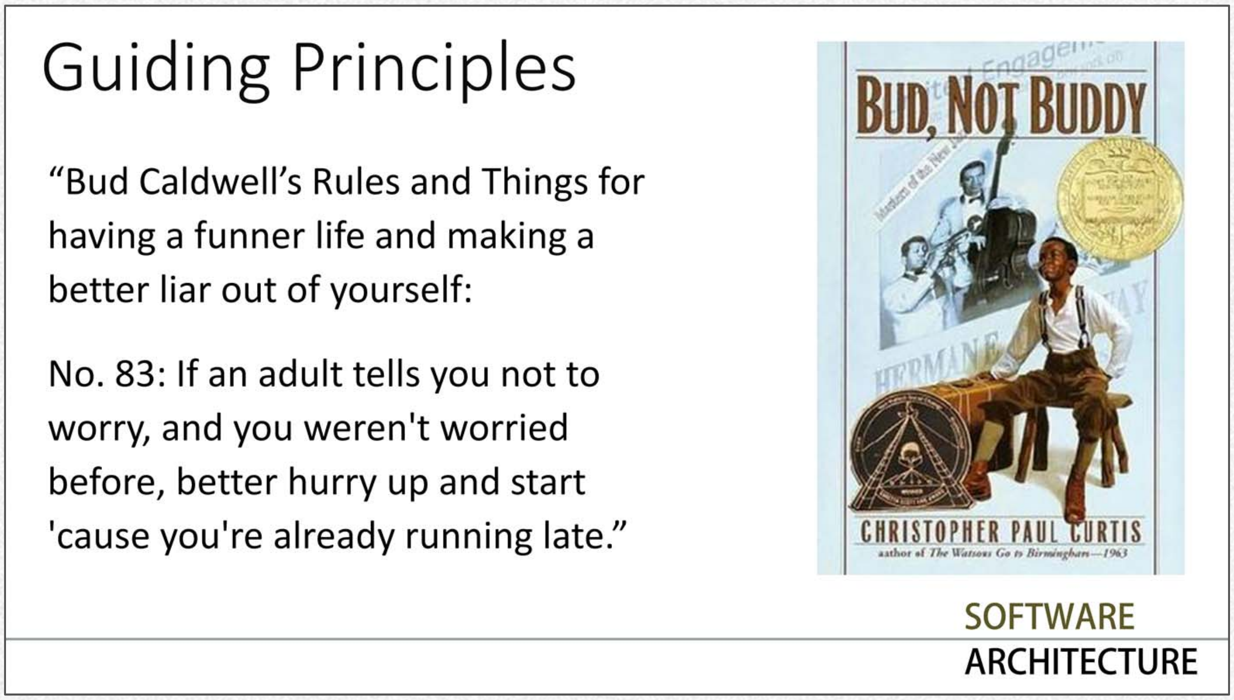


FIG. 107.--EARLY GOTHIC FLYING BUTTRESS.

Relating guidance to contexts, to shape future behavior

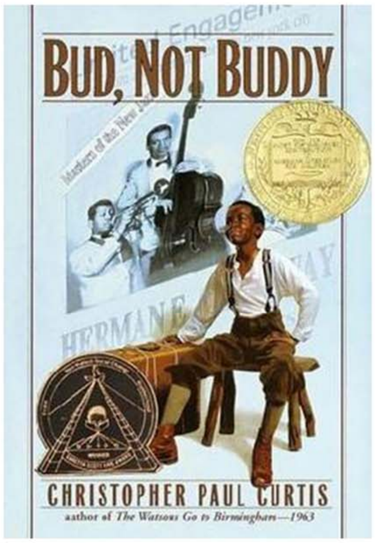


Guiding Principles

"Bud Caldwell's Rules and Things for having a funner life and making a better liar out of yourself:

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."

SOFTWARE
ARCHITECTURE



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:

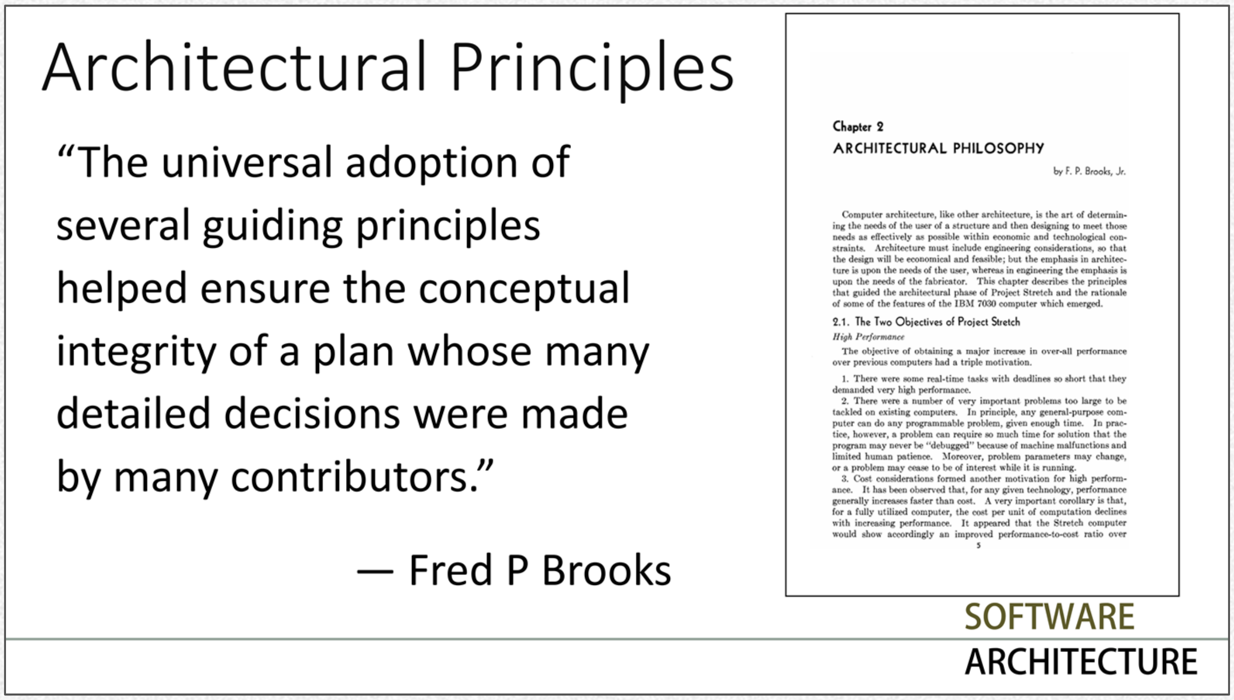
> Clear, precise, and easy to remember.

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??



Architectural Principles

"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

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

SOFTWARE ARCHITECTURE

"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

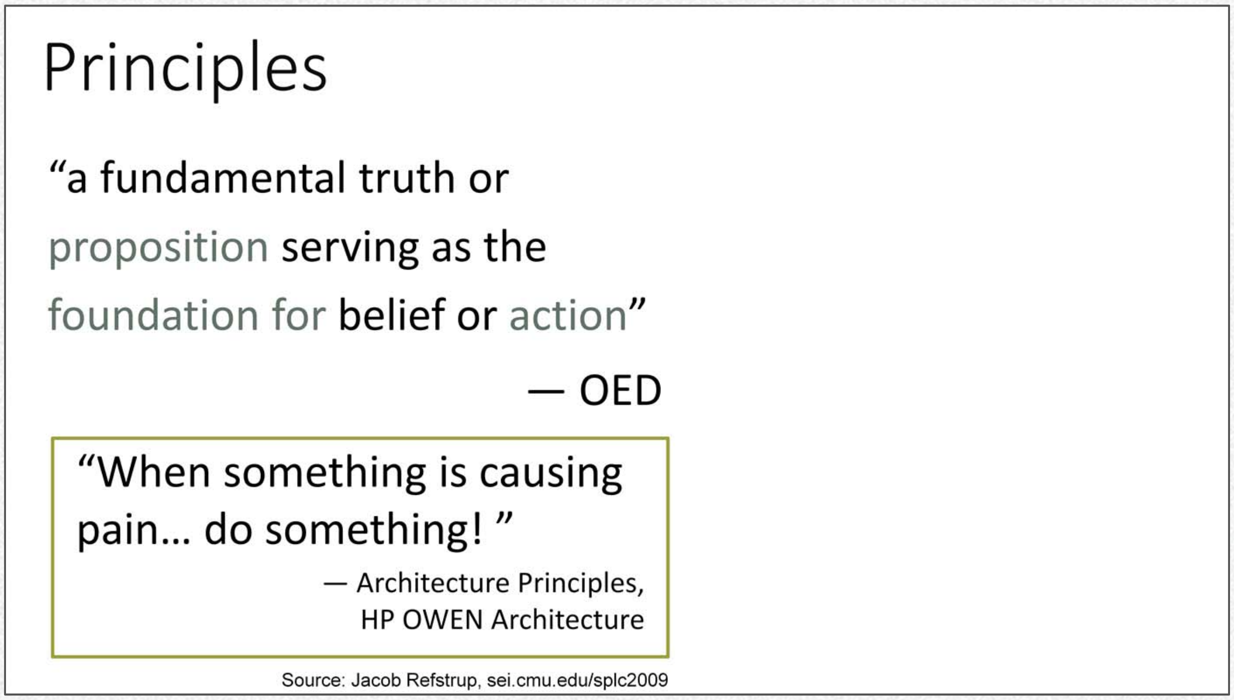
We know Fred Brooks from reading *The Mythical Man Month*. But it is also worth reading Fred Brooks' Architectural Philosophy chapter in Project Stretch, for the history and for the lessons about architectural design. The architectural principles Fred Books describes are:

- 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. "

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



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.

[..] In the book *Architecture Principles*, Danny Greefhorst and Erik Proper created probably the most comprehensive definition:

> a declarative statement that normatively prescribes a property of the design of an artefact, which is necessary to ensure that the artefact meets its essential requirements.

Although this definition is a little abstract, it clarifies the design principle's role as ensuring that some aspect of your architecture meets some aspect of its requirements."

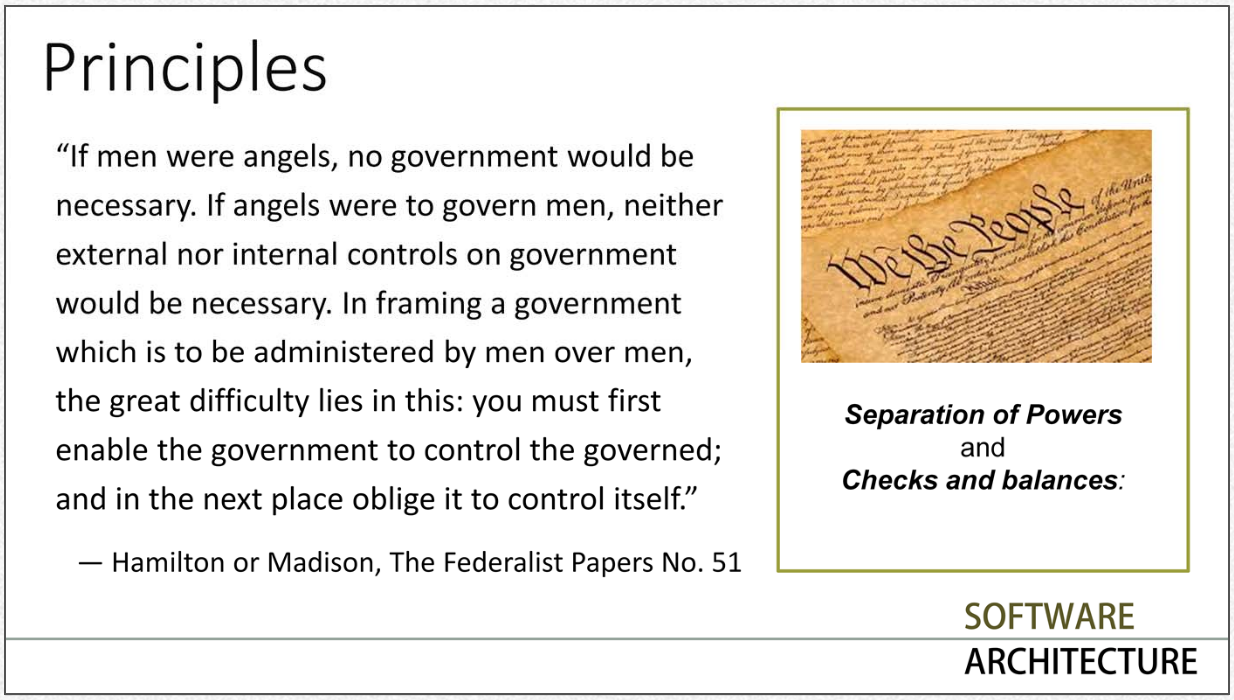
Source:
https://www.infoq.com/articles/architectural-design-principles/

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.

"Every day, each of us is faced with a blizzard of situations we must respond to. Without principles we would be forced to react to all the things life throws at us individually, as if we were experiencing each of them for the first time. If instead we classify these situations into types and have good principles for dealing with them, we will make better decisions more quickly"

— Ray Dalio (via Ersin Er)

Referring back to the Constitution: its underlying principles shaped the structure of government



Principles

"If men were angels, no government would be necessary. If angels were to govern men, neither external nor internal controls on government would be necessary. In framing a government which is to be administered by men over men, the great difficulty lies in this: you must first enable the government to control the governed; and in the next place oblige it to control itself."

— Hamilton or Madison, The Federalist Papers No. 51

Separation of Powers
and
***Checks and balances**:*

SOFTWARE
ARCHITECTURE

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

The US Constitution

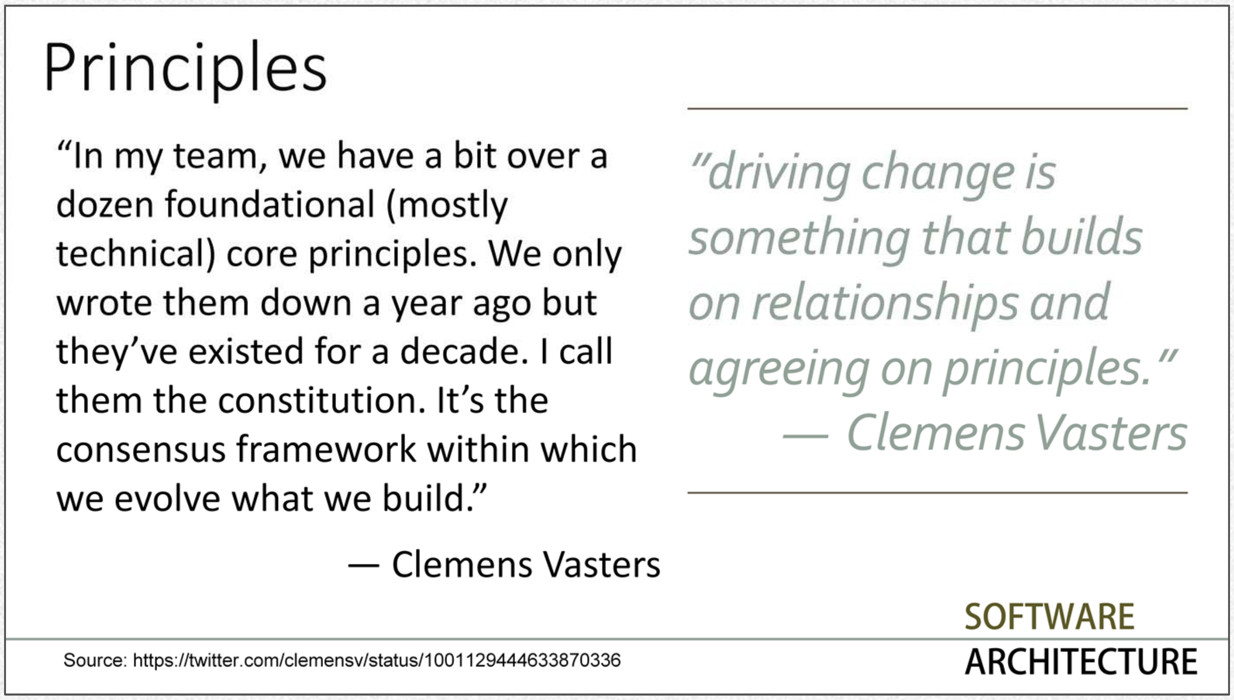
The US Constitution was guided by two principles: *separation of powers* and *checks and balances*. "Separation of powers is a model that divides the government into separate branches, each of which has separate and independent powers. [..] Article 1 of the United States Constitution establishes the Legislative Branch [..] Article 2 [..] establishes the Executive Branch [..] Article 3 [..] establishes the Judicial Branch, which consists of the United States Supreme Court. [..] The Checks and Balances system provides each branch of government with individual powers to check the other branches and prevent any one branch from becoming too powerful." Legal Informatics Institute, Cornell Law School

"TO WHAT expedient, then, shall we finally resort, for maintaining in practice the necessary partition of power among the several departments, as laid down in the Constitution? The only answer that can be given is, that as all these exterior provisions are found to be inadequate, the defect must be supplied, by so contriving the interior structure of the government as that its several constituent parts may, by their mutual relations, be the means of keeping each other in their proper places. Without presuming to undertake a full development of this important idea, I will hazard a few general observations, which may perhaps place it in a clearer light, and enable us to form a more correct judgment of the principles and structure of the government planned by the convention."

— Hamilton or Madison, *The Federalist Papers* No. 51

Source:
https://www.law.cornell.edu/wex/separation_of_powers_0

https://avalon.law.yale.edu/18th_century/fed51.asp



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

"driving change is something that builds on relationships and agreeing on principles."
— Clemens Vasters

SOFTWARE ARCHITECTURE

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

"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 (@clemensv) is the Principle Architect of Azure's eventing and messaging services. He has 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:

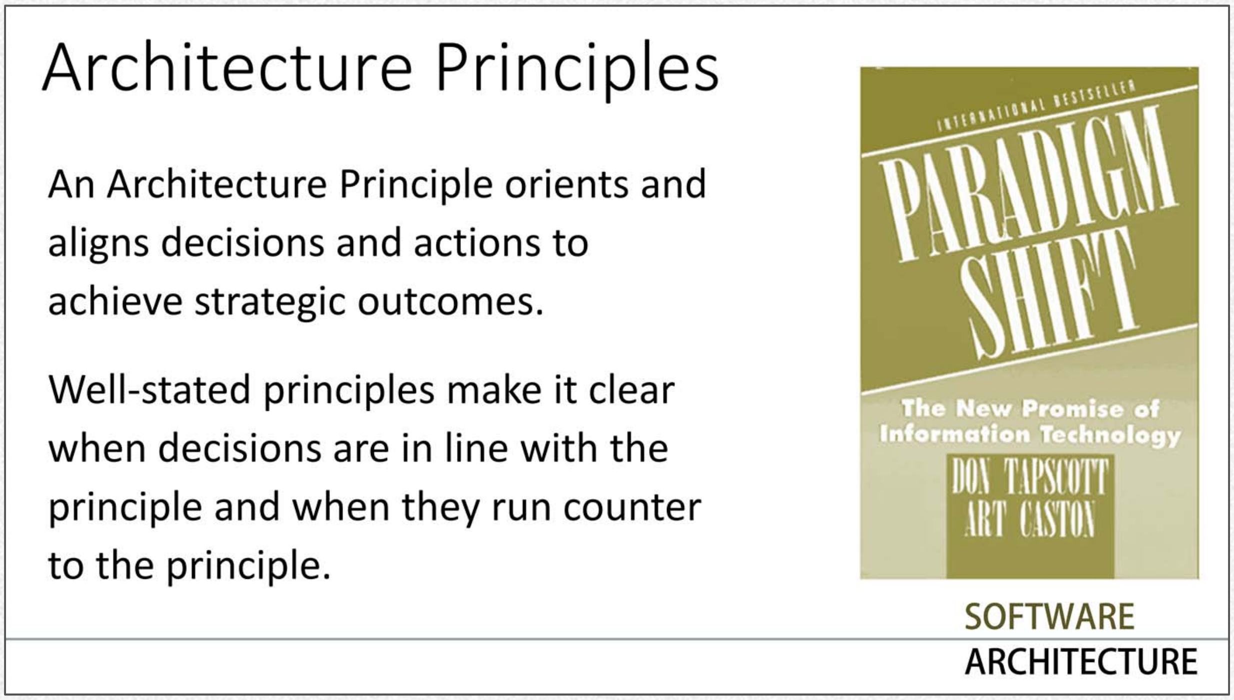
1: The services implement *open-standards based protocols* as primary form of communication. If a standards-based alternative emerges for a proprietary capability, the proprietary capability is phased out.

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."

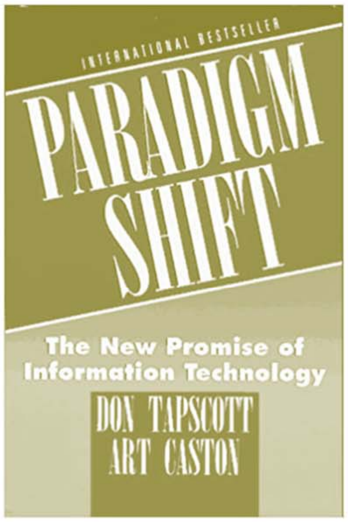
Source: https://twitter.com/clemensv/status/1001133005593858048



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.



SOFTWARE
ARCHITECTURE

"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

principle 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

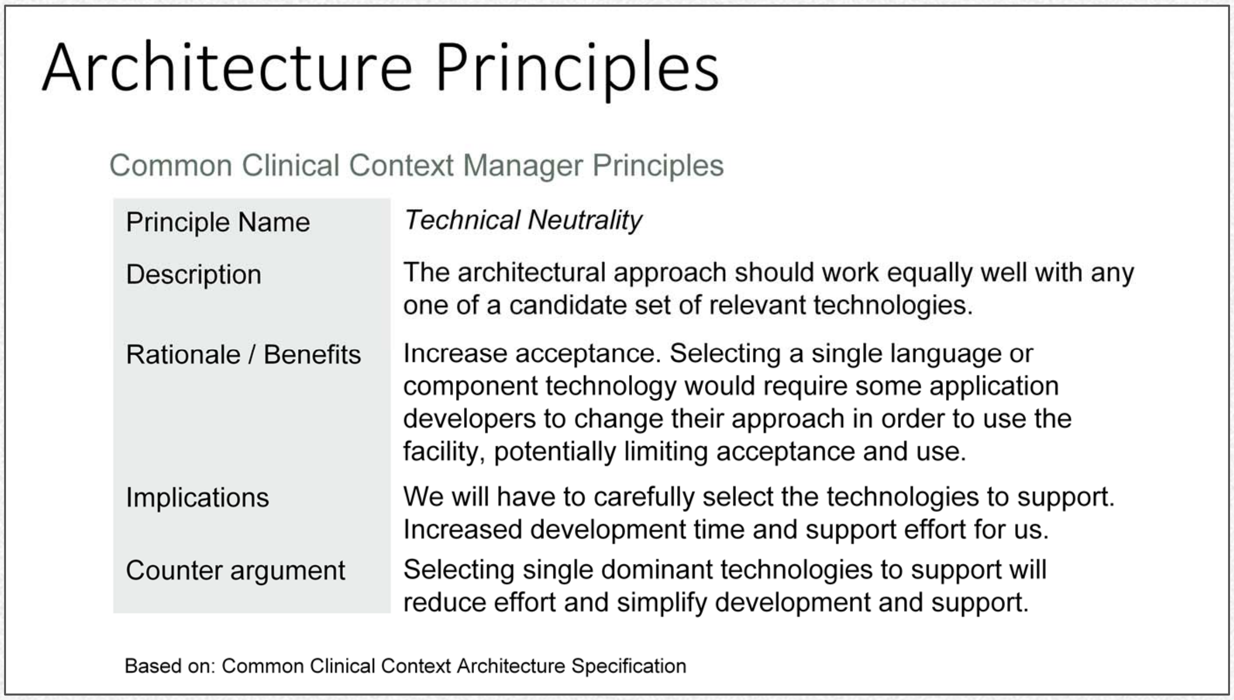
Counter forces (aka counterarguments or alternatives considered): provides a place to say we recognize what factors weigh against the principle and what other approaches we might take (that other people in our environment would argue for) and why we shouldn't do that. One way to think about the counterargument/counter 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."

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

Scope/applicability (optional, as relevant): identify the scope or domain of applicability (so we don't need to think about it and get exceptions where it's not applicable).

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

An example Architecture Principle



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

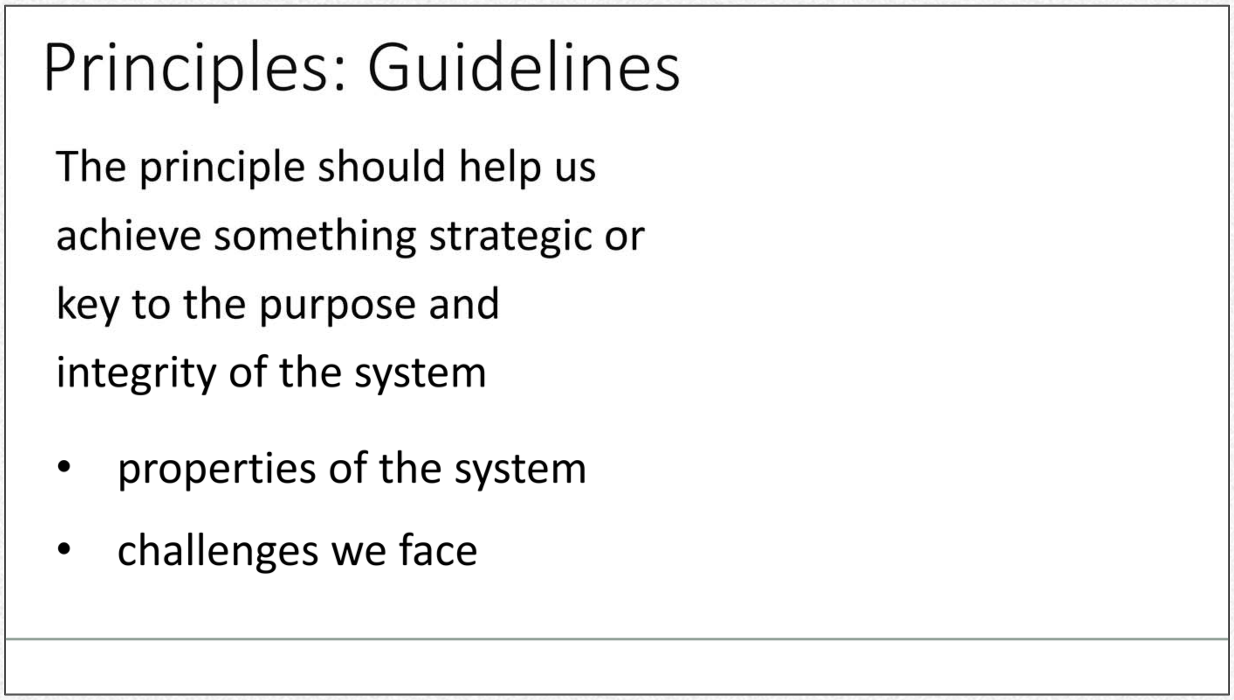
Cautionary note:
"a fad [..] to write down a list of "architectural [principles]" and that's fine, what's weird is how infrequently those align with the org they have, and how often they lead nowhere." — Amy Tobey

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

"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."

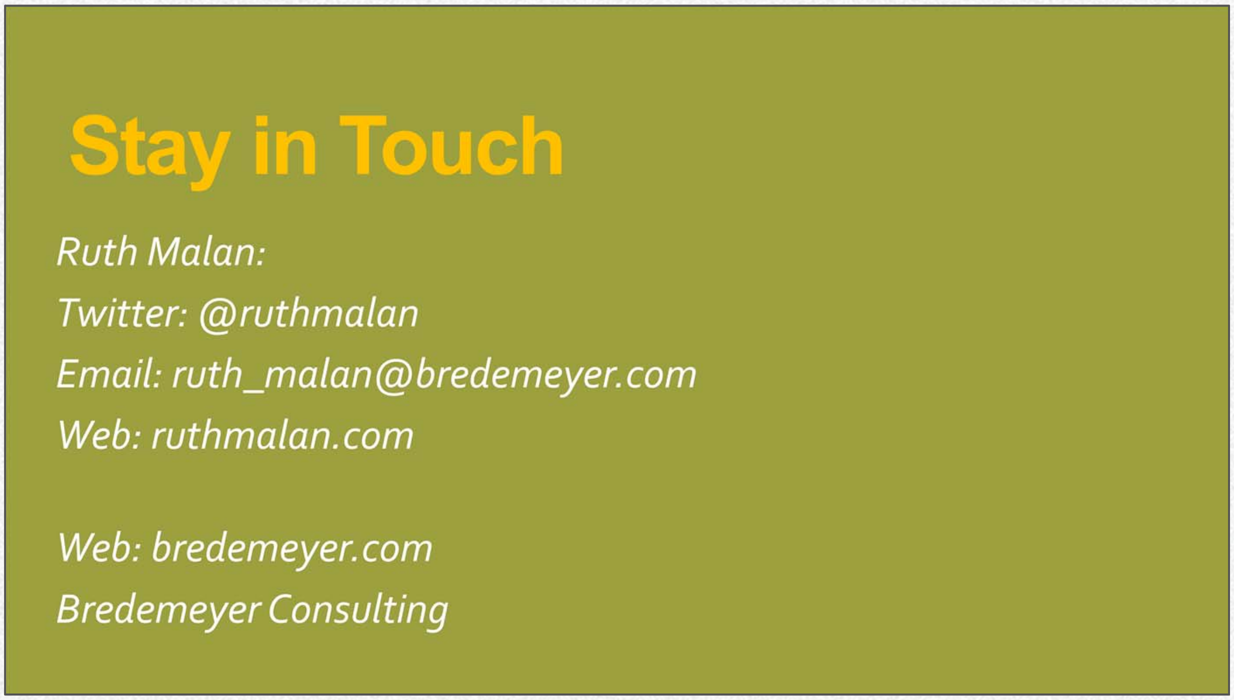
— Ersin Er

Guidelines

Each 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



Stay in Touch

Ruth Malan:

Twitter: @ruthmalan

Email: ruth_malan@bredemeyer.com

Web: ruthmalan.com

Web: bredemeyer.com

Bredemeyer Consulting

This is a module from our system design workshop.