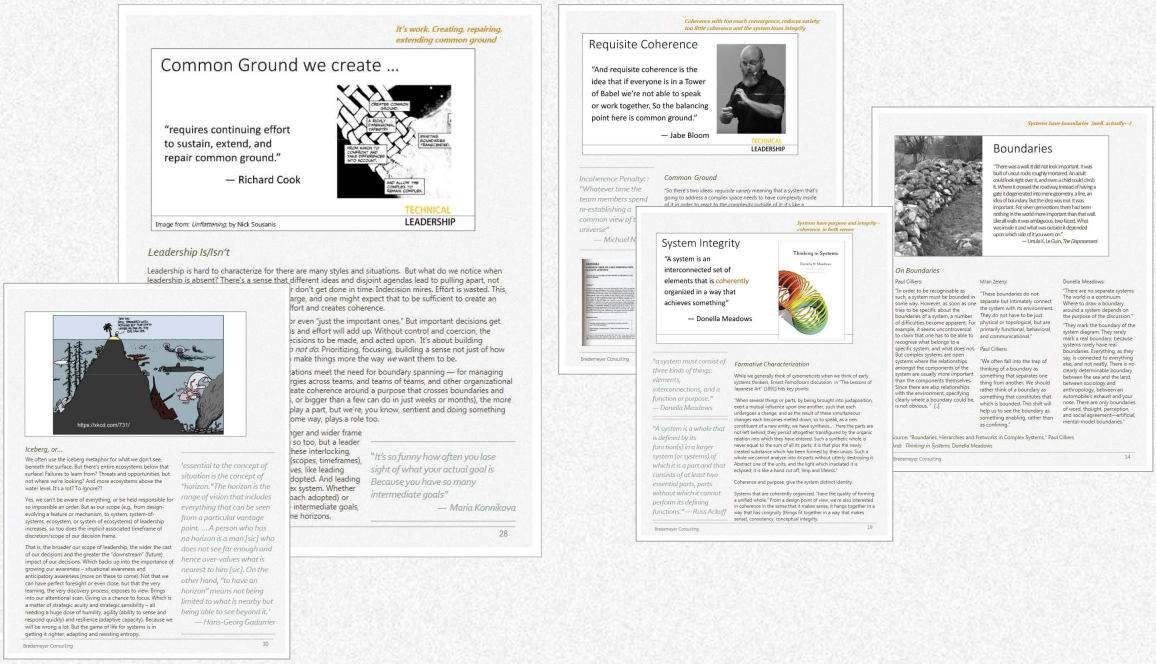
(Socio-) Technical Decisions

Ruth Malan



September 2026



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

(Socio-) Technical Decisions

Leadership and decision making

Decisions constrain and enable

When do we make them?

Forces and Tradeoffs

Mental Models

Context

How do we make decisions?

Significance

Experiments and Feedback

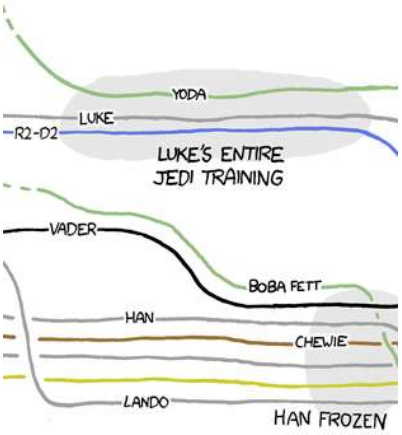
Communication

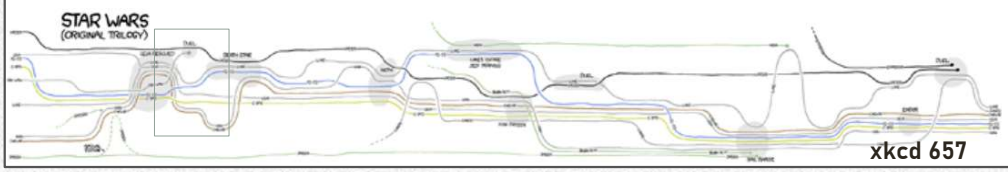
DECISIONS

Decisions
- Constrain, and enable

Making Decisions
- How we make decisions
- Context and tradeoffs

Socializing Decisions





xkcd 657

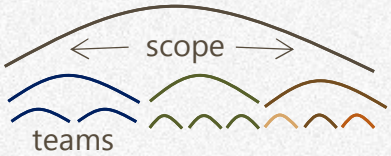
Decisions Across Boundaries

Recall: Leadership plays a role when we're trying to do something bigger than what we can accomplish alone.

We can't have everyone making every decision; it's not feasible or practical or desirable. And, typically, we value moving decision making to those who have necessary perspective and insight into options and impact. This also means that some decisions — those that need perspective across boundaries — need to be made at broader scope.

As scope of decision impact increases, what is riding on them is all the more critical.

The demands on expertise and experience, increase with impact. Further, as the scope or span of impact increases (that is, the decision has impact across team and other organizational or system boundaries), so too do the organizational challenges. Decisions with organizational (strategy, team, etc.) impact, need to be made with wisdom (understand impact of choices), strategic acuity, and keen organizational sensibility. Those with technical impact, require technical wisdom.



These "across scope" decisions set context for further decisions. They enable something strategic, but also constrain and shape — but only as essential to system outcomes. And decisions that cross contexts or boundaries, need leadership — to bring perspectives and expertise together in making the decision, and to help others understand the need and outcomes, and consequences, and what their role is in making the decision effective. Leaders communicate strategic intent and decisions, and foster organizational will and goodwill, to facilitate work towards coherent strategic outcomes.

Decisions Constrain

‘Limiting or closing off alternatives is the most common understanding of the term “constraint.”’

— Alicia Juarrero



Image from video from LeanUX 2015

Constraints are limitations we need to be aware of. They restrict choices open to us.

"The notion of a constraint is not a negative one. It's not something which merely limits possibilities, constraints are also enabling."

— Paul Cilliers

Decisions Reduce the Options Space

Decisions constrain—they eliminate options. Alicia Juarrero observes that this is what we commonly mean by constraint—this limiting or closing off of alternatives; this altering of the probability distribution of available alternatives. But! In so doing, Alicia notes, they make the system "diverge from chance, from randomness."

Illustration of Constraints that Limit

"The connection of the tibia and the peronei to the knee joint constrains the movement of the lower leg in such a way that it makes no sense to examine the tibia's physiology, for example, independently of the knee. The tibia's connection to the knee gives the former characteristics which it wouldn't have otherwise: it can move in some ways but not others. The constraints which the connections subject the lower leg to reduce the number of ways in which the leg can move: it can bend backwards but not forwards, for example. In this example a constraint is a reduction of the leg's state space. This is the most common understanding of the term "constraint" . "

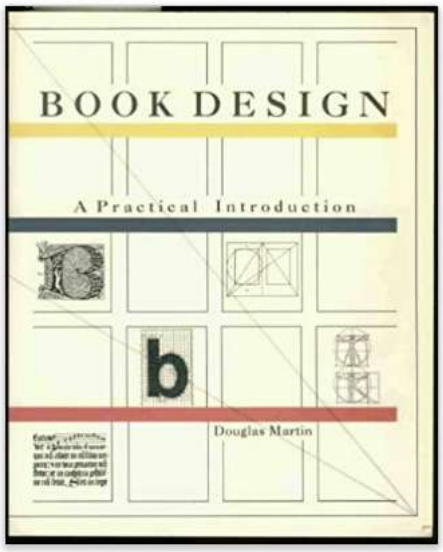
— Alicia Juarrero, "Causality as Constraint"

Not Make Decisions?

"Questions about whether design is necessary or affordable are quite beside the point: design is inevitable.

The alternative to good design is bad design, not no design at all."

— Douglas Martin



Decisions will be made — Implicitly or Explicitly; Intentionally or Accidentally,

The question is not "do we have a strategy?" or "does the organization, product or system have an architecture?" What we have is more or less intentional, more or less emergent, and more or less accidental. If we're not making big decisions (intentionally), we're allowing a myriad small decisions, some implicit and not reflectively weighed and checked, to add up, to determine strategy or architecture. So the question is not do we have a strategy or design. But rather "how good is it?" Can it be better? How so?

For example, if we want agility, we need to design and guide evolution for agility, for change and for responsiveness. We need to do this for the organization (teams, organizational and team dynamics, ..) and for the systems (architecture and design) and for the development, deployment and operations environment.

While the slide uses the Martin quote, we might acknowledge that our situations aren't that extreme. We are doing design (often more locally to the team's realm of focus, and in the medium of code), and teams work on improving the design. It's more a matter of anticipatory design and responsive design at the system level, so system outcomes versus locally maximizing due to local scope of attention and focus of work (objectives and pressures and realm of attention).

* Quote source: *(Almost) Every infrastructure decision I endorse or regret after 4 years running infrastructure at a startup, 2/1/2024*
https://cep.dev/posts/every-infrastructure-decision-i-endorse-or-regret-after-4-years-running-infrastructure-at-a-startup/

"Every software-intensive system has an architecture. In some cases that architecture is intentional, while in others it is accidental. Most of the time it is both"

— Grady Booch

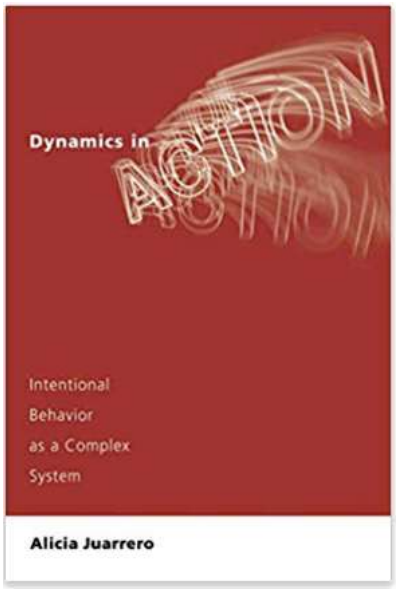
*" Like most tech debt, we didn't **make** this decision, we just did not **not** make this decision."*

*— Jack Lindamood**

Constraints Restrict, But

"But if all constraints restricted a thing's degrees of freedom in this way, organisms (whether phylogenetically or developmentally) would progressively do less and less."

— Alicia Juarrero



While True, ...

Constraints close off avenues, restrict the degrees of freedom, but if this was all they did, systems, including organisms, would just do less and less, as they became more constrained (Alicia Juarrero).

From Alicia Juarrero's talk (Deliberate Complexity Conference):

Constraints are conditions or factors that raise or lower barriers to energy, matter, and information flow – without themselves directly transferring energy. Example: an organisms vasculature does not impart energy directly; it channels and organizes energy flow.

Context dependent constraints enable complexity: some constraints link separate and independent elements and processes such that they become conditional on one another. They become inherently context-dependent. Enabling constraints facilitate the weaving together of interdependencies (among parts, and between parts and context). Examples: synchrony, entrainment, alignment. Enabling constraints self-organize interdependent, coherent, coordination dynamics (to create/enable new coherent dynamics). As a result, a complex system is embedded (not just plunked) in a context (temporal as well as spatial).

Source: Video of Alicia Juarrero's talk at the Deliberate Complexity online conference in 2022: Complexity is not Complication, https://www.youtube.com/watch?v=WmtjQZCIsqY

Also recommended: Constraints that Enable Innovation - Alicia Juarrero https://vimeo.com/128934608

"Think of constraints not just as a restrictions, but as changes in probability of what's going on, changes in the likelihood of something"

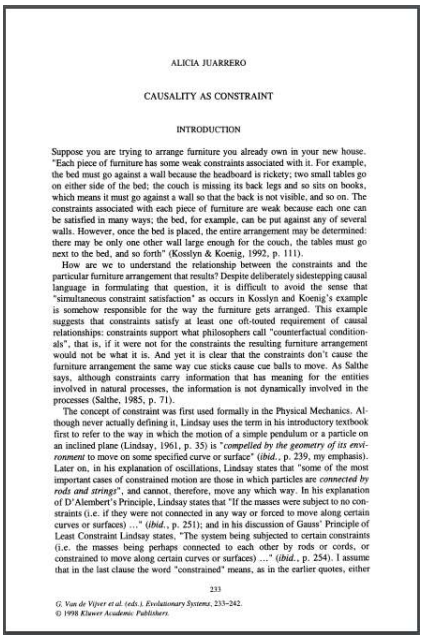
— Alicia Juarrero

Constraints reduce some options, and create others

Constraints Enable

"constraints not only reduce the alternatives — they also create alternatives. Constraints, that is, can also create properties which a component exhibits in virtue of its embeddedness in a system, properties it would otherwise not have."

— Alicia Juarrero
"Causality as Contraint"



ALICIA JUARRERO

CAUSALITY AS CONSTRAINT

INTRODUCTION

Suppose you are trying to arrange furniture you already own in your new house. "Each piece of furniture has some weak constraints associated with it. For example, the bed must go against a wall because the headboard is rickety; two small tables go on either side of the bed; the couch is missing its back legs and so sits on books, which means it must go against a wall so that the back is not visible, and so on. The constraints associated with each piece of furniture are weak because each one can be satisfied in many ways; the bed, for example, can be put against any of several walls. However, once the bed is placed, the entire arrangement may be determined: there may be only one other wall large enough for the couch, the tables must go next to the bed, and so forth" (Kosslyn & Koenig, 1992, p. 111).

How are we to understand the relationship between the constraints and the particular furniture arrangement that results? Despite deliberately sidestepping causal language in formulating that question, it is difficult to avoid the sense that "simultaneous constraint satisfaction" as occurs in Kosslyn and Koenig's example is somehow responsible for the way the furniture gets arranged. This example suggests that constraints satisfy at least one oft-touted requirement of causal relationships: constraints support what philosophers call "counterfactual conditionals", that is, if it were not for the constraints the resulting furniture arrangement would not be what it is. And yet it is clear that the constraints don't cause the furniture arrangement the same way cue sticks cause cue balls to move. As Salthe says, although constraints carry information that has meaning for the entities involved in natural processes, the information is not dynamically involved in the processes (Salthe, 1985, p. 71).

The concept of constraint was first used formally in the Physical Mechanics. Although never actually defining it, Lindsay uses the term in his introductory textbook first to refer to the way in which the motion of a simple pendulum or a particle on an inclined plane (Lindsay, 1961, p. 35) is "*compelled by the geometry of its environment* to move on some specified curve or surface" (*ibid.*, p. 239, my emphasis). Later on, in his explanation of oscillations, Lindsay states that "some of the most important cases of constrained motion are those in which particles are *connected by rods and strings*", and cannot, therefore, move any which way. In his explanation of D'Alembert's Principle, Lindsay states that "If the masses were subject to no constraints (i.e. if they were not connected in any way or forced to move along certain curves or surfaces) ..." (*ibid.*, p. 251); and in his discussion of Gauss' Principle of Least Constraint Lindsay states, "The system being subjected to certain constraints (i.e. the masses being perhaps connected to each other by rods or cords, or constrained to move along certain curves or surfaces) ..." (*ibid.*, p. 254). I assume that in the last clause the word "constrained" means, as in the earlier quotes, either

233

G. Van de Vijver et al. (eds.), Evolutionary Systems, 233–242.
© 1998 *Kluwer Academic Publishers.*

Constraints Create Alternatives

"Constraints not only reduce alternatives—they also create alternatives." If we take (Alicia Juarrero's example of) language, the constraints of syntax allow meaning to emerge.

Wholes arise from Constraints, and Wholes give rise to Constraints

"parts interact to produce novel, emergent wholes; in turn, these distributed wholes as wholes regulate and constrain the parts that make them up"

— Alicia Juarrero, "Dynamics in Action: Intentional Behavior as a Complex System"

Juarrero (1999) distinguishes governing from enabling constraints: governing constraints regulate and restrict, while enabling constraints make a new level of complexity possible.

Context-sensitive constraints [..] synchronize and correlate previously independent parts into a systemic whole

— Alicia Juarrero

By curtailing the potential variation of component behavior, [..] context-dependent constraints paradoxically also create new freedoms for the overall system.

— Alicia Juarrero

We need to make decisions. But when⋯?

Decisions Constrain and Enable

"In *Context Changes Everything*, Juarrero shows that coherence is induced by enabling constraints [..] and that the resulting coherence is then maintained by constitutive constraints."

— MIT Press

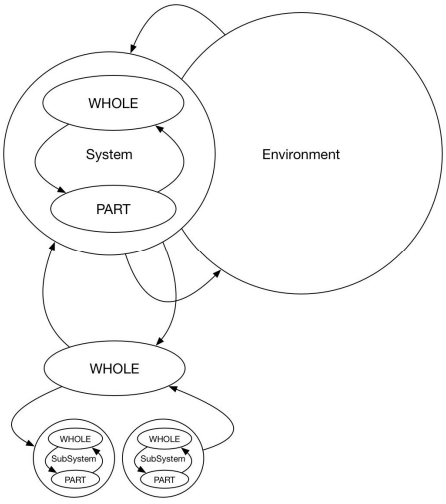


Image source: Jabe Bloom (on twitter)

Systems – but make it Wicked!

The systems we are designing interact within environments, that act back on the system. As the system begins to emerge, it also starts to act back on itself, placing constraints on its elements, to enable connections and flows, and so on. This "placing constraints on" may be more intentional and considered, or more accidental. The sociotechnical system is also placing constraints on itself, to foster coherence. Protocols, standards, decisions and other agreements. Creating common ground by collaboratively modeling, so that a shared language and shared understandings emerge.

Decisions

Decisions, making choices and constraining the subsequent design space, is both inevitable and necessary. Decisions about modularity and coupling, decisions about mechanisms to support capabilities, decisions about technology we will integrate and depend on (within the system, or our development and operations environments), all contribute to our ability to create and evolve a system that is sufficiently stable to exist, yet dynamic and evolving.

"The causal mechanism at work between levels of hierarchical organization can best be understood as the operations of constraint"

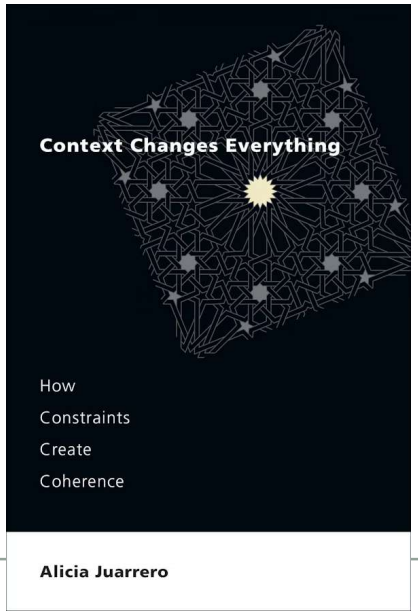
— Alicia Juarrero

"coherence is induced by enabling constraints, not forceful causes, and that the resulting coherence is then maintained by constitutive constraints. Constitutive constraints, in turn, become governing constraints that regulate and modulate the way coherent entities behave."

Context Changes Everything

"Constraints sculpt a rugged and multidimensional landscape, the possibility and probability space of what can happen at all, what will most likely happen, and, when it cannot, can or must happen."

— Alicia Juarrero, Context Changes Everything, 2023



Building our Understanding of the Context

~100 years ago:

"Take a situation made by credit conditions, customers' demand, output facilities, and workers' attitude. They all together make a certain situation, but they constitute that situation through their relation to one another. If you change one, usually some, if not all, of the others are changed"

— Mary Parker Follett, The Illusion of Final Authority

Recently:

"lifted from the context in which they are naturally embedded, some events not only would not occur; they could not occur. Others would change dramatically. The keyword here is embedded [..] In various guises, context dependence is widespread. Nevertheless, settings, circumstances, conditions, and context in general continued to be ignored"

— Alicia Juarrero, *Context Changes Everything*, 2023

"Since a system's prior experience constrains its behavior, that history, too, is embodied in its ontogenetic landscape."

— Alicia Juarrero, Dynamics in Action

Context Factors

"good architecture is a response to context, not a template you copy"

— Gregor Hohpe

"Look at this roofline. That steep pitch isn't a style choice — it's physics. In the Alps, snow has to slide off before its weight collapses the structure. The design is the constraint, solved elegantly."

— Will Webster



Image and quote source: Will Webster, on LinkedIn
https://www.linkedin.com/posts/willwebster_gregor-hohpe-has-a-line-i-keep-coming-back-activity-7505787328674545664-cHPh

Context for Context

Will Webster goes on to say:

> "Now the funny part: I keep seeing this exact roof style built on new homes here in Perth. Beautiful timber work, steep alpine pitch — and nowhere for the snow to slide off, because there's no snow. We kept the form and lost the function."

And again, context factors! What in the context is drawing folk to this design or style, even when the original design constraints no longer apply? Fashion and taste? And are those an appeal to nostalgia? A signaling of European roots, for transplants to a new context? But those design choices come with tradeoffs.

So we need to understand the context, and the forces in that context. And note too that forces (costs, tastes, ⋯) shift over time, so we want to understand that too.

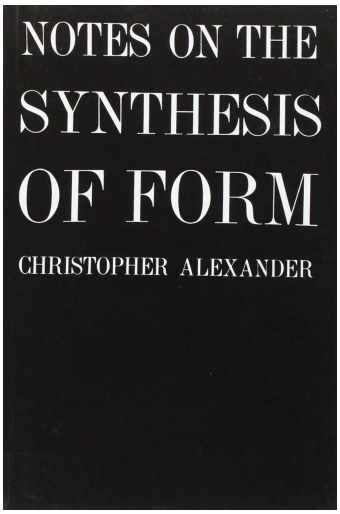
"One of the biggest architecture dysfunctions inside large organizations is architects justifying a decision they already made instead of exploring the solution space, understanding trade-offs, and making a conscious choice."

— Gregor Hohpe

Form and Context

"Every design problem begins with an effort to achieve fitness between two entities: the form and its context. The form is the solution to the problem; the context defines the problem."

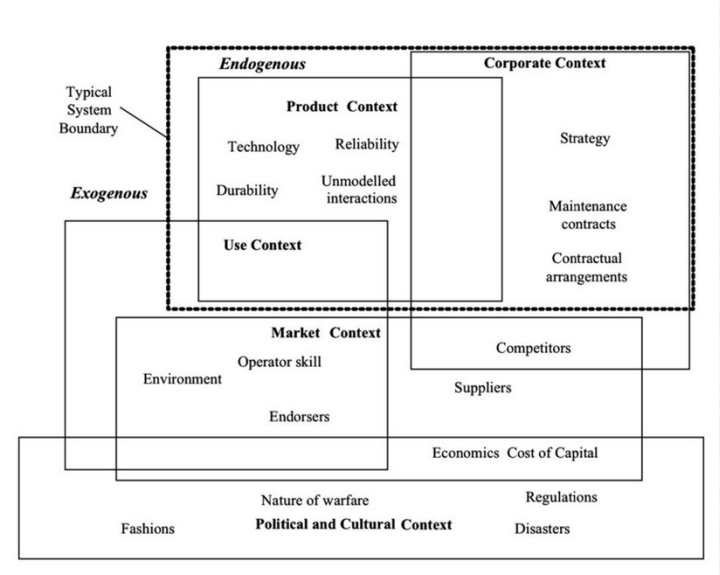
— Christopher Alexander, Notes on the Synthesis of Form, 1964.



Building our Understanding of the Context

Christopher Alexander and other building architects, emphasize the fit of architecture to context.

DeWeck et al (image below), in their classification of sources of uncertainty, delineate various contexts of (potential) relevance to us. Our contexts are sources of change, uncertainty and ambiguity, forces (gradients, pushes and pulls,..).



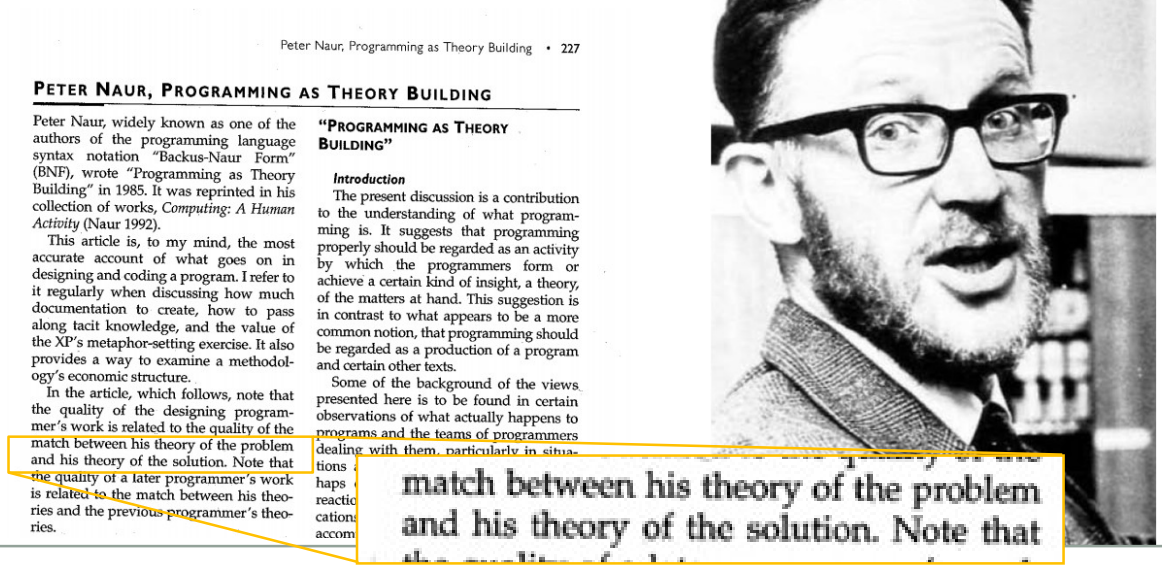
source: DeWeck et al, A Classification of Uncertainty

"The form is a part of the world over which we have control, and which we decide to shape while leaving the rest of the world as it is. The context is that part of the world which puts demands on this form; anything in the world that makes demands of the form is context. Fitness is a relation of mutual acceptability between these two. In a problem of design we want to satisfy the mutual demands which the two make on one another."

— Christopher Alexander, Notes on the Synthesis of Form, 1964

We're building a theory (of what the system is and ought to be, and how it works)

BUT SOFTWARE!



Peter Naur, Programming as Theory Building • 227

PETER NAUR, PROGRAMMING AS THEORY BUILDING

Peter Naur, widely known as one of the authors of the programming language syntax notation "Backus-Naur Form" (BNF), wrote "Programming as Theory Building" in 1985. It was reprinted in his collection of works, *Computing: A Human Activity* (Naur 1992).

This article is, to my mind, the most accurate account of what goes on in designing and coding a program. I refer to it regularly when discussing how much documentation to create, how to pass along tacit knowledge, and the value of the XP's metaphor-setting exercise. It also provides a way to examine a methodology's economic structure.

In the article, which follows, note that the quality of the designing programmer's work is related to the quality of the match between his theory of the problem and his theory of the solution. Note that the quality of a later programmer's work is related to the match between his theories and the previous programmer's theories.

"PROGRAMMING AS THEORY BUILDING"

Introduction

The present discussion is a contribution to the understanding of what programming is. It suggests that programming properly should be regarded as an activity by which the programmers form or achieve a certain kind of insight, a theory, of the matters at hand. This suggestion is in contrast to what appears to be a more common notion, that programming should be regarded as a production of a program and certain other texts.

Some of the background of the views presented here is to be found in certain observations of what actually happens to programs and the teams of programmers dealing with them, particularly in situa-

match between his theory of the problem and his theory of the solution. Note that

Image source: University of Copenhagen

Building our Theory of the Problem

Our field contends with complex software-intensive systems and their evolution, and one of the classics (1980) is "Programs, Life Cycles, and Laws of Software Evolution." In it, Meir Lehman observed:

> "The installation of the program together with its associated system [..] change the very nature of the problem to be solved. The program has become a part of the world it models, it is embedded in it. Analysis of the application to determine requirements, specification, design, implementation now all involve extrapolation and prediction of the consequences of system introduction and the resultant potential for application and system evolution. This prediction must inevitably involve opinion and judgment."

Peter Naur, in "Programming As Theory Building" (1985), argues

> "programming properly should be regarded as an activity by which the programmers form or achieve a certain kind of insight, a theory, of the matters at hand."

A theory, that is, of the problem* being solved, and how the code relates to and addresses this problem.

Returning to Lehman:

> "any program is *a model of a model within a theory of a model of an abstraction of some portion of the world or of some universe of discourse"*

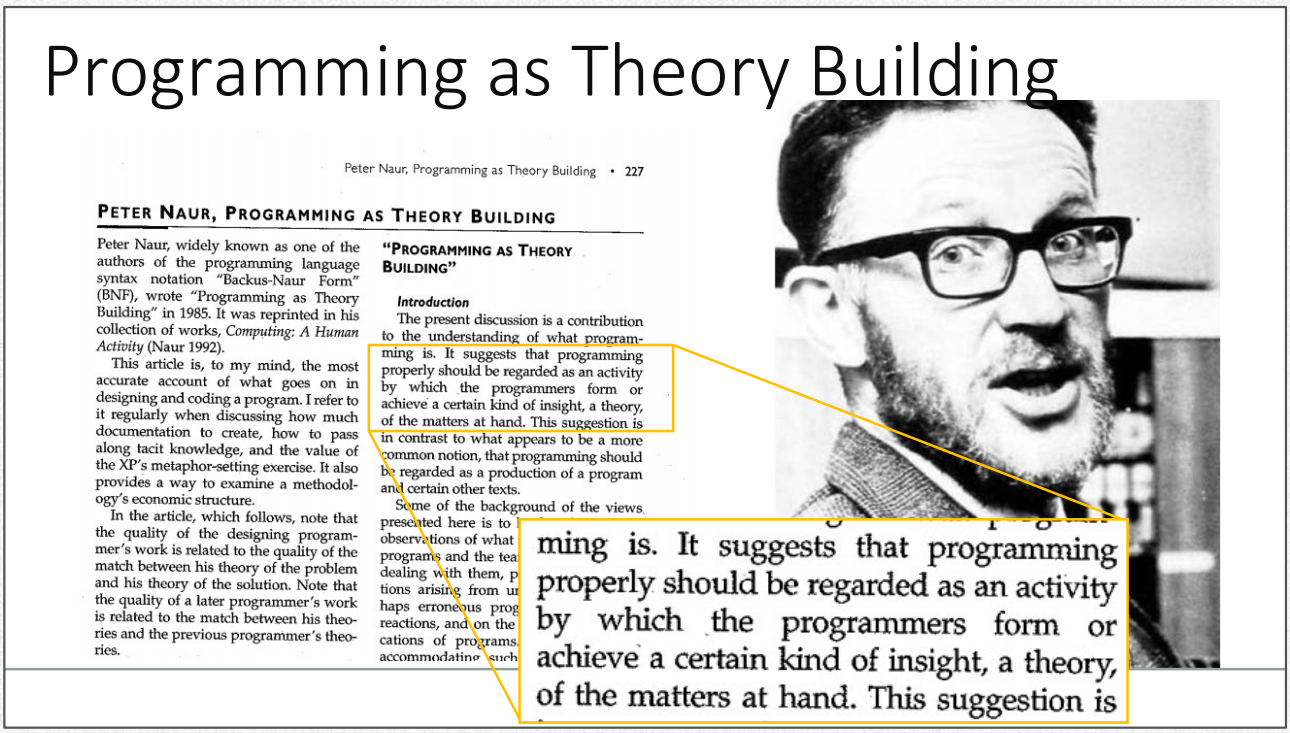
* Where the "problem" is the opportunity we're creating, the need we're addressing, etc, with the capability we're building.

Between Lehman, Floyd, and Naur, we have an important set of ideas for software, or any systems, really. We're building a theory, that informs our (design) decisions. We need to anticipate the impact of our decisions, in making them. And probe, to assess/amend our theory.

> *"what has to be built by the programmer is a theory of how certain affairs of the world will be handled by, or supported by, a computer program."*
>
> *— Peter Naur*

These classics advanced ideas about design that are important today

Programming as Theory Building



Theory Building

Peter Naur draws on Ryle's notion of theory and its role in intellectual (we might, today, rather emphasize cognitive) activity:

"where theory is understood as the knowledge a person must have in order not only to do certain things intelligently but also to explain them, to answer queries about them, to argue about them, and so forth. A person who has a theory is prepared to enter into such activities; while building the theory the person is trying to get it."

Peter Naur is articulating a theory of the difficulty of developing shared theories, and the importance of direct conversations in conjunction with the code to communicate our webs of understandings (or theories, or interrelated, interwoven mental models).

I would add that our theory (or system of theories) supports coherence and design integrity, and we're ever working towards requisite coherence among our (various folk design-build-evolving the system) individual mental models and perspectives. Seeking to understand the system, and its context(s) (or situation), and each other more, and evolve our theories in that direction, anyway.

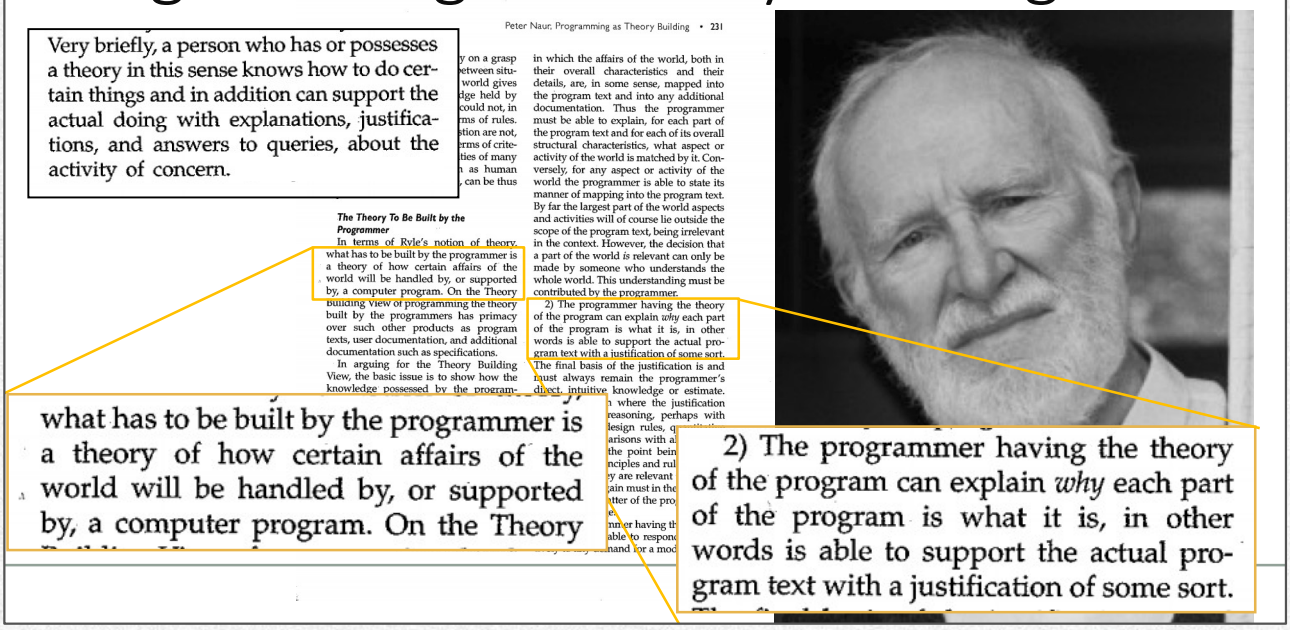
"people got their opinions
where do they come from?
each day seems like a natural fact
and what we think changes how we act."

—Why Theory? Gang of Four lyrics

"[A social] system always contains at least three elements or dimensions which are locked into one another: a social structure—which is a set of related roles and authority relationships—a technology and a theory. And by a theory I don't mean an academic or sociological theory about the system: I mean what it is that's believed that causes people in the system to do what they do."

— Donald Schon, Reith Lecture 2

Programming as Theory Building



Building our Theories of the Problem and Solution

Our systems change the world, based on our understandings and assumptions, so these are, well, important.

Alberto Brandolini (@ziobrando on twitter): "It is not the domain experts knowledge that goes into production, it is the assumption of the developers that goes to production" (via Krisztina Hirth)

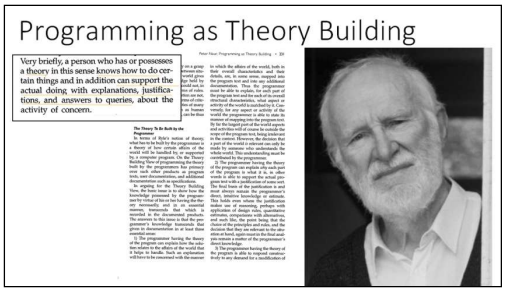
Theory is a way to talk about not just our mental models, and our reasoning-explaining and coherence making (attempts), but also a discipline of probing and testing our theories. Theories! Because we're not only developing theories of how our system works, or how it is structured to meet needs like adaptability and understandability, but theories of why the system matters and what matters (to users) and how.

And to the extent that we can build up shared, or at least sufficiently overlapping, theories, we're working on common ground (Klein et al) and requisite cohesion (Jabe Bloom). And we can probe and test elements of our theories, with thought experiments and walking through our reasoning "out loud" and "where we can see it," or user studies, or prototypes, as well as experiments in the A/B testing and market "bets" sense).

"It's developer's (mis)understanding, not expert knowledge that gets released in production"
— Alberto Brandolini

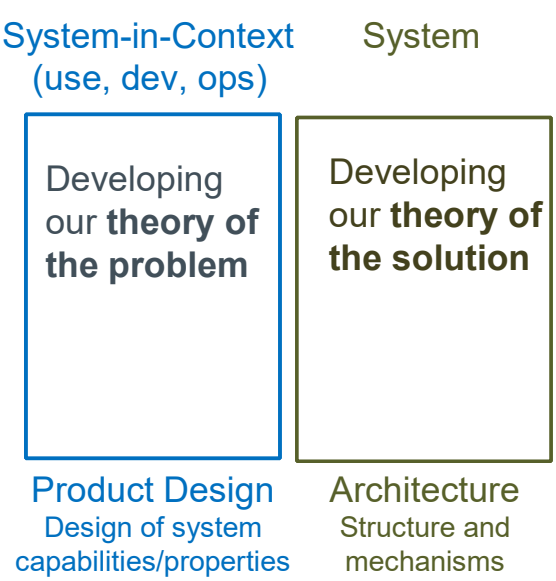
Brandolini's Law: "The amount of energy needed to refute bullshit is an order of magnitude bigger than that needed to produce it"

Design as Theory Building



"It's developer's (mis)understanding, not [domain] expert knowledge that gets released in production"

— Alberto Brandolini



Design of What the System Is and Is Becoming

We use design in two senses: the system (as currently built) has a design (noun; what the system is), and we design (verb) to make the system more the way *we* want it to be (we design to shape, or bring intent to what the system is becoming). *We*, there, is a complex! We, users. We, designers. We, business leaders. And more. And "we" all have different ideas and experiences of what the system is, and ought to become.

In the STELLA report, David Woods draws on Richard Cooks' diagram (figure 4) of the various people (roles) interacting with a system (developers, architects, operations, users), where below the line there are the artefacts (code that may be internally and externally sourced, tools like monitoring, deploy and testing tools, etc.) and above the green line "of representation" people are interacting with the system to do things they need to do, via their (unique to each) mental models. (https://snafucatchers.github.io/)

Design works to create a theory of not just what the system is and is becoming, but how that matters. And to build sufficient shared understanding.

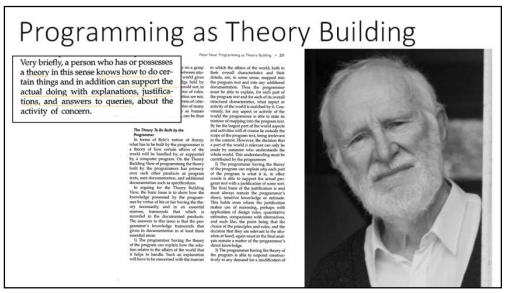
"When you analyse a problem you see what kind of problem it is, and identify the concerns and difficulties you have to deal with to solve it."

— Michael Jackson

"In analytical thinking the thing to be explained is treated as a whole to be taken apart. In synthetic thinking the thing to be explained is treated as part of a containing whole. The former reduces the focus of the investigator; the latter expands it."

— Russell Ackoff

BUT Ruuuth, it's 2026!


Programming as Theory Building

Very briefly, a person who has or possesses a theory in this sense knows how to do certain things and in addition can support the actual doing with explanations, justifications, and answers to queries, about the activity of concern.

"It's ~~developer's~~ [agent's] (mis)understanding[*], not [domain] expert knowledge that gets released in production"
— Alberto Brandolini

** yes, we need to avoid anthropomorphizing*

	System-in-Context (use, dev, ops)	System
	Developing our **theory of the problem**	Developing our **theory of the solution**
	Product Design Design of system capabilities/properties	**Architecture** Structure and mechanisms

Technical, Cognitive, and Intent Debt

From Margaret-Anne Storey:

"When developers think about a software system, they typically think first about code: the files, functions, and architecture that implement its features. But a software system exists across three distinct layers:

→ Goals and intent: the requirements, constraints, and objectives the system is meant to fulfill, as held by stakeholders and captured in specifications, tests, and documentation.

→ Code and structure: the representation of a system and the implementation of its intent: the source code, architecture, dependencies, and deployment infrastructure that make the system executable.

→ Shared understanding: the dynamic mental models that developers, architects, product managers, and other stakeholders hold about how the system works and how it can be reasoned about, what Naur called the "theory of the system" [Naur 1985]. Importantly, no single individual needs to understand all aspects of these layers; what matters is that sufficient shared understanding exists across the team to support safe change and coordination."


From Technical Debt to Cognitive and Intent Debt:
Rethinking Software Health in the Age of AI

Margaret-Anne Storey
University of Victoria, Canada
mstorey@uvic.ca

Generative AI is dramatically accelerating the velocity of software development, enabling small teams to ship features at a pace that would have seemed implausible just a few years ago [Peng et al. 2023]. I saw this firsthand in an entrepreneurship course I taught recently. Student teams were building software products over the semester, moving quickly to ship features and meet milestones. By week eight, one team hit a wall. Simple changes were breaking things in unexpected places, and progress had stalled. When I met with them, they initially blamed technical debt: messy code, hurried implementations, architectural shortcuts. But as we dug deeper, a different problem emerged. No one on the team could explain why certain design decisions had been made, or how different parts of the system were supposed to work together. The code might have been messy, but the deeper issue was that the team's shared understanding, the theory of the system [Naur 1985], had quietly fragmented. They had also failed to write down or communicate the rationale behind decisions. They had accumulated cognitive and intent debt faster than technical debt, and it had paralyzed them.

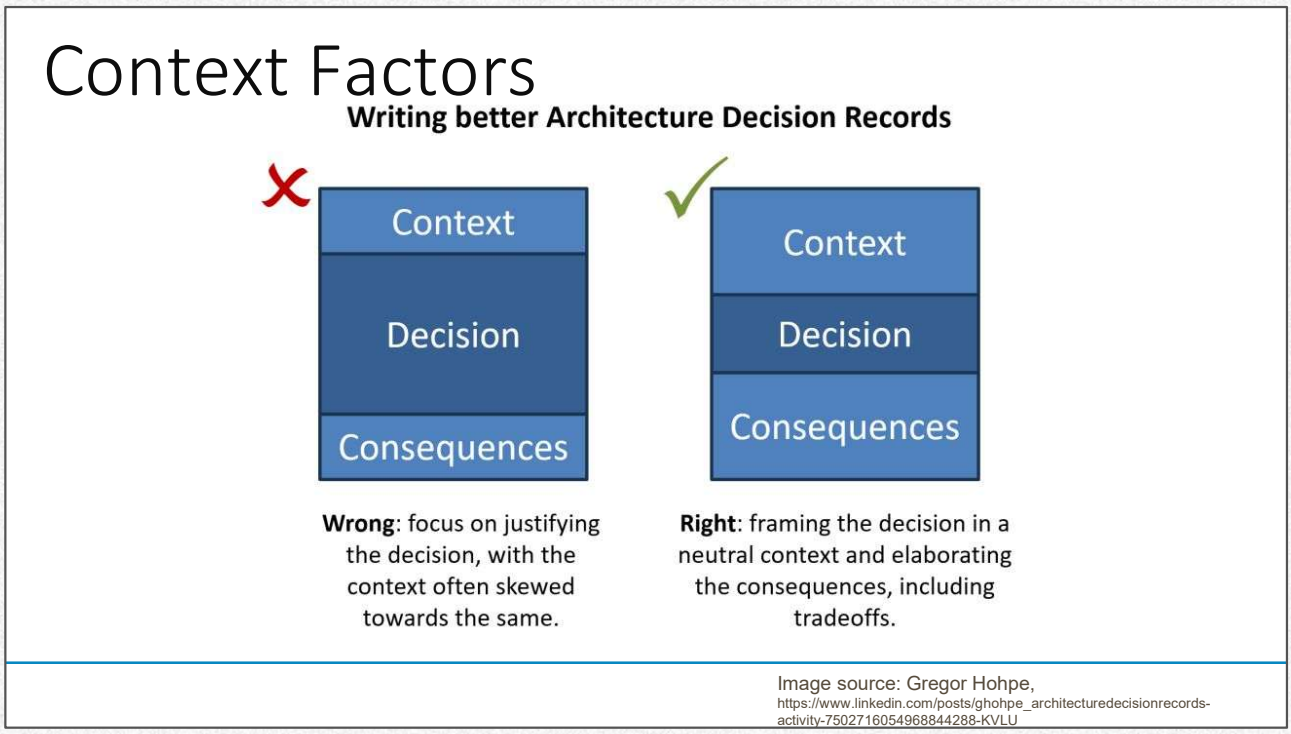
This is not an isolated story [Willison 2026]. Generative AI does not remove the challenges of software engineering; it redistributes them. In this article, I propose a **triple debt model for reasoning about software health**, built around three interacting debt types: **technical debt** refers to problems in the code layer, **cognitive debt** refers to erosion of shared understanding across a team over time, and **intent debt** refers to a lack of externalized goals, constraints, and rationale that both humans and AI systems need to work safely and efficiently with the codebase. Technical debt makes systems harder to change. Cognitive debt makes systems harder to understand. Intent debt makes it difficult to know what the system is actually for.

The Hidden Cost of AI-Generated Code

For years, software engineers have worried about **technical debt** [Cunningham 1993], the long-term cost of messy code that accumulates and compounds when teams prioritize speed over quality. But generative AI may be quietly shifting where the real risk lies. Today, AI systems can generate code faster than developers can read or understand it, and as those models improve, the systems show growing promise for reducing debt through automated refactoring, test generation, and automated code review [Hou et al. 2024]. That is, generative AI may reduce technical debt while simultaneously accelerating the accumulation of cognitive and intent debt. Organizational expectations for rapid productivity gains, without corresponding investment in learning support, can create a paradoxical effect where developers lack the time to develop the understanding that would actually save them time [Miller et al. 2026].

The code may work. It may even be well-architected. But the team may not adequately understand *how* it works or remember *why* it was built the way it was. Over time, the shared understanding that makes a software system safe to change quietly erodes. This gradual loss of understanding increases **cognitive debt,** while the loss of captured rationale leads to **intent debt.** In AI-assisted development, cognitive and intent debt may quietly become the risks that matter most.

Source: Margaret-Anne Storey, "From Technical Debt to Cognitive and Intent Debt: Rethinking Software Health in the Age of AI"



Context for Context

From Gregor Hohpe on LinkedIn:

> 'A major annoyance with frameworks or templates is that people will invariably use them incorrectly. **ArchitectureDecisionRecords** are a classic example. Folks zoom in on the Decision (perhaps lured in by the name) and fill in the other parts to suit their proposed decision.
>
> That's not what an ADR is meant to be. It's meant for you to explore the context and make a better-informed decision, including an understanding of the trade-offs that you made. If you only need a decision, you may as well flip a coin: it's faster and cheaper.
>
> Classic warning signals:
>
> - The context expresses preferences, trying to justify the decision ("open source solutions are better because...")
> - Consequences blindly accept downsides without any action ("it's more complex, bu oh well")
> - Lack of credible alternatives ("we could do nothing, but then we'd get nothing")'

"One of the biggest architecture dysfunctions inside large organizations is architects justifying a decision they already made instead of exploring the solution space, understanding trade-offs, and making a conscious choice."

— Gregor Hohpe

Quote and image source: Gregor Hohpe, https://www.linkedin.com/posts/ghohpe_architecturedecisionrecords-activity-7502716054968844288-KVLU

So how do we approach consequential decisions?

Decision Making

'You need strategies that help rule things out. That's the opposite of saying, "This is what my gut is telling me; let me gather information to confirm it."'

— Gary Klein



Conditions for Intuitive Expertise

A Failure to Disagree

Daniel Kahneman, Princeton University
Gary Klein, Applied Research Associates

This article reports on an effort to explore the differences between two approaches to intuition and expertise that are often viewed as conflicting: heuristics and biases (HB) and naturalistic decision making (NDM). Starting from the obvious fact that professional intuition is sometimes marvelous and sometimes flawed, the authors attempt to map the boundary conditions that separate true intuitive skill from overconfident and biased impressions. They conclude that evaluating the likely quality of an intuitive judgment requires an assessment of the predictability of the environment in which the judgment is made and of the individual's opportunity to learn the regularities of that environment. Subjective experience is not a reliable indicator of judgment accuracy.

Keywords: intuition, expertise, overconfidence, heuristics, judgment

In this article we report on an effort to compare our views on the issues of intuition and expertise and to discuss the evidence for our respective positions. When we launched this project, we expected to disagree on many issues, and with good reason: One of us (GK) has spent much of his career thinking about ways to promote reliance on expert intuition in executive decision making and identifies himself as a member of the intellectual community of scholars and practitioners who study naturalistic decision making (NDM). The other (DK) has spent much of his career running experiments in which intuitive judgment was commonly found to be flawed; he is identified with the "heuristics and biases" (HB) approach to the field.

A surprise awaited us when we got together to consider our joint field of interest. We found ourselves agreeing most of the time. Where we initially disagreed, we were usually able to converge upon a common position. Our shared beliefs are much more specific than the commonplace that expert intuition is sometimes remarkably accurate and sometimes off the mark. We accept the commonplace, of course, but we also have similar opinions about more specific questions: What are the activities in which skilled intuitive judgment develops with experience? What are the activities in which experience is more likely to produce overconfidence than genuine skill? Because we largely agree about the answers to these questions we also favor generally similar recommendations to organizations seeking to improve the quality of judgments and decisions. In spite of all this agreement, however, we find that we are still separated in many ways: by divergent attitudes, preferences about facts, and feelings about fighting words such as "bias." If we are to understand the differences between our respective communities, such emotions must be taken into account.

We begin with a brief review of the origins and precursors of the NDM and HB approaches, followed by a discussion of the most prominent points of contrast between them (NDM: Klein, Orasanu, Calderwood, & Zsambok, 1993; HB: Gilovich, Griffin, & Kahneman, 2002; Tversky & Kahneman, 1974). Next we present some claims about the conditions under which skilled intuitions develop, followed by several suggestions for ways to improve the quality of judgments and choices.

Two Perspectives

Origins of the Naturalistic Decision Making Approach

The NDM approach, which focuses on the successes of expert intuition, grew out of early research on master chess players conducted by deGroot (1946/1978) and later by Chase and Simon (1973). DeGroot showed that chess grand masters were generally able to identify the most promising moves rapidly, while mediocre chess players often did not even consider the best moves. The chess grand masters mainly differed from weaker players in their unusual ability to appreciate the dynamics of complex positions and quickly judge a line of play as promising or fruitless. Chase and Simon (1973) described the performance of chess experts as a form of perceptual skill in which complex patterns are recognized. They estimated that chess masters acquire a repertoire of 50,000 to 100,000 immediately recognizable patterns, and that this repertoire enables them to identify a good move without having to calculate all possible contingencies. Strong players need a decade of serious play to assemble this large collection of basic patterns, but of course they achieve impressive levels

Daniel Kahneman, Woodrow Wilson School of Public and International Affairs, Princeton University; Gary Klein, Applied Research Associates, Fairborn, Ohio.

We thank Craig Fox, Robin Hogarth, and James Shanteau for their helpful comments on earlier versions of this article.

Correspondence concerning this article should be addressed to Daniel Kahneman, Woodrow Wilson School of Public and International Affairs, Princeton University, Princeton, NJ 08544-0001. E-mail: kahneman@princeton.edu

© 2009 American Psychological Association 0003-066X/09/$12.00

Vol. 64, No. 6, 515–526 DOI: 10.1037/a0016755

TECHNICAL DECISIONS

Decisions Are Perfectly Rational, Right?

"Rigor is not a substitute for imagination."

— Gary Klein

How we think decisions are made: we list and weigh reasons. And demonstrate the superior approach to take. Gary Klein makes the case that experts tend not to do this (though novices might), especially not under (time) pressure. Still, when it comes to decisions of consequence to organizations and system design, we do well to better understand what's at stake, what's impacted and how, as well as what options or solution approaches we might take.

The rational in rationalize is a head-fake. And yet. We want to develop our reasons and reasoning. Make decisions with significant impact explicit, and probe and improve them.

" I worry about leaders in complex situations who don't have enough experience, who are just going with their intuition and not monitoring it, not thinking about it."

— Gary Klein

A Simple Model of a Decision



What we will do,
what approach we will take

TECHNICAL
DECISIONS

Decisions

So we're talking about how we make better consequential (system architecture, organizational architecture, strategic, etc.) decisions. So let's start there. With a decision, which we'll model, as one does, with black box or abstraction.

How does a decision come into view? In the previous module (Sense/Make Sense), we explored situation or context awareness and orienting to the landscape, identifying where action and leadership is needed. It is helpful (as we explore and clarify, and also as we document, the decision) to briefly describe the situation prompting the decision.

The Anatomy of a Decision

To understand something like a decision, and what factors in making a consequential decision, the structure (and diagram of structure) isn't enough — though how we structure our thinking about decisions, focuses attention and indicates what we seek to bring into view, for consideration. It helps to understand and frame the problem or situation, separately from identifying the solution or decision options, and determining relative fitness to the situation.

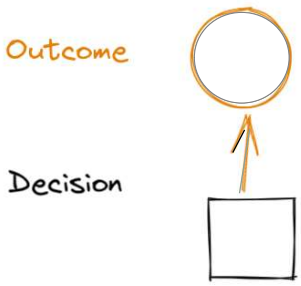
Of course, whether a decision "sinks or swims," depends on much, including the socio-political context, and how we influence and are influenced.



Image Source:
https://en.wikipedia.org/wiki/Anatomy

If our decision is about our response, about what we will do, the outcome is about what we want to achieve

Decision: Outcome



To achieve some outcome (have some stated positive impact, meet a goal), address some issue or challenge

Outcomes

In the context of a technical decision, an outcome may be a capability we need to build for users or the business or for the system (logging, or co-ordination and consistency mechanism, etc.), or a system property (quality attribute) we want to improve (scaling or latency or some other aspect of availability as we improve as demand grows or grows in new regions, etc.). Or it may be some issue (or risk) we face in the dev or devOps organization, that we want to address for ourselves, and see benefit to others in the organization. (This is often enough the case, that some decision templates use "issue" rather than "outcome" and it may even be separated out.)

The outcome sought, frames the problem that the decision addresses. It identifies what we are concerning ourselves with (as we explore and make this decision), and why.

The framing of the outcome or problem is itself a (set of) judgment call(s), as it helps bound the consideration space or frame the situation that we are attending to. Because it bounds the consideration space, we want to hold the frame somewhat loosely, at least to begin with, as we explore options (and possible reframings that bring other options into view).

Speaking of judgment calls, can we stop here?

For intentional, considered decisions, what is our intended outcome, goal, or objective? What does the decision seek to make true in this context or situation?

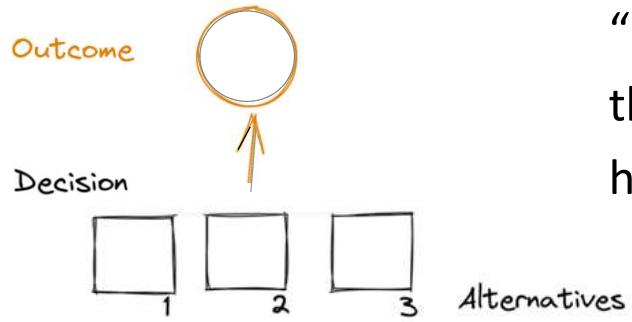
Problem: What are we trying to solve specifically? I try to be as specific as possible here. Since people have read the context, they can now understand the questions much better.

Opportunity: Here, I describe why solving this problem is essential and how it improves things for the business or the user.

Question to decide on: Describe in a sentence what question you are trying to answer.

From Indu Alagarsamy, *Document your product and software architecture decisions,* https://domainanalysis.io/p/document-your-product-and-software

Decision: Alternatives



"If you haven't thought of three possibilities, you haven't thought enough."

— Jerry Weinberg

TECHNICAL DECISIONS

Decision as a choice among options

Architecturally Significant

However. For strategically or architecturally significant decisions, we want to explore what our options are.

'"architecturally significant" decisions: those that affect the structure, non-functional characteristics, dependencies, interfaces, or construction techniques'

"One ADR describes one significant decision for a specific project. It should be something that has an effect on how the rest of the project will run."

— Michael Nygard, Documenting Architecture Decisions

That is, if we're making a technology choice that will shape other choices in an impactful way, or we're coming up with, designing, an approach to building a system capability or mechanism, or addressing some critical issue or challenge, we want to be intentional about it, to bring consideration to bear, and also to be able to visit and revisit our reasoning. So we bring options or alternatives into view. Moreover, as pointed out by Wisen Tanasa, it's helpful to consider whether a hybrid of what we've thought of as alternatives, positions us better in the tradeoff space.

Each option considered, is described briefly, outlining trade-offs, and impact. Typically the option proposed/adopted comes first in this list. You may want to describe why the other alternatives were not chosen, as it is part of the reasoning/argumentation (later when looking back at the decision, others can see which objections were already taken into account). — [source?]

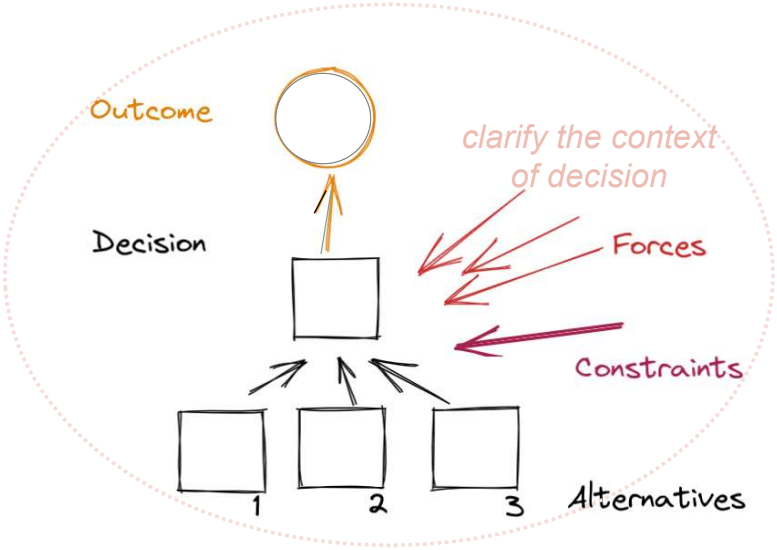
While we're at it, think of 3 ways we might be wrong!

" Eric Evans had recommended having at least 3 options in a proposal
1 option leads to evaluation of that option: yes/no
2 options lead to comparisons of A vs B
3 options suggest there are a set of possible solutions, of which there may be more."

— James Maier

Identify forces; what are the shaping considerations in this situation

Forces and Constraints: Decision



"A force [..] is [..] anything that has a potential non-trivial impact of any kind on an architect when making decisions."

— Uwe van Heesch et al

Forces, Considerations, What Impinges

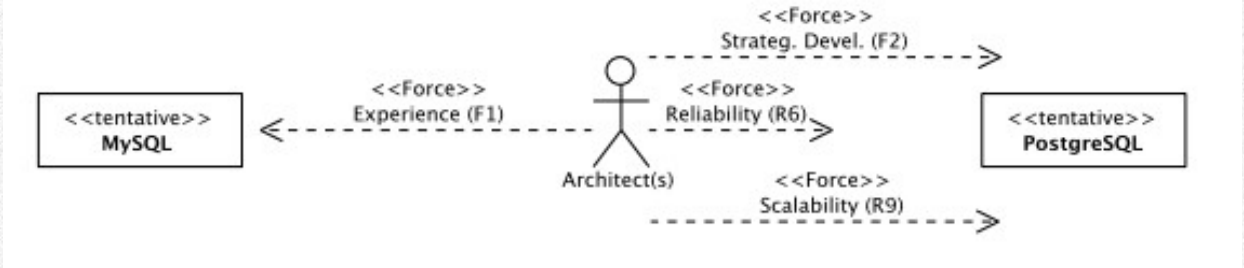
Whether we're weighing options or developing alternative approaches, the situation has a bearing — we need to identify and characterize the relevant forces, contributing factors, governing variables, complications, assumptions, constraints.

"A force [..] is [..] anything that has a potential non-trivial impact of any kind on an architect when making decisions." We're using force to mean something *impactful*, impinging on an architectural problem. Forces arise in the system or its environment — the operational, development, business, organizational, political, economic, legal, regulatory, ecological, social, etc.) context or situation.

"Forces arise from many sources; most often from requirements, but also from constraints, architecture principles and other "intentions" imposed upon the system; including personal preferences or experience of the architect(s) and the development team; and business goals such as quick-time-to market, low price, or strategic orientations towards specific technologies (see [9] for an empirical study on influence factors on software architecture)."

"The architect evaluates each architectural decision alternative in the context of the forces. As a result of the evaluation, a force can have a positive, negative, currently unknown, or neutral impact on the architect with respect to a decision; it either attracts the decision maker towards a specific decision alternative, or it repels the decision maker from an alternative, or it has no effect."

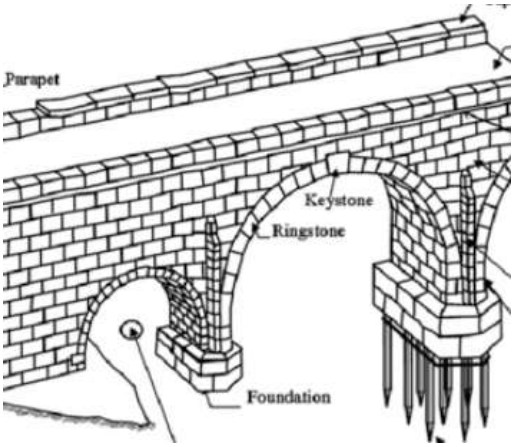
— Uwe van Heesch, Paris Avgeriou, Rich Hilliard
Forces on Architecture Decisions – A Viewpoint



Designing a Bridge: Forces

"Masonry is strong when you try to squeeze it and weak when you try to stretch it. In the jargon, it's strong in compression and weak in tension. That has consequences."

— Brian Marick



What forces are relevant, and how does our design behave under those forces?

Forces (in bridges and buildings)

"Suppose you're required to build a bridge, meaning a horizontal surface over some empty space. The simple solution would be a series of walls to hold up the floor of the bridge. OK, but now consider a horizontal floor span going from one wall to its neighbor. The span is supported on its ends, but unsupported in the middle. Gravity pulls down on the middle, creating tension. Since masonry is weak in tension, you'd have to have short spans and a lot of walls, which would be expensive, plus awkward if you want any traffic to go under the bridge — like, say, boats going down a river that it spans.

The arch is a clever solution to this problem. Consider an arch made out of bricks. Each brick mostly presses down on the brick next to and below it, meaning that all the bricks are in compression. The full weight of the structure supported by the arch is delivered to the feet of the arch. Some of the force is vertical, which is opposed because the arch is sitting on the ground. Some of the force is horizontal, which can be opposed if there's the leg of another arch of the same weight pushing against it - like in a bridge with multiple arches. Or, for the end two arches of the bridge, by anchoring them to a strong enough foundation. Essentially the forces transferred down the arch to the ground are balanced by forces *from* the ground, and it's all compression, all the time."

Source: Brian Marick, "Christopher Alexander's forces"

Forces push or pull, attract (gravity) or repel, inhibit (friction or drag, resistance) or propel (applied, spring), can be used to hold in place (tension, compression, ..)

"a flying buttress [..] uses the power of downward compression to balance an outward force. (Or something like that — I'm not an architect.)"

— Brian Marick

Identify what matters, what characterizes the "problem" or situation and impacts the solution or decision we're proposing

Potential Forces

What shapes this decision space?

- user or business need and criticality
- experience/capabilities
- system properties (availability, reliability, observability, auditability, ..)
- Costs (cost to build, license costs, etc.)
- Time: how long will this last? (short term impact, or something users/engineering will have to live with for long time)
- Time: engineering effort

What pushes or pulls, distorts or organizes, resists or attracts, …

- Time: time to value; feedback loops and learning cycles
- complexity., technical challenges
- team autonomy, independence, co-ordination costs
- consistency (UX, devX, OpX)
- Available capacity
- History (bad and good experiences)
- Uncertainty

What attracts or repels, inhibits or induces, creates friction, drag and inertia or flow, prevents or fosters, impacting the outcome in good or bad ways?

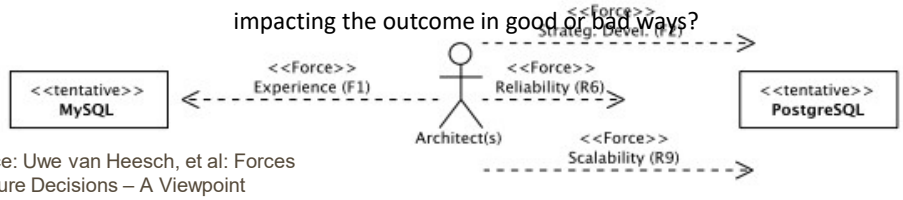


Image source: Uwe van Heesch, et al: Forces on Architecture Decisions – A Viewpoint

Forces, Considerations, What Impinges

As we're making a decision, and then as part of conveying it, we want to understand (and convey) what has substantive bearing on the decision. This means characterizing the situation in terms that are relevant to the decision.

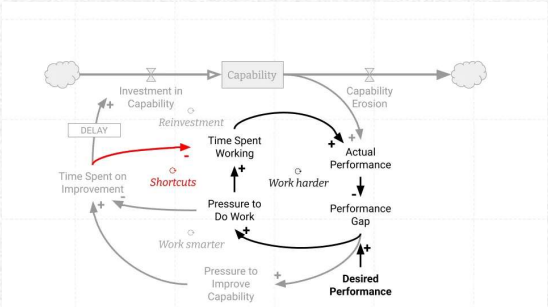
Whether we call them forces (or "forces" as an analogy) or factors or criteria, we're exploring what matters (in the use, development, operations, or broader context or situation), and how much it matters. And how that interacts. And what doesn't matter, that we thought might, and why.

What concerns do stakeholders have, that we need to take into account and address with this decision? Now, and as various stakeholders have to "live with" it. What makes a difference to the outcome and attributes of the solution, and how do the various alternatives we're weighing impact these concerns and goals (and objections)?

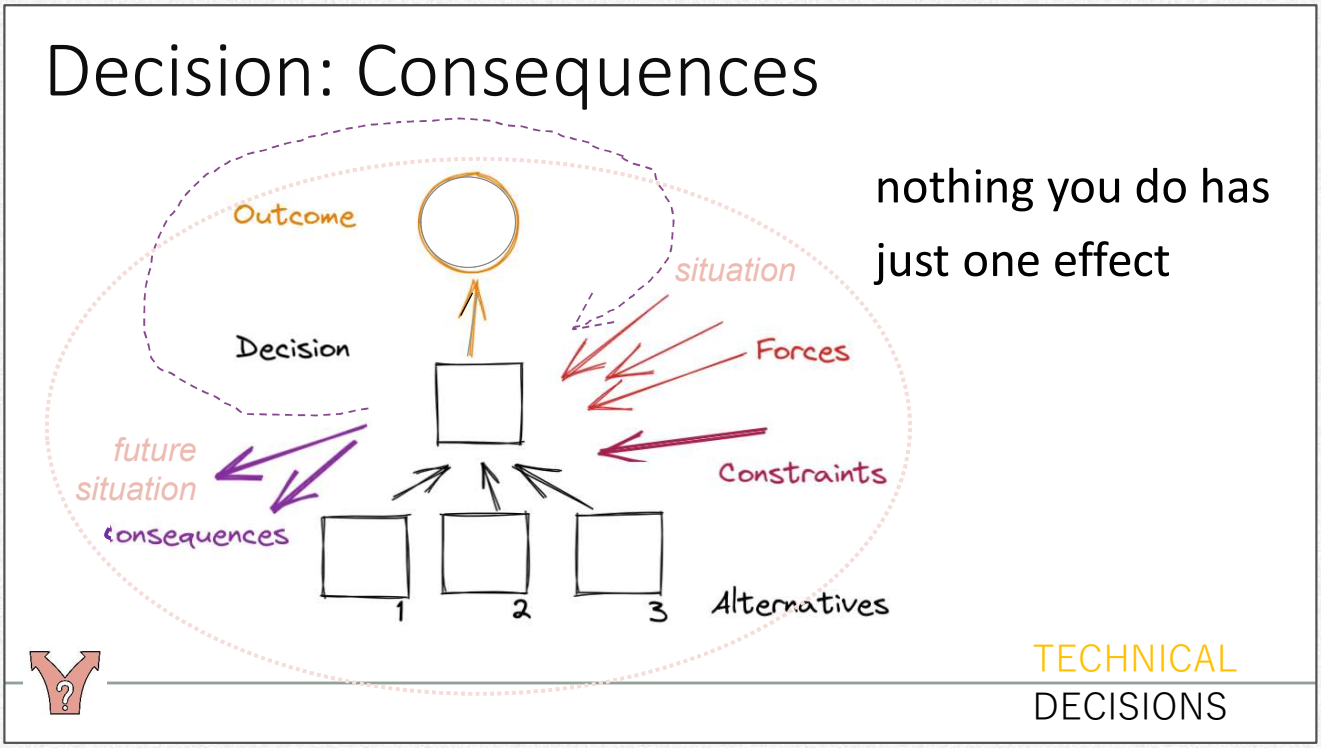
We want to identify what is consequential or significant to this decision, and get this out where we can see it, and reason about it and do so together, and bring others in to the process of identifying what matters and what interacts, and how we can best resolve the forces and tradeoffs (due to interacting and even conflicting goals and constraints).

Note about the diagram on slide: F2 is development of strategically important capability — it will become critical to the business, given the evolution of the system (increasingly large datasets, complicated queries, ⋯).

What matters to our situation? To our stakeholders, now and over time?



Causal loop diagrams can be used to explore effects (what is impacted, and how). Image source: Xavier Briand, What is technical debt? And how to talk about it?



Consequences and Second Order Effects

Further, in addition to the outcome or positive impact we're directing our attention to achieve by making this decision, and the forces and demands impinging on it, we also need to take into account, and weigh, the effects or consequences of the decision and arguably, the consequences of the consequences or second order effects. We won't know all of them in advance (they unfold, interacting with changing contexts), but exploring, and bringing in diverse perspectives, will bring much into view.

"Second-order thinking is the practice of not just considering the consequences of our decisions but also the consequences of those consequences. Everyone can manage first-order thinking, which is just considering the immediate anticipated result of an action. It's simple and quick, usually requiring little effort. By comparison, second-order thinking is more complex and time-consuming. The fact that it is difficult and unusual is what makes the ability to do it such a powerful advantage." — fs.blog

Consequence Scanning is an important approach to discovering the wider impacts of our technical products and choices. Ask:

- What are the intended and unintended consequences of this product or feature?
- What are the positive consequences we want to focus on?
- What are the consequences we want to mitigate?

More at https://doteveryone.org.uk/project/consequence-scanning/

"If you give a mouse a cookie,"
"he's going to ask for a glass of milk."
"When you give him the milk,"
"he'll probably ask for a straw"
"When he's finished, he'll ask for a napkin."
"Then he'll want to look in the mirror
To make sure he doesn't have a milk mustache."
— Laura Numeroff

To think more clearly about consequences, start with "what will be different, for whom?"

Consequences: What Changes?

"a consequence is just a statement about how the future will differ from the past"

— Michael Nygard

TECHNICAL DECISIONS

"As you make this list of consequences, try to avoid coloring your thoughts about the consequences by what your intentions are. [..] once the change is made your intentions are irrelevant. Only the resulting system state matters."

*— Michael Nygard**

* Michael Nygard, Consequences are not Pros or Cons, *https://www.michaelnygard.com/blog/2020/06/consequences-are-not-pros-or-cons/*

How Does the Decision Change Things?

In a blog post* that is a great companion to his post describing how he recommends documenting architecture decisions, Michael Nygard observes that we tend to focus on pros and cons, and can lean into justifying the choice we have or want to make. He notes:

"Instead, I suggest we first describe simply consequences, not benefits or problems. That's because a consequence is just a statement about how the future will differ from the past. [..]

Whether you judge that consequence to be a "pro" or "con" depends entirely on your relationship to the change. If you perceive the change as an improvement to status quo then you call it a "pro". If you don't like the version of the future which includes that consequence, then you call it a "con". That means labelling a consequence as a benefit is subjective. It describes the relationship of you and the change.

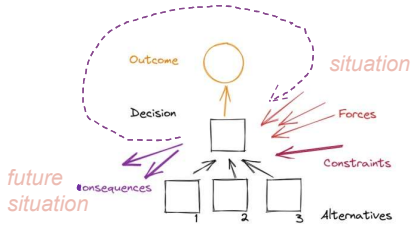
What about the changes that you don't particularly like or dislike? The ones that are neither "pro" nor "con"? Most of the time those don't get written down at all!

I recommend that you begin by listing the consequences. Find all the ways that the future will be unlike the past, if we choose that path. Look for second-order effects — the consequences of the consequences."

All of this is summarized in Michael Nygard's template for recoding architecture decisions

Document Decisions

aka think it through



Title: short noun phrase

Context: desired **outcomes** and the **forces** at play (probably in tension)

Alternatives

Decision: describes our response to these forces

Status: proposed, accepted, deprecated or superseded

Consequences: describes the resulting context, after applying the decision

From: Michael Nygard, Documenting Architecture Decisions, Nov 2011

Architecture Decision Records

ADRs are a way to share decision reasoning. Examples of ADRs at: https://web.archive.org/web/20210506014629/https://upmo.com/dev/decisions/0010-som-synthetic-monitoring.html

Michael Nygard's Template:

Title These documents have names that are short noun phrases. For example, "ADR 1: Deployment on Ruby on Rails 3.0.10" or "ADR 9: LDAP for Multitenant Integration"

Context This section describes the forces at play, including technological, political, social, and project local. These forces are probably in tension, and should be called out as such. The language in this section is value-neutral. It is simply describing facts.

Decision This section describes our response to these forces. It is stated in full sentences, with active voice. "We will …" Justify the decision.

Consequences This section describes the resulting context, after applying the decision. All consequences should be listed here, not just the "positive" ones. A particular decision may have positive, negative, and neutral consequences, but all of them affect the team and project in the future.

The consequences of one ADR are very likely to become the context for subsequent ADRs. This is also similar to Alexander's idea of a pattern language: the large-scale responses create spaces for the smaller scale to fit into.

"In practice, our projects almost all live in GitHub private repositories, so we can exchange links to the latest version in master. Since GitHub does markdown processing automatically, it looks just as friendly as any wiki page would."

— Michael Nygard

"Writing about your decision forces you to explain your thinking." — fs.blog, Creating a Decision Journal

Decisions Are Trade-offs

"For me, "engineer" means knowing that all decisions are tradeoffs. It means considering both upsides & downsides of each technical choice, and doing so with explicit consideration of the larger system context."

– Sarah Mei



TECHNICAL DECISIONS

"When you build a bridge, you don't build it as a perfect structure that will never collapse. Instead you build it to withstand 500 year winds, 200 year floods, 300% expected maximum load, etc. If you didn't make these design trade-offs, every bridge would be solid concrete [..] Engineering is all about making these compromises"

Pragprog.com/articles/the-art-of-tradeoffs

Decisions Entail Tradeoffs and Tradeoffs Don't Stay Their Lane ¯_(ツ)_/¯

As a manager in IT or product development, our decisions don't just impact teams but the systems they create. We see this in Conway's Law:

> "The basic thesis [..] is that organizations which design systems [..] are constrained to produce designs which are copies of the communication structures of these organizations."
>
> -- Melvin Conway, How Do Committees Invent?, 1968

Likewise, as an architect, the choices we're making are technical, but the impacts don't remain neatly in the technical space. The tradeoff space isn't just about qualities that impact developer experience, or security properties or operational complexity, but user experience and partner experience through properties of the system in use. And more. So we investigate the upsides and downsides of our technical decisions, in these various contexts.

We want to surface the trade-offs inherent in our decisions, both to better understand the decision space, and because we may be able, or need, to contend with the downsides of these decisions explicitly, to offset them.

Leadership isn't inherently about hierarchy. Though hierarchy is not irrelevant.

An Example

Read (next slide) and identify

- the Decision
- the Outcome(s)
- Forces (identified, and not)
- Consequences (identified, and not)

Spotify

This organization structure, combined with the global-ish nature of JavaScript in the browser, has made us build the desktop client UI out of many small, self-contained web apps called *Spotlets*. They all run inside Chromium Embedded Framework, each app living within their own little iframe, which gives squads the ability to work with whatever frameworks they need, without the need to coordinate tooling and dependencies with other squads. While this approach has the disadvantage that we have many duplicate instances of different versions of libraries, increasing the size of the app, but it offers the *massive* advantage that introducing a library is a discussion between a few people instead of decision that involves ~100 people and their various needs. Not only would such a big discussion extremely time-consuming and hard, it would also force us to use a least-common-denominator approach to picking libraries, instead of picking the ones specifically tailored to the problem domain of each squad. Considering the size of a single song compared to the size of a JavaScript library, this trade-off is a no-brainer for us.

Mattias Petter Johansson, on Quora (2017)

TECHNICAL DECISIONS

Weigh tradeoffs

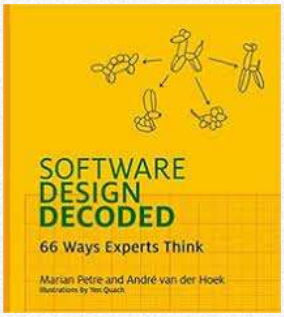
To make better decisions, we need to weigh and resolve the inherent tradeoffs — the upsides and downsides of the choice or approach.

That is, to make tradeoffs intentionally, we need to identify and characterize the tradeoff space. What is relevant to the decision is a (set) of judgment calls. How we balance and resolve the tradeoffs is again a set of judgment calls (though of course there may be precedent in the industry, or in our experience, that gives us more to go on).

	Design Alternative 1	Design Alternative 2	Design Alternative 3
Development time	****	**	******
Cost of acquiring COTS components	******	***	****
Reuse of our existing codebase	******	*******	****
Compatibility	******	****	*******
Performance	******	******	*******
Security	****	*******	****

"strive for the least worst combination of trade-offs"
— Neal Ford et al

Design Alternatives image from:



We're going to consider an example (next slide)

A remote presentation like this has some advantages in terms of screen distance, but for those who can't read the screen we will recap some of the main points in just a bit. Now, though, we will take a moment to allow a chance to read the text on the next slide, and identify the decision, the outcome, the forces impinging on this situation (those identified in the description, and those your experience is prompting) and consequences or effects of this decision.

Exercise: Read this narrative description of a decision

Spotify

This organization structure, combined with the global-ish nature of JavaScript in the browser, has made us build the desktop client UI out of many small, self-contained web apps called *Spotlets*. They all run inside Chromium Embedded Framework, each app living within their own little iframe, which gives squads the ability to work with whatever frameworks they need, without the need to coordinate tooling and dependencies with other squads. While this approach has the disadvantage that we have many duplicate instances of different versions of libraries, increasing the size of the app, but it offers the *massive* advantage that introducing a library is a discussion between a few people instead of decision that involves ~100 people and their various needs. Not only would such a big discussion extremely time-consuming and hard, it would also force us to use a least-common-denominator approach to picking libraries, instead of picking the ones specifically tailored to the problem domain of each squad. Considering the size of a single song compared to the size of a JavaScript library, this trade-off is a no-brainer for us.

Mattias Petter Johansson, on Quora (2017)

Example

Let's spend a moment and read the discussion (see slide above) from Mattias Peter Johansson on Quora, about Spotify (written in 2017). Ref: https://www.quora.com/How-is-JavaScript-used-within-the-Spotify-desktop-application-Is-it-packaged-up-and-run-locally-only-retrieving-the-assets-as-and-when-needed-What-JavaScript-VM-is-used/answer/Mattias-Petter-Johansson

This was written years ago, about the past; things changed.

Of note (from the same post by Mattias Johansson), related to (ir)reversible decisions: "The latest versions of all Spotlets are zipped and bundled with the desktop client binary on every release, assets and all. However, we can also live-deploy new versions of the apps into the running clients of users. This is very useful for emergency releases, but we can also release Spotlets to certain user groups, such as employees or designated testers only. We can even release apps that aren't originally bundled with the client, which is very handy during our hack weeks, allowing us to quickly get apps into the hands of non-techie decision makers in the company. After demoing your thing, you end off with "and now type this uri into the search field of your desktop client, to try it out with your own account!". This whole system was originally invented as part of the now-defunct Spotify Apps initiative, to support third-party developers adding functionality to the client."

"Almost everything in architecture is a trade-off. You never get all of it. You make the best decision you can with what you know and you accept that something will be lost either way. The skill is not finding the option with no downsides. Usually, there is no such option. The skill is understanding what each choice costs and deciding which cost you can live with."

— Karol Wiszowaty

What did you notice?

Spotify

Decision

Divergence

–

–

+

Team autonomy

This organization structure, combined with the global-ish nature of JavaScript in the browser, has made us build the desktop client UI out of many small, self-contained web apps called *Spotlets*. They all run inside Chromium Embedded Framework, each app living within their own little iframe, which gives squads the ability to work with whatever frameworks they need, without the need to coordinate tooling and dependencies with other squads. While this approach has the disadvantage that we have many duplicate instances of different versions of libraries, increasing the size of the app, but it offers the *massive* advantage that introducing a library is a discussion between a few people instead of decision that involves ~100 people and their various needs. Not only would such a big discussion extremely time-consuming and hard, it would also force us to use a least-common-denominator approach to picking libraries, instead of picking the ones specifically tailored to the problem domain of each squad. Considering the size of a single song compared to the size of a JavaScript library, this trade-off is a no-brainer for us.

Mattias Petter Johansson, on Quora (2017)

What do we Notice?

Our point here isn't to criticize Spotify's choices in that timeframe and point of the evolution (in the market and of the technology and organization), but to appreciate how, even in this narrative format, so much of the decision and considerations are being conveyed, and to explore the decision.

The decision: to use Spotlets, or small, self-contained apps within their own iframe

The outcome: increased team independence or autonomy

Positive effects (or forces): reduced cross-team co-ordination; speed of movement (so speed of learning)

Negative effect (or force): duplicate instances of different versions of libraries

Negative consequence: reduced cross-team communication; divergence among teams as a result

(These social costs and consequences are not just as a result of this decision, but the decision is part of a reinforcing loop.)

Negative consequence (not surfaced; potentially future): multiple versions of licenses and purchasing and security headaches (knowing what patches to roll out where)

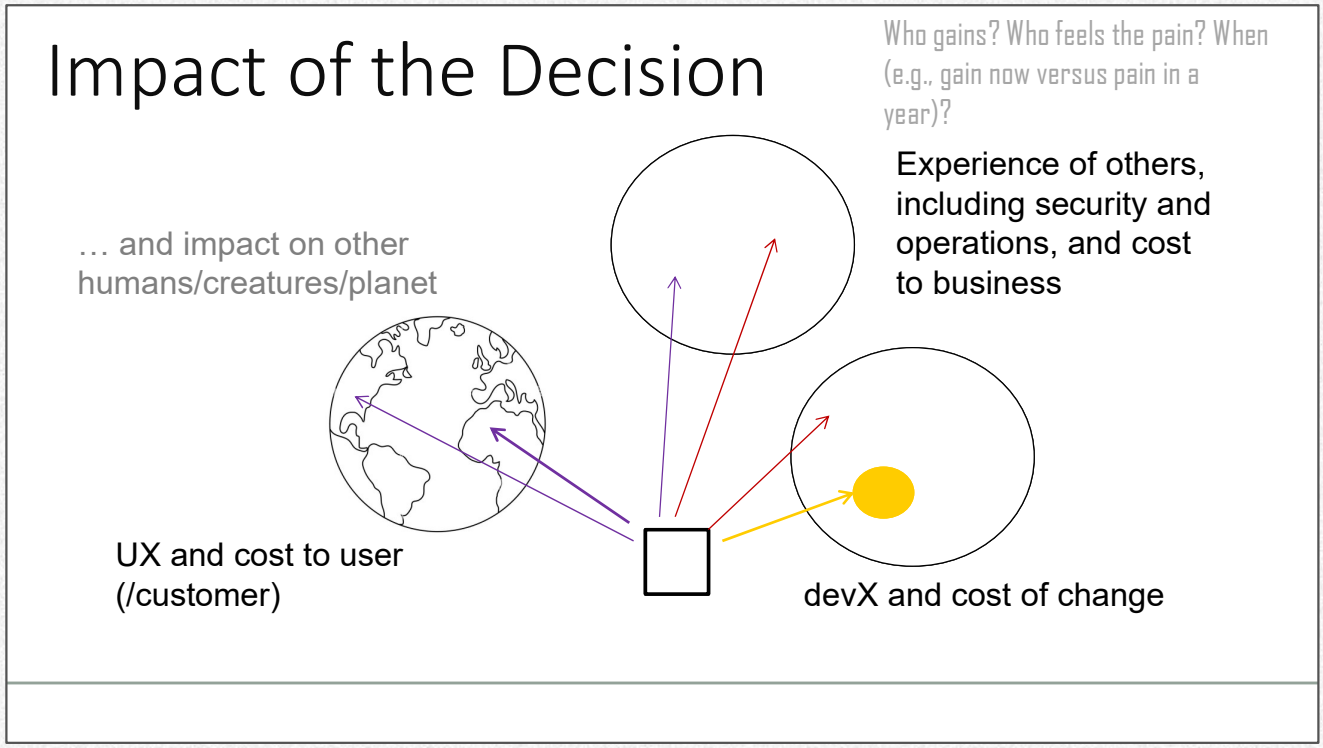
Tradeoff? size of songs so dominates size of app, that they could make this decision to support team autonomy without perceived cost to user.

We see that allowing duplicate instances of different versions of various libraries enabled Spotify squads (teams) considerable independence, removing the need to coordinate with other squads on libraries and versions. Because song size so dominates considerations that it generally falls beneath the threshold of sensitivity for the user, the tradeoff of team freedom for app size is easily (in their view) within the design acceptance envelope.

So in this case, a technical decision is being made for organizational gain (lowering team coordination costs and increasing team's degrees of freedom) at the expense of app size, which works as long as it's below the app user's tolerance threshold for resource consumption.

We're building econo-sociotechnical systems, within econo-sociotechnical systems, and we need to factor this in, as we scan for forces, constraints and consequences (that we factor in as forces).

Leadership isn't inherently about hierarchy. Though hierarchy is not irrelevant.



Different Impact in Different Areas

What this example highlights, is "what's going on" in terms of what is being paid attention to in the decision, what the forces and tradeoffs are and what has not been drawn into explicit consideration, possibly because it isn't yet a significant noticed force. And in particular, this important point: impacts (positive outcomes, as well as other positive and negative effects) and consequences (including downstream and future consequences) are borne by different sets of "stakeholders" – not just different persons or internal groups, but users (downloading the app and listening to songs), customers (paying bills), these people in different regions of the world, with different bandwidth and cost constraints. As well as different stakeholders within the organization, and not limited to developers.

But we would draw on experience to point out what to be watching for, as the situation evolves.

"Good engineering is less about finding the "perfect" solution and more about understanding the tradeoffs and being able to explain them."
— Jaana B. Dogan

[Reflecting on the Ackoff video] "The systemic cultural and societal impacts of the software we build: I feel that especially in venture capital backed startups, the software industry is prone to not thinking in systems when it comes to the impact of what their products are doing — as opposed to the return on investment they have. From the harms of social media on mental health, to discriminatory bias in AI, I see many parallels with the notion of "doing the wrong thing right." — Mike Stallard

Tradeoffs may only kick in after some threshold has been reached

Decision Space and Pareto Front

"Tradeoffs only occur when you reach [a] Pareto frontier."

— Donald Reinertsen



Pareto efficiency

From Wikipedia, the free encyclopedia

Pareto efficiency or **Pareto optimality** is a situation where no individual or preference criterion can be made better off without making at least one individual or preference criterion worse off. The concept is named after Vilfredo Pareto (1848–1923), Italian civil engineer and economist, who used the concept in his studies of economic efficiency and income distribution. The following three concepts are closely related:

- Given an initial situation, a **Pareto improvement** is a new situation where some agents will gain, and no agents will lose.
- A situation is called **Pareto dominated** if there exists a possible Pareto improvement.
- A situation is called **Pareto optimal** or **Pareto efficient** if no change could lead to improved satisfaction for some agent without some other agent losing or, equivalently, if there is no scope for further Pareto improvement.

Pareto Front

What we're seeing in this example, is that, with respect to team degrees of freedom and app size on the one hand, and song size and by implication user experience and space and cost concerns, a Pareto Front has not being reached. These things aren't being traded off for one another. We can improve team independence without decreasing user experience in an appreciable way.

Experts and Seeing Tradeoffs

From John Cutler*:

> 'Ask an everyday driver about driving tradeoffs, and you'll likely hear something like, "When you go around a corner, you need to trade off speed and control." The mental model is "slow down just enough to keep control around the corner."
>
> A professional driver will think differently. Their mental model revolves around tire grip and temperature, the optimal racing line, throttle control, suspension, aero settings, brake balance, tuning the car for the track, and weight transfer management. They might point out that the amateur isn't exactly wrong, but they might say, "At the end of the day, it boils down to tire friction, aerodynamic limits, mechanical limits, and human limits."
>
> Amateurs aren't entirely wrong; they're recognizing some inherent limitations. What differentiates professionals is their ability to approach these limits more closely and consistently without exceeding them. They have a deeper understanding of where the boundaries are and how to navigate them.'

"A threshold effect exists when there is a critical level of effort necessary to affect the system. Levels of effort below this threshold have little payoff."

— Richard Rumelt

* John Cutler, "Dear Executive..." 2023, https://cutlefish.substack.com/p/tbm-250-dear-executive

Trade-offs

Space-Time Trade-Off



- Spotify Example:
  - ↑ Size of songs, to
  - ↓ Co-ordination overhead between teams

TECHNICAL DECISIONS

"A trade-off (or tradeoff) is a situational decision that involves diminishing or losing one quality, quantity or property of a set or design in return for gains in other aspects. In simple terms, a tradeoff is where one thing increases and another must decrease."

— wikipedia

Space-Time or Time-Memory Trade-Off

"Usually, a TMTO is developed to improve the speed of an algorithm by utilizing one-time work, which results in increased storage (memory) requirements when the resulting algorithm is executed. Of course, it is also possible to work in the opposite direction by reducing the one-time work at the expense of more work each time the algorithm is executed. The goal is to balance the one-time work (memory) requirement with the speed of the algorithm (time)."

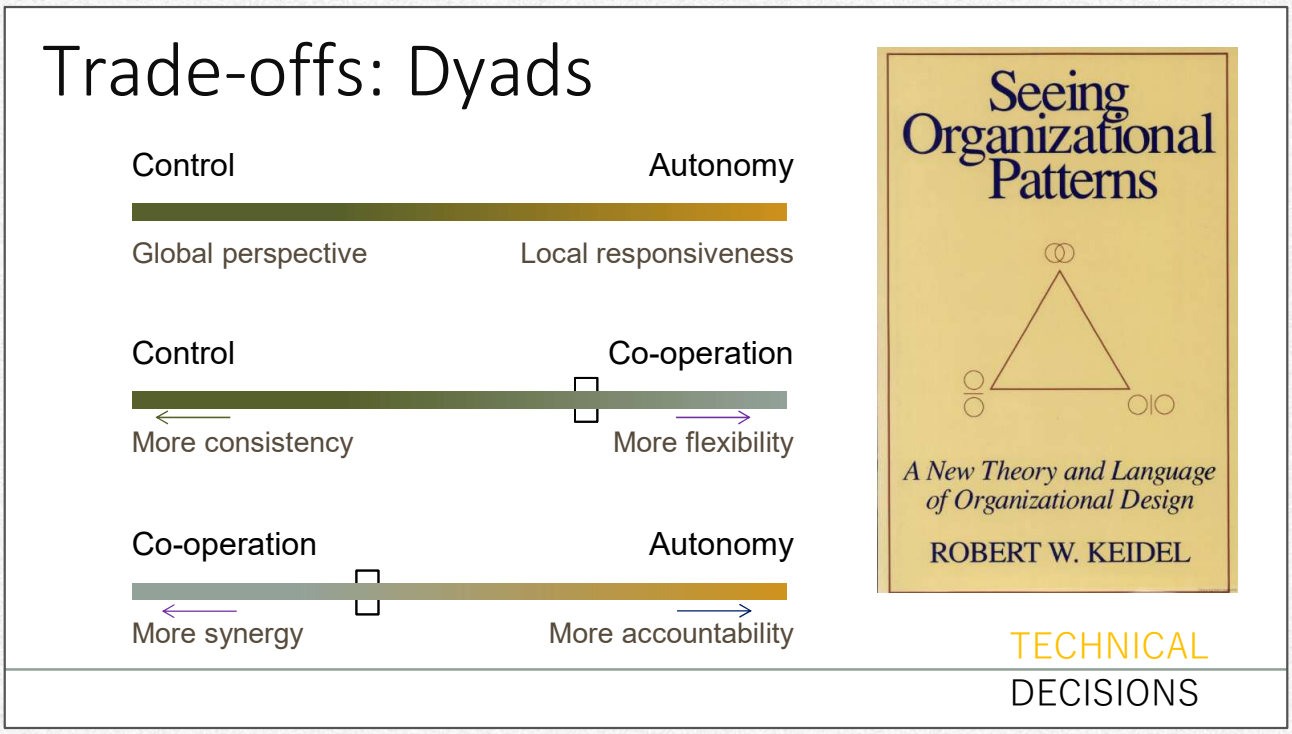
— Mark Stamp, Once Upon a Time-Memory Tradeoff

A classic illustration of the trade-off entails using a lookup table (uses upfront work and a lot of space to enable a fast lookup when the result is needed) versus calculating on demand (uses little space, but can take a long time at the point of demand, depending on the calculation).

Another space-time trade-off arises in data storage. If data is stored uncompressed, it takes more space but less time than if the data were stored compressed.

We're talking about this as a space-time trade-off, but it translates into a cost-performance (i.e., user experience) trade-off.

What are we giving up and what are we gaining? Do different groups gain and feel pain? Over different time horizons?



"For example, continuous evolution pulls against product stability[..]. Low-level decisions pull against strict process control"

— Eberhardt Rechtin and Mark Maier

Trade-Off Dyads (Picturing the Dilemma)

We have a trade-off when design variations improve one dimension (something we value, like a performance metric), but diminish another. Factor in multiple of these trade-off dimensions, and there is no unique optimal design; the choice lies in what is valued in that context.

By drawing the trade-offs out — making them visible — we can make judgments, and subject them to discourse to better assess impact and value.

Many trade-offs can usefully be thought of in terms of dyads: performance and cost (another way to frame the space-time trade-off); data confidentiality or security (via encryption) and performance; safety and cost; structural mass (for physical structures) and safety; usability or convenience and security; etc.

In *Seeing Organizational Patterns*, Robert Keidel considers organizational structures and interaction dynamics, and pivotal trade-offs underlying organizing choices.

These could be presented as the dyads shown (slide above).

While considering pair-wise trade-offs can help understand the design space, it can obscure the tensions when multiple variables are simultaneously in play. Keidel points out that "every organization must blend autonomy, control, and cooperation." The trade-off space (the design options), is more usefully visualized as a triad, or triangle.

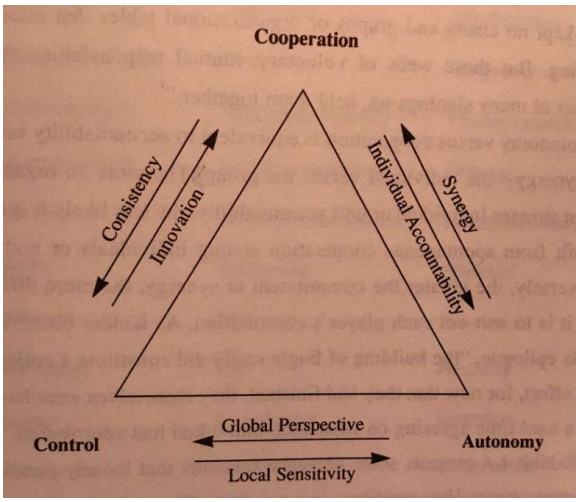
The multiple library versions example earlier, is missing impacts (eg security implications).

But really, it's a trade-off space. An example with more than two variables

Trade-offs: Triads

"most organizational issues are a balance of three variables: individual autonomy, hierarchical control, and spontaneous cooperation. By learning to frame issues as trade-offs among these design variables, one can see underlying patterns"

– Robert Keidel



TECHNICAL DECISIONS

A Trilemma of Trade-offs

According to Keidel, any particular organization will focus on at most two of autonomy, control, and co-ordination. (Attempting all three is an unstable form.) These are the organizational forms he identifies:

Organizations that are autonomy-based have as their distinctive competence adding value through solo performers; they are truly star systems. Example: any first-rate university.

Control-based organizations compete on the basis of their ability to reduce costs and/or complexity through global coordination. Authority, information, and initiative reside chiefly at the top levels.

A cooperation-based organization builds synergy across teams. The distinctive organizational competence is innovation through cooperation.

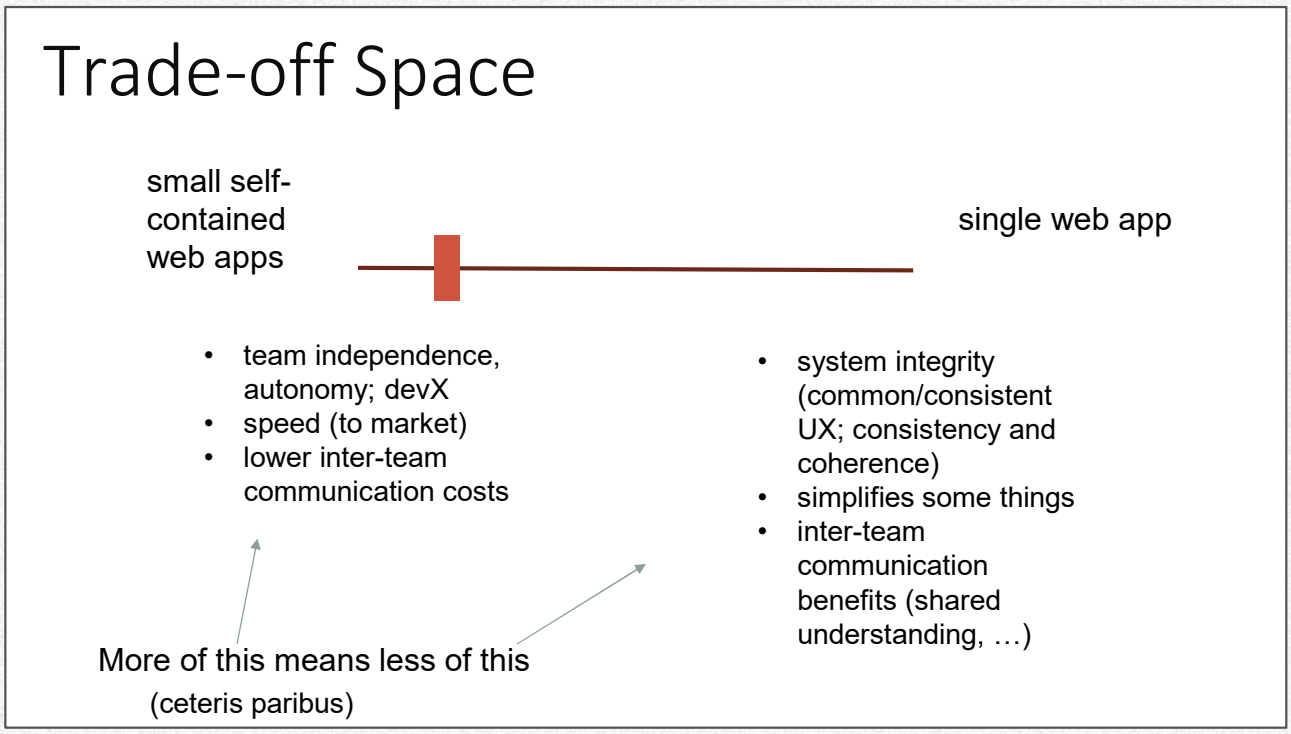
Probably the most familiar example of an autonomy/control hybrid is the divisionalized corporation.

A control/cooperation hybrid may be described as a "humanistic hierarchy." Top-down control remains essential but every effort is made to meld it with voluntary, lateral processes among individuals, functions, and units.

The autonomy/cooperation has the oldest roots. This combination goes back to the craft organizations of the late 18th century, which featured a blend of individual initiative and informal cooperation.

"Equally dangerous is an overemphasis on a single variable to the point that the other two are neglected. Autonomy becomes problematic when a relatively freestanding part-individual or organizational unit-overdoes its own thing."
— Robert Keidel

Seeing Organizational Patterns, Robert Keidel



Choices Among Options

When we are deciding among alternatives, we're deciding among the clusters of effects and consequences of those alternatives (like modular monolith or microservices; small self-contained web apps or single web app; etc.).

While the concept of "to decide" holds within it the notion of what we're deciding *not* to do, along with what we are deciding *to* do, part of (what we factor in) the trade-off space may include what it takes to mitigate the negative effects or downsides of the approach we go with.

Examples

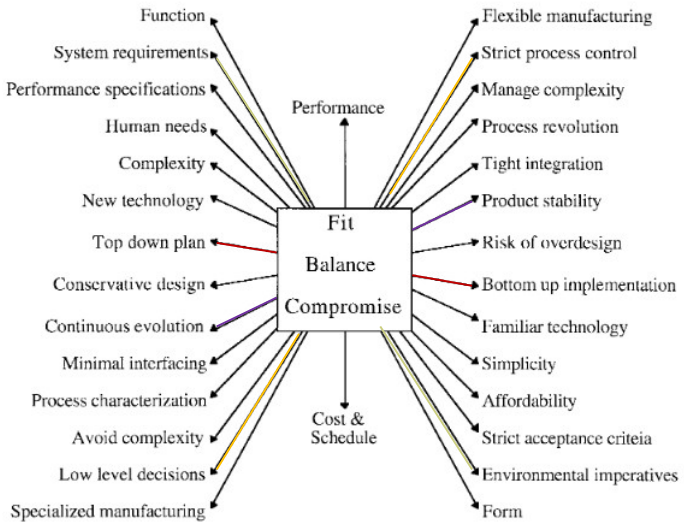
We might seek to minimize downtime with rapid failure detection and recovery, but this incurs the overhead of continuous monitoring and detection. Additionally, automated detection and recovery mechanisms may be triggered by false positives (for example, a node acting as if it has failed, when it's just running slowly for a moment) or introduce performance degradation during failover. Balancing the trade-offs involves optimizing detection sensitivity and response times while minimizing false alarms and impact on performance.

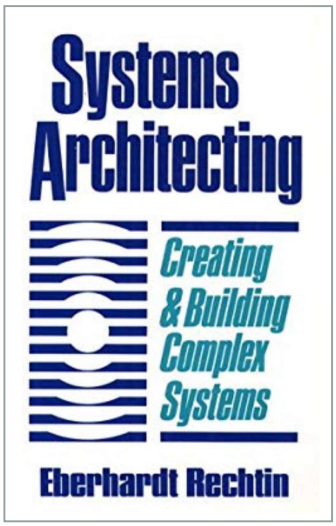
"Two key questions I always advise people to reflect on [..]:
1. What happens if this succeeds? Does it make the [..] world better?
2. Who is harmed by the changes this causes? What would you choose to do if you loved them?
Every single choice gets easier if you know those answers.'

— Anil Dash

"A central tenant of the ecosystem approach is that the path to sustainability is one of tradeoffs. Science can illuminate the tradeoffs but a resolution, that is, the choice of path, is a political decision" — Michelle Boyle, et al

Design Force Field





TECHNICAL
DECISIONS

Tensions

Design has to balance tensions caused by different imperatives, needs, and perceptions.

"Some of competing technical factors are shown in [the figure in the slide above]. This figure was drawn such that directly opposing factors pull in exactly opposite directions on the chart. For example, continuous evolution pulls against product stability; a typical balance is that of an architecturally stable, evolving product line. Low-level decisions pull against strict process control, which can often be relieved by systems architectural partitioning, aggregation, and monitoring. Most of these tradeoffs can be expressed in analytic terms, which certainly helps, but some cannot" —Eberhardt Rechtin and Mark Maier

"design is the [..] structure or behavior of a system whose presence resolves or contributes to the resolution of a force or forces on that system. A design thus represents one point in a potential decision space."

—Grady Booch

"A culture is not a painted picture; it is a living process, composed of countless social interactions. Like a river whose form and velocity are determined by the balance of those forces that tend to make the water flow faster, and the friction that tends to make the water flow more slowly the cultural pattern of a people at a given time is maintained by a balance of counteracting forces."

—Kurt Lewin, Resolving Social Conflicts, 1948, p.46

Sources of Forces

"we build systems out of pure thought, in order to balance the static and dynamic forces of cost, schedule, functionality, performance, reliability, usability, and ethical implications"

– Grady Booch



TECHNICAL DECISIONS

Image source: Grady Booch

Sources of Forces

"We do not analyze requirements; we construct them from our own perspective. This perspective is affected by our personal priorities and values, by the methods we use as orientation aids, and by our interaction with others" — Christiane Floyd

'The word "requirements" represents a fundamental misunderstanding of software. They're theories, at best.' — Sarah Mei

Design Envelopes

In engineering, we contemplate, weigh, and experiment to find the boundaries of the design envelope.

"Hard" requirements tend to be areas where our design envelope has less "give", so other parts of the requirements design have to flex.

"The better you understand the problem, the closer you can design to tolerances." — Dana Bredemeyer

We innovate by pushing the design envelope — extending the range of possible, into the adjacent possible.

[with reference to the slide:] "Of course they are categories: each describing a class of forces. For example, compatibility encompasses pressures that arise from legacy, frameworks, and standards" — Grady Booch

"Architecture is the set of design decisions that provide a reasonably satisfying resolution to the static and dynamic forces on the system." — Grady Booch

There is a multidimensional decision *space.* We want to surface not just options, but assumptions about forces in play.

"the force field of a software project starts with Requirements. Requirements are often categorized in some way, like "functional" and "nonfunctional", or "user requirements" and "system requirements. However, requirements of any kind [..] contribute to shape the overall field."

— Carlo Pescio

Who: Is Impacted? Cares? Influences?

- Who is impacted?
- Who cares about this?
- Who influences the outcome?
- Who would care, if the outcome was different?



EXPLORE DDD CONFERENCE

Why Start With Who?

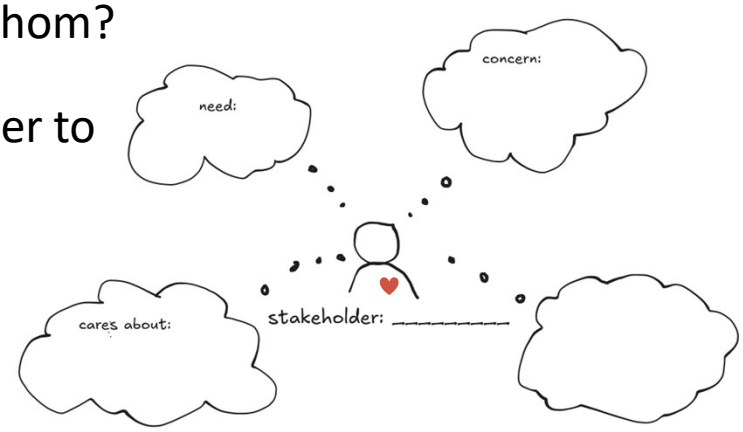
We are really starting with framing the matter we're addressing, and working to understand the situation that has given rise to a challenge, a stated outcome, a perceived need for a decision, etc.

And one way to navigate towards that understanding, is to begin to explore the situation in terms of who impacts, and is impacted by, has relevant knowledge or interest and energy about the situation, who has influence over what happens, and also, who would come into focus, if things were different and that raised their investment in the outcome. And so forth.

What problem are we actually trying to solve? That is where we need to start.

Cares and Concerns

- What matters, to whom?
- Why does this matter to them?



EXPLORE DDD CONFERENCE

Agendas, Knowledge, Power

The different stakeholders (impact and impacted by the system, even if indirectly) have orientations, knowledge, and abilities to effect the decision and its impact on the organization and the system we're build-evolving.

To understand the forces acting on the system, we draw on our own, and other's knowledge, understanding and experience. Who understands the market forces (now, and as they are likely to evolve)? Who understands the (causal) dynamics of system resilience, and the related forces and tradeoffs? What forces are relevant, and how, are matters of judgment, but the sources of expertise tend to be varied, and we need to seek them out.

"Power might be defined as simply the ability to make things happen, to be a causal agent, to initiate change. [..] Control might be defined as power exercised as means toward a specific end; authority, as vested control."

— Mary Parker Follett, Dynamic Admistration, Power (ch 4)

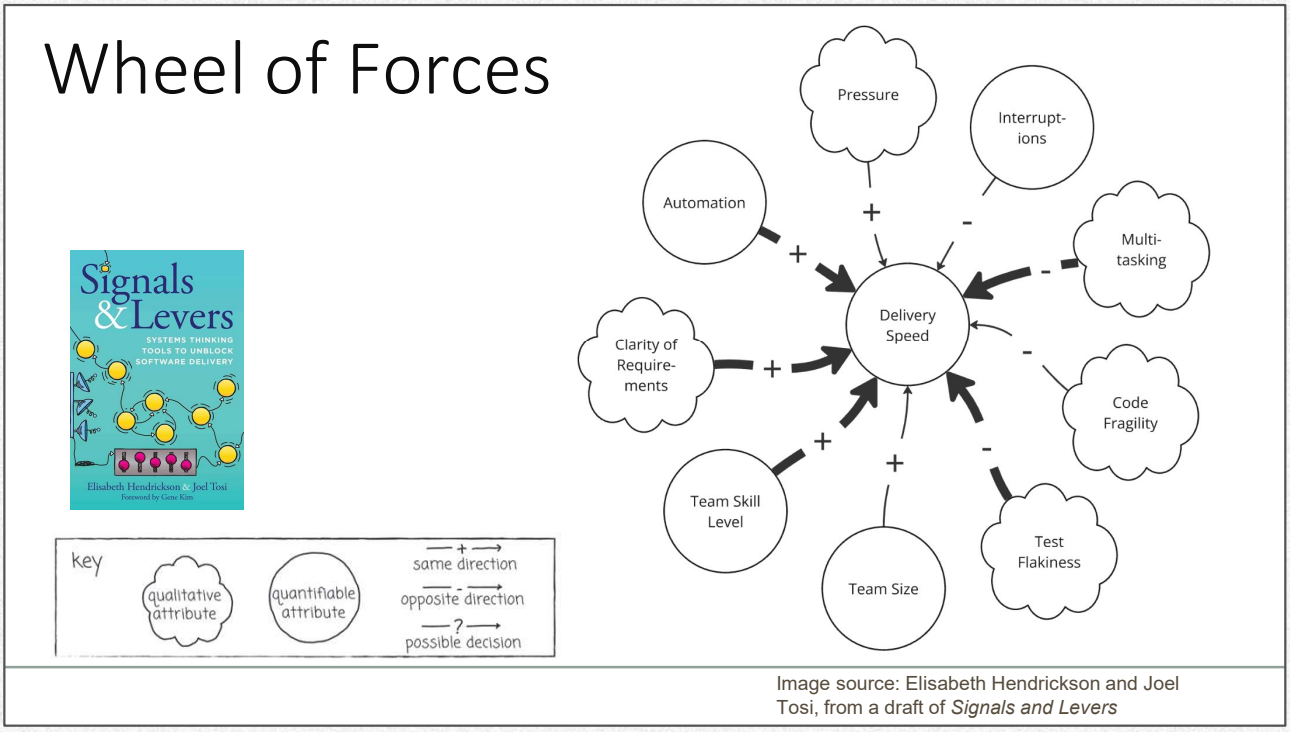
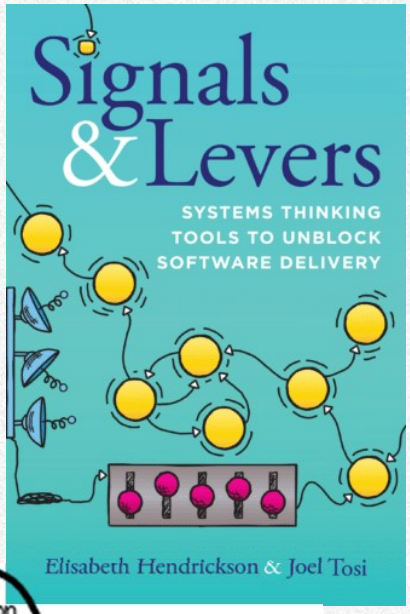


Image source: Elisabeth Hendrickson and Joel Tosi, from a draft of *Signals and Levers*

Forces Impacting Outcomes



In the Sense/Make Sense (Situation Awareness) section, we introduced causal loop diagrams (CLDs) as a useful way to gain an understanding — mapping out networks of causal influences and effects, and gaining an understanding of how the situation is being shaped. And develop theories of the situation that inform responses, including decisions and options to probe and try. A "wheel of forces" (Elisabeth Hendrickson and Joel Tosi) explores a property we're interested in, and what impacts that property (with strength of influence indicated by the thickness of the arrow). As with Causal Loop Diagrams, a + on the arrow indicates movement ion the same direction, so increasing the skill level in the team, influences delivery speed to increase. An increase in code fragility, influences delivery speed to go down (indicated by the – on the arrow).

A causal loop diagram gives us more to work with, in terms of understanding the influence network. Here we have a CLD from *Signals and Levers*, by Elisabeth Hendrickson and Joel Tosi, which is due out in September 2026.

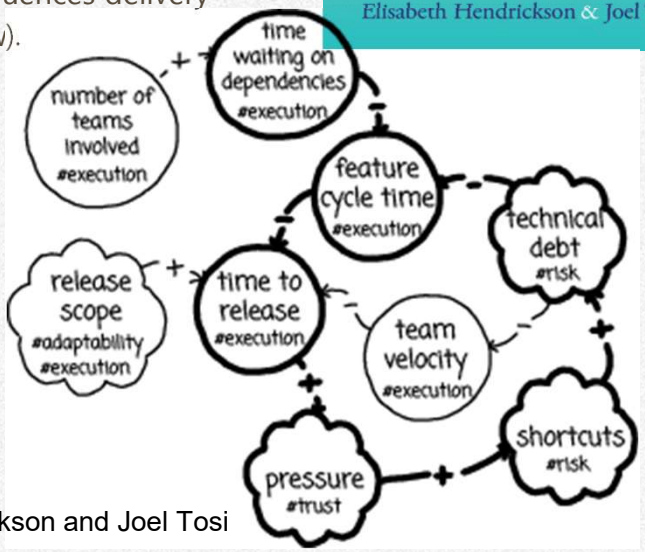
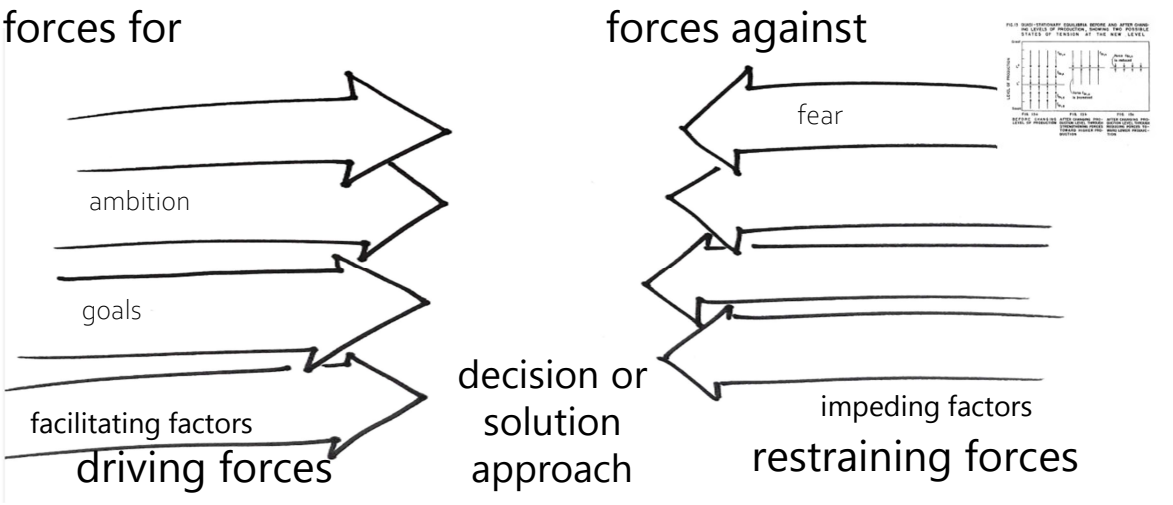


Image source: Elisabeth Hendrickson and Joel Tosi

Force Field Canvas

"AI usage is now a baseline expectation", Shopify CEO Tobi Lütke

adding force to try to shift



Adapted from: Gamestorming.com, by Dave Gray, et al

Force Field Analysis

Kurt Lewin did pioneering work in group dynamics, Action Research, and organizational development.

Of particular interest to us here, is Force Field Analysis, using Force Field Diagrams, developed by Kurt Lewin. Lewin was interested in group and organizational change or adaptation, and forces holding the organization in quasi-equilibrium. Force field analysis is useful in the context of organizational change, but can also help visualize forces that any decision balances or compromises across. "[Forces] tend to bring about locomotion or changes" (Lewin, Frontiers in Group Dynamics, 1947)

'According to Kurt Lewin "An issue is held in balance by the interaction of two opposing sets of forces - those seeking to promote change (driving forces) and those attempting to maintain the status quo (restraining forces)." Lewin viewed organizations as systems in which the present situation was not a static pattern, but a dynamic balance ("quasi-stationary equilibrium") of forces working in opposite directions. In order for any change to occur, the driving forces must exceed the restraining forces, thus shifting the equilibrium.

The Force Field Diagram is a model built on this idea that forces - persons, habits, customs, attitudes - both drive and restrain change.'

http://www.valuebasedmanagement.net/methods_lewin_force_field_analysis.html

"If you want truly to understand something, try to change it."

*— Kurt Lewin**

* this quote is attributed to Kurt Lewin by Charles Tolman in *Problems of Theoretical Psychology,*

"Any given change may be a positive for some people and a negative for others. Who benefits from they way things are now? Who will benefit from a change? Who will experience the negative space, and what will that negative be?

— Esther Derby

We're, in effect, investigating the failure boundaries of the design decision space

Forces in Dynamic Tension

Rasmussen's dynamic safety model describes the feasible operating space for a sociotechnical system.

Adapted here to explore interaction of code habitability and software habitability

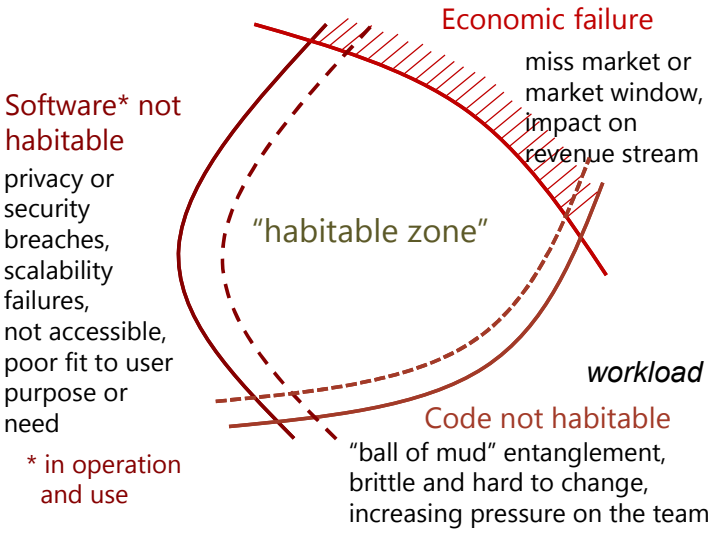


Image: Adapted from the Dynamic Safety Model presented in Cook and Rasmussen, 2005

Habitable Zone

The dynamic safety model was developed by Jens Rasmussen; adapted by Cook, Rasmussen, and others. It is described by Richard Cook in his presentation titled "Resilience In Complex Adaptive Systems" (Velocity 2013). This talk is available to watch on Youtube (under 19 minutes), and highly recommended.

We can combine the notion of habitable code and habitable software, adapting Rasmussen's dynamic safety model to design, to illustrate Rob Smallshire's point that "We're trying to find habitable zones in a large multidimensional space, in which we're forced to make regrettable, but necessary, tradeoffs." I'm not sure of the origin of the notion of code habitability, but it was Rebecca Wirfs-Brock who drew my attention to it. And in his keynote at OOPSLA in 1995, Christopher Alexander pleaded with our field to pay attention to the habitability of the software we create.

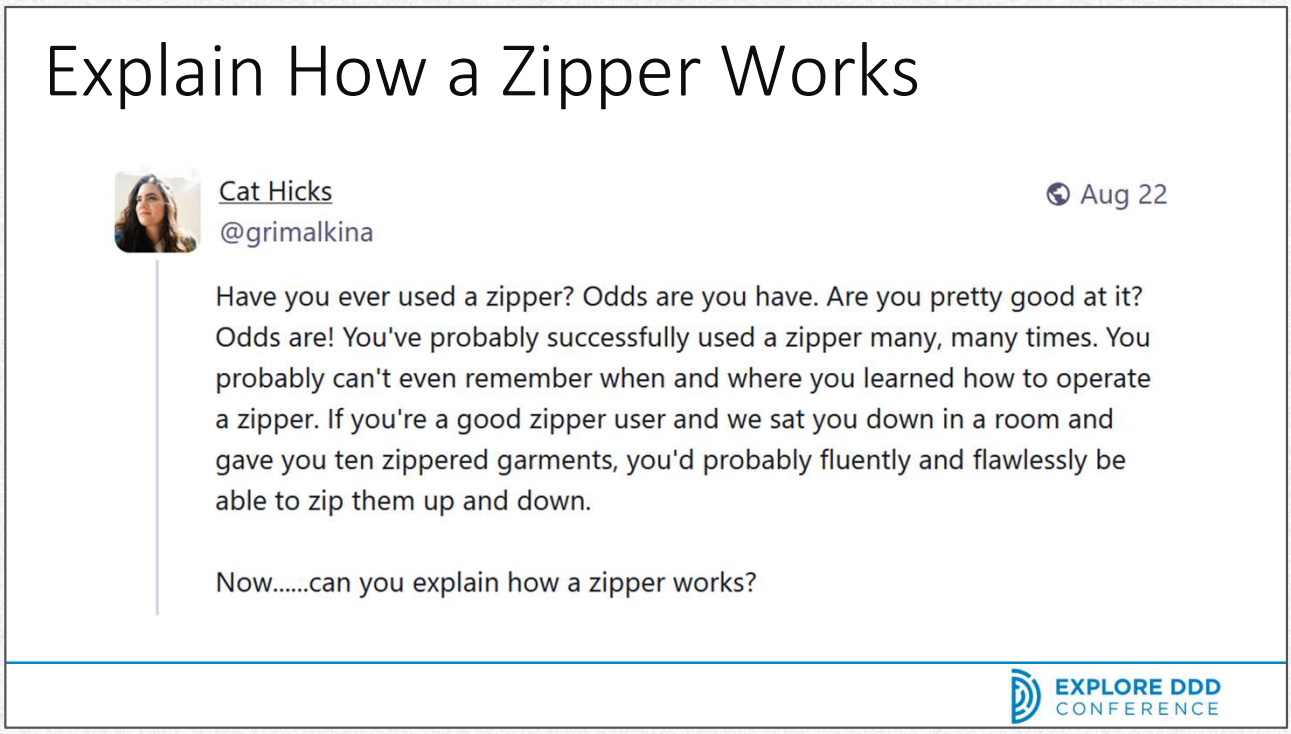
The idea that is being illustrated here is that if we push too hard to get features to market to stay away from the economic failure boundary, we may defer investments in code habitability and repair and in so doing increase developer decision fatigue and stress because of an overload of conceptual and decision burden with entangled code and hard to predict consequences of changing the code. But some of the things we do to keep the code habitable may also keep us away from failures on the boundary of operation and use.

"Most software architects do not think of themselves accounting for social issues, but that is one of the characteristics of good architecture. Accounting for social issues gives designers an easier life, which gives the software a longer life."

— Alistair Cockburn

"We're trying to find habitable zones in a large multidimensional space, in which we're forced to make regrettable, but necessary, tradeoffs."

— Robert Smallshire



Give it a try!

Spend a moment thinking about how a zipper works, and perhaps draw a diagram. Now explain it to others.

Cat Hicks: This is a charming, famous example that comes from cognitive science/psychology work on The Illusion of Explanatory Depth: "Most people feel they understand the world with far greater detail, coherence, and depth than they really do"

Cat Hicks: "But the illusion is something you can puncture! People's perceptions and awareness of their own knowledge drop quite sharply once they go through this explicit kind of exercise. Forcing yourself to truly reason about cause can make you aware of connections you're just assuming without really understanding."

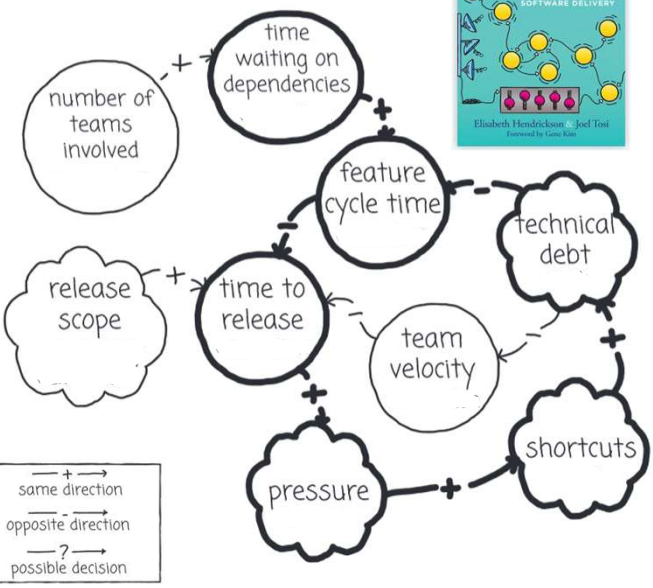
"The illusion for explanatory knowledge–knowledge that involves complex causal patterns—is separate from, and additive with, people's general overconfidence about their knowledge and skills."

— Leonid Rozenblit and Frank Keil, "The misunderstood limits of folk science: an illusion of explanatory depth"

Quote source: Rozenblit, L., & Keil, F. (2002). The misunderstood limits of folk science: An illusion of explanatory depth. Cognitive science, 26(5), 521-562.

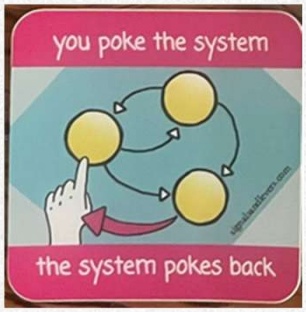
Explain Technical Debt

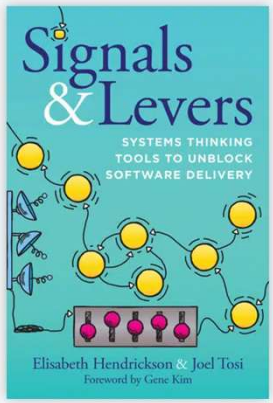
- We keep meaning not to let tech debt grow out of control, and yet…



Exploring Causal Interactions

CLDs (causal loop diagrams, and also stock and flow diagrams) are useful ways to make interacting causal dynamics visible — where forces are varied (market dynamics and incentives, organizational dynamics, individual dynamics, technical effects, etc.), but we can still seek to understand causal dynamics better by drawing out the causal relationships, and looking for feedback loops that amplify dynamics, or bring them into balance.



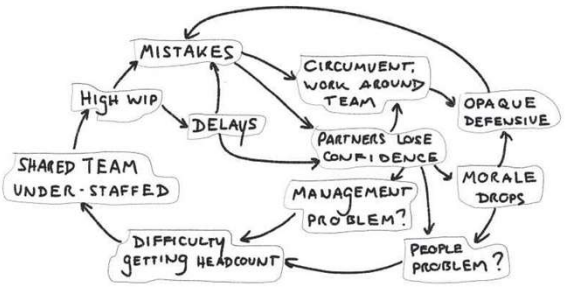


Sticker and book by Elisabeth Hendrickson and Joel Tosi

Exploring the System: The Narrative

"There is a stigma when one team is singled out as "the bottleneck". Teams have grown accustomed to working around the problem, and now you suddenly have to face the fact that a single team (or individual, or dynamic) has been rate-limiting the whole organization. There's a tendency to place blame, when meanwhile this poor team has been maxed out."

– John Cutler



source: John Cutler, https://medium.com/@johnpcutler/the-system-bites-back-bcf3f16b461

More John Cutler on WIP

John Cutler continues:

- Suddenly you'll have team members with "nothing to do" which is always an uncomfortable feeling. Sure you can address this with encouraging them to help other teams in need and/or "small stuff" queues but you often find resistance ("I wasn't hired to do that, I was hired to do my job!")
- High WIP situations sometimes devolve into the "10 teams of 5 are actually 50 teams of 1" problem···namely individual backlogs, and individuals maxed out. So, you aren't dealing with 10 teams working things. You are dealing with 50 individuals.
- Countless structures and rituals exist to manage all of that work in progress. Consider the management functions that have thrived because there's a need to figure out how to keep utilization high···to nail down estimates, have long meetings to juggle dependencies, and to "protect" teams because they are so busy. Reducing WIP changes the game.
- You are changing the focus from local optimization to a more global perspective. Unearthing the real dependencies in a system — especially if teams have kept the veneer of autonomy — is hard.

More at: https://medium.com/@johnpcutler/the-system-bites-back-bcf3f16b461

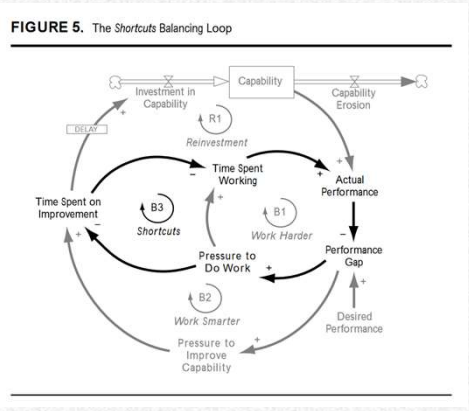


Figure from: Nobody Ever Gets Credit for Fixing Problems that Never Happened https://web.mit.edu/nelsonr/www/Repenning=Sterman_CMR_su01_.pdf



Donella Meadows teaching Causal Loop Diagramming, https://www.youtube.com/watch?v=XL_lOoomRTA

High WIP, by John Cutler

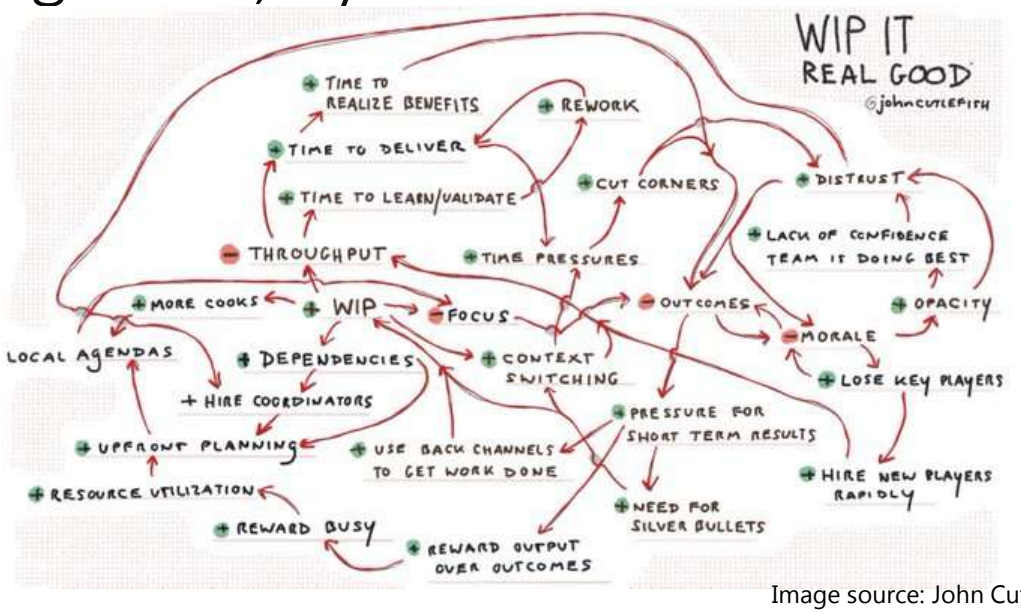


Image source: John Cutler

System Dynamics and WIP

John Cutler uses the system map (slide) to explore the dynamics, and explore typical interventions intended to lower WIP.

"You have built that system. And now try to lower WIP. What happens? The well-meaning, passionate, and intelligent people in the system have crafted roles, interactions, structures, egos, and relationships around doing a lot at once, and that's not something you can simply rationalize away. The system will push back (and often for very real, painful reasons). The same thing happens to individuals who attempt to change a habit.

What are you up against? Below I point out some areas of optimization that will no longer "make sense" (and will fight back) with efforts to reduce WIP. To improve the situation, you'll need to address the current coping mechanism(s), and that will not be easy. It's easy to single out individuals for being "change adverse", but these challenges are mostly systemic, and far from trivial."

It is well worth a read!

Source: John Cutler, "WIP It Real Good," https://cutle.fish/blog/wip-it-real-good

Another System Mapping resource:
Systems Mapping - a brief overview of what, why and how (Part 1)
mojdigital.blog.gov.uk/2020/06/02/systems-mapping-a-brief-overview-of-what-why-and-how-part-1/

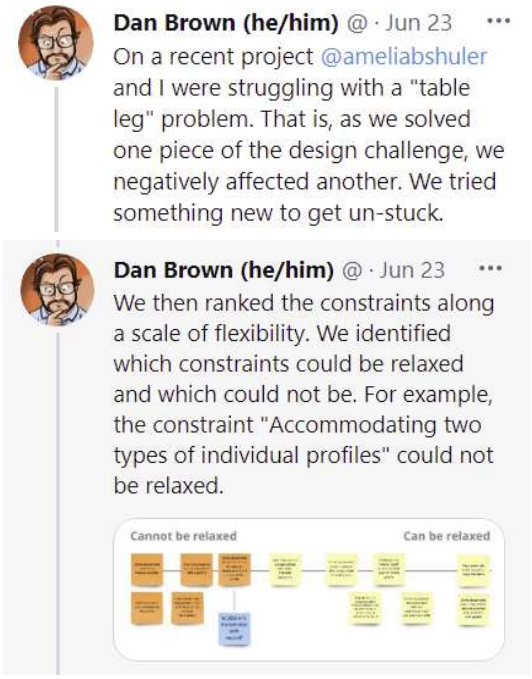
"Handoffs help keep people busy. While Person A is working on something, Person B can start on an upstream task, and Person C can work on a downstream task. Now they could work together, but that would be (seemingly) inefficient, and may require new tools. So a culture of handoffs emerges, and you begin to see folks actually looking for opportunities to "get ahead" of things. Handoffs require coordination and meetings"

— John Cutler

Constraints

1. Make a list of constraints
2. Rank constraints by flexibility
3. Evaluate design concepts by how many constraints they address
4. Discuss the right balance of constraints
5. Sketch ideas based on that balance

— Dan Brown



Constraints and the Design Envelope

Engineers have a concept of a design envelope or design space that is created by constraints – outside the design envelope, the design is (technically) infeasible or (economically or socially) not viable.

The concept of a Pareto frontier is useful, not because we know (in general) where this frontier lies exactly (though we find out when we cross it), but because it reminds us we're working in a space of interacting decisions and constraints – some of which may only "bite" (factor crucially) at some point.

Pareto Frontier: "The Pareto frontier is the set of all Pareto efficient allocations, conventionally shown graphically. ... It is a statement of impossibility of improving one variable without harming other variables in the subject of multi-objective optimization (also termed Pareto optimization)." (Source: wikipedia)

"*Pareto efficiency* or Pareto optimality is a situation where no individual or preference criterion can be better off without making at least one individual or preference criterion worse off or without any loss thereof." (Source: wikipedia)

We're, in effect, investigating the failure boundaries of the design decision space

Over Time

'I've used 100% stacked area graphs to visualize tradeoffs or strategic allocation of "fixed" resources, where the allocation changes over time.'

— Juno Suárez

'I tend to use "graphic equalizer" with scaler x-axis so that you can overlay to compare and contrast snapshot values for variables for trade off.'

— Dawn Ahukanna

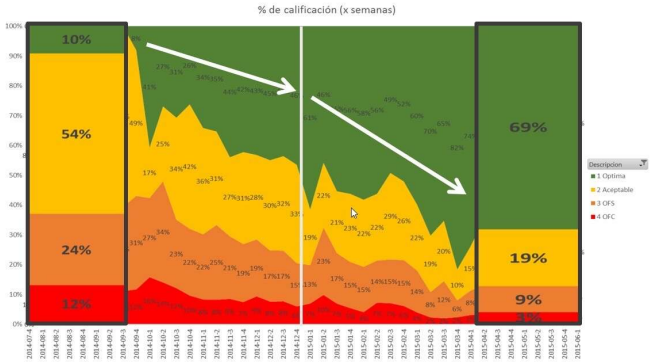


Image source: Juno Suárez, https://hachyderm.io/@juno/110945173941162351

Where the Forces Change over Time

I asked folk on mastodon what other visual forms they use to bring tradeoffs into view.

Dawn Ahukanna pointed out that many of our representations tend to be at a point in time. Dawn suggested: ""instance in time" snapshot metric for contrast and comparison with other snapshots. It's like taking a time-lapsed set of photographs/ sampling of a specific spot and turning it into an motion interaction where you can "pan through time."'

Peter Gassner pointed to a neat prototype they developed for visualizing project constraints and dependencies:





https://lab.interactivethings.com/confluence-diagram/#/

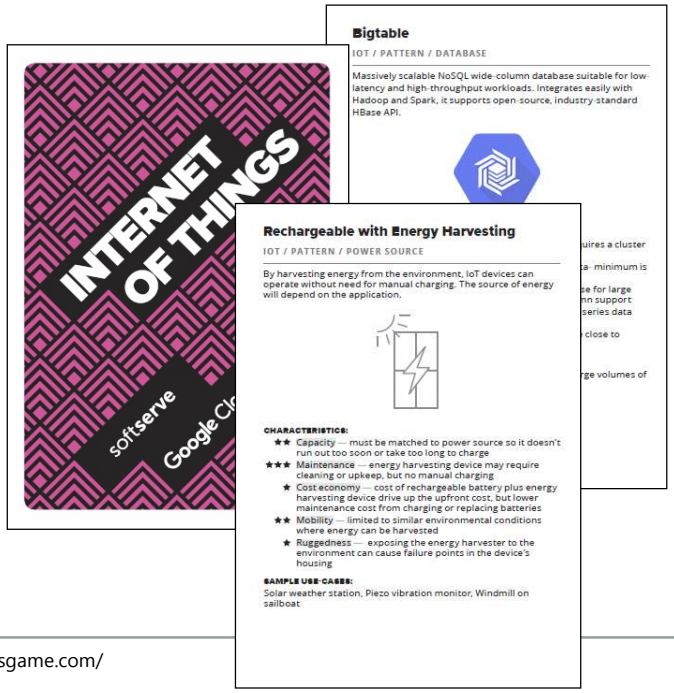
And James Fairbairn: "I ask people more and more these days about their theory of change — like, understanding the complexity of this space, and how everything is a chain of causes and conditions, let's walk through how we think "X leads to Y" *actually* works…"

"Eg: on a platform team driving an enterprise technology migration, focusing time between focus areas like maintenance, new development, and support/training/customer success. Conditions and opportunities change over the migration lifecycle (adoption curve), and capturing these requires tradeoffs of team attention."

— Juno Suárez

Smart Decisions

The Smart Decisions Game highlights the tradeoffs inherent in each decision and across decisions



Images from: Smart Decisions Game site: https://smartdecisionsgame.com/

The Smart Decisions Game

"Smart Decisions is a game that simulates the design process of software systems and promotes learning about it in a fun way." -- from the Smart Decisions Game website; but having played the game at SATURN, I agree. The game can be downloaded, and used in a team learning activity.

It's a good way to highlight for the team that each technology and related decision has its strengths and weaknesses, and architectures are not just about individual decisions, but weighing across the decisions for a fit to the context and purpose of the system. Further, there will not always be agreement on the approach to take, because the nature of tradeoffs is that they entail judgment about the strengths/weakness as well as the value of the outcomes, and the degree to which the consequences (in other areas of the system, or its containing systems) need to be taken into account.

The SEI team has done important work in the system qualities and trade-offs space, including developing the Architecture Tradeoff Analysis Method:

"ATAM gets its name because it not only reveals how well an architecture satisfies particular quality goals, but it also provides insight into how those quality goals interact—how they trade off against each other"

Judgment Factors

We may notice where we're being constrained (that's where we've hit a point of tension in the tradeoff space). But discerning tradeoffs is very much a matter of experience and judgment.

"Because the situation is ill-structured, the goal cannot be optimization. The architect seeks satisfactory and feasible problem-solution pairs."
— Mark Maier and Eb Rechtin

Smart Decisions Game site: https://smartdecisionsgame.com/

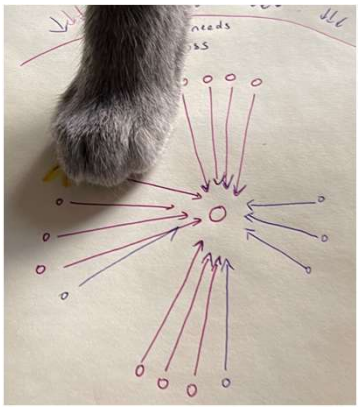
Acknowledging that it's hard; there's risk and uncertainty, and uneven willingness to make hard calls

Real Talk

What (really) shapes this decision space?

- What are we avoiding (talking about)?
- What consequences are in "don't go there" zones?
- What forces feel too career-dangerous to write down?

Besides, we're addressing future impacts that are uncertain



… pawlitical forces…

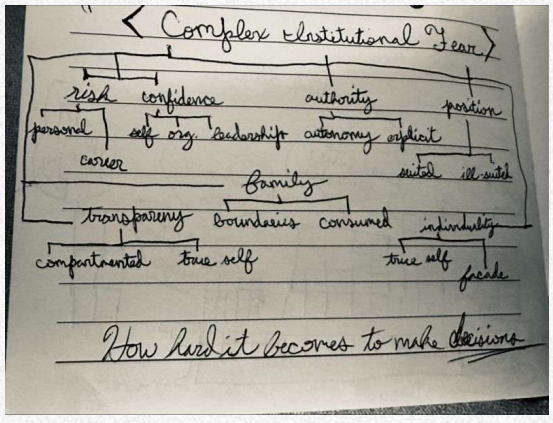
When Consequences have Consequences

Significant decisions impact the paths or options we have, but also change the possibility space of some areas of system context or environment. They create paths, and close off paths, for ourselves, for users, for others impacted. Decisions and situations have reciprocal effects on one another; design acts back as consequences, that we (may) take into account in making the decision. When this has to do with action now, and possible future consequences, it may cause indecision, or be costly, in organizational terms, to probe and discuss openly. There are no pat answers here. The culture of the organization overall, and the part of the organization involved, plays a role. We can point to the importance of psychological (or social) safety in creating a learning environment where implications can be probed, and responded to together. And we're weighing positive outcomes (intended direct effects, and as side effects or positive externalities) along with negative. In the context of uncertainty. Sometimes avoiding real talk may be about uncertainty/ambiguity or conflict avoidance, but restricting the consideration space may be due to decisions made elsewhere… Similar to learning from incidents, we need to be able to seek even conflicting perspectives, and explore options and impacts, and feed that learning back into the decision. While being pragmatic about uncertainty and the need to make decisions.

Part of what makes leadership and experience important here, is the willingness and ability to discern and take on these kinds of organizational challenges, and navigating them. (Caveats apply; alternately put: there is more to say, or nuance to add.)

*decision (n.) from decidere "to decide, determine," literally "to cut off," from de "off" (see de-) + caedere "to cut" (from PIE root *kae-id- "to strike")*

etymonline.com



https://www.linkedin.com/posts/christopherwilliams2018_complexsystems-journals-systemsexploring-activity-7159612234229301248-l4ip/

That's … a Lot … so

How do we clarify the situation and identify forces?

— modeling, canvases and structured conversations

How do we design and compare alternatives?

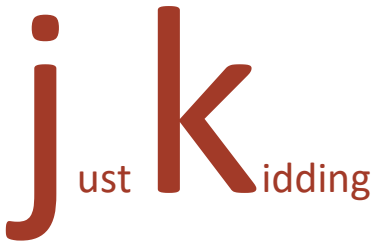
— modeling, canvases and structured conversations

How do we reach a decision?

— modeling, canvases and structured conversations

How do we build understanding and buy-in?

— modeling, canvases and structured conversations



(but also not)

How We Work is Part of the Work

We've covered a fair bit of conceptual ground. The "what" of the Architecture (or other strategically significant) Decision Record indicates areas of work that are separable but intertwined. There's exploring the context or situation (sometimes this goes by "the problem"), with an emphasis on forces (or criteria) so that when we evaluate alternative approaches ("solutions") we can identify tradeoffs (identifying pains or costs we incur for what gains) and consider approaches against the desiderata we've established. But how?

Ideally, we do this in a collaborative way, together with those who have perspectives and experience that inform our understanding of the situation, its challenges, and options. An informal session at a white board is generative, but canvases (such as the Force Field canvas from *Gamestorming* on the next page) and diagrams focus the discussion, while also drawing attention to areas the discussion might otherwise avoid or neglect. It's good to have them in the mix. It also builds a deeper understanding of the decision than one made in a "hub-and-spoke" way, where one person is the main center of thinking about the decision and puts ideas out for response. How we work, can get a good part of the larger work done, if we're strengthening the decision and building understanding and "buy-in" organically.

"You cannot coordinate purpose without developing purpose, it is part of the same process." — Mary Parker Follett

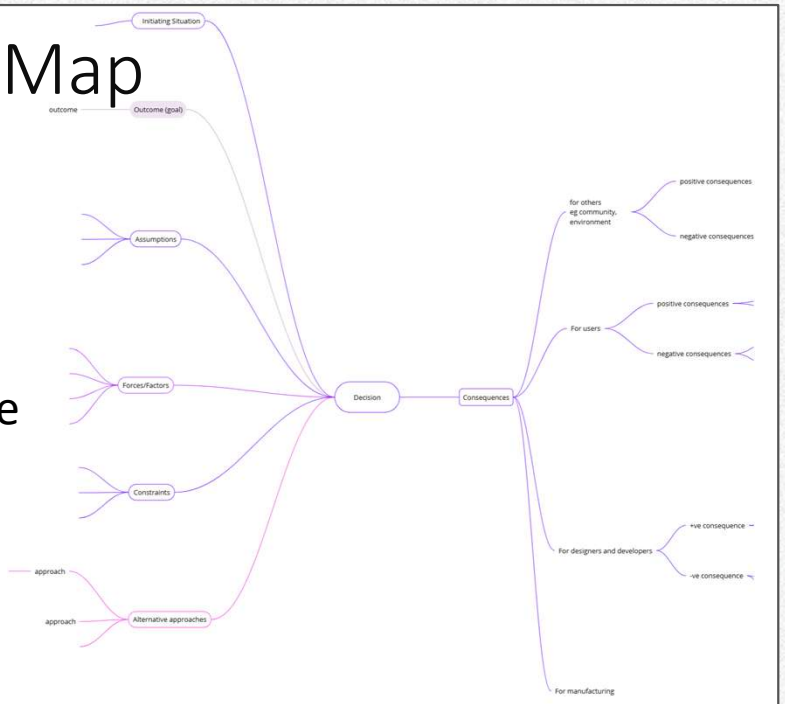
"I get it. Meeting culture sucks. It's too easy for people to thoughtlessly take each others' time, occupy standing slots, show off with performative teamwork, and generally suck your energy. Meetings feel like dead time. Meetings are time spent with people yet strangely devoid of social gratification. Meetings typically bore most participants — the greatest sin in knowledge work — and when they're over, nothing has changed except us all being that much closer to retirement. [..]

*But what if, hear me out, what if the *only* work that matters in a knowledge economy happens when we are together?."*

*— Elizabeth Ayer, Meetings *are* the work*

Decision Mind Map

The decision template is itself a great structure for conversation. Here, the template is in "mind map" like form



Decision Template as Mind Map

Using the decision template in textual form, or mind map* form, encourages attention to the different facets of the decision. Starting with the context (or situation) and sought outcome, and identifying forces, constraints, assumptions, before turning to alternatives and describing options or design ideas. The idea with the mind map, is to tease out — adding tendrils and following threads, exploring down a path. By having the emerging picture on a whiteboard or (miro, etc.) frame, we're encouraged to add relevant detail to other areas of the map, whenever such a detail emerges in the course of the conversation. For example, if we notice we're making some assumptions while we explore forces or alternatives, we add those in. It is just as well to notice that as we explore the decision space, the outcome may come into clearer focus (and even shift, as we understand the problem better). As we explore consequences, we might find ourselves revisiting alternatives and exploring trade-offs and consequences further. The "how" is non-linear. We document the decision so that it reads in a way that conveys clarity. But getting to clarity means some holding space for ambiguity that uncertainty and complex interactions kicks up.

* Mind Maps were popularized by Tony Buzan. Simon Wardley protests such a use of the word "map." Perhaps we can call it a Decision Root Ball (haha).

"Our job [..] how to devise methods by which we can best discover the order integral to a particular situation."

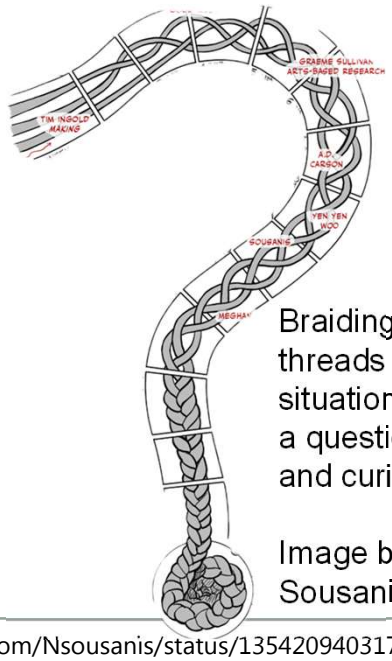
— Mary Parker Follett

"When you truly understand the problem and when you know what the system must be, what it must do and what you are willing to trade away, the choice becomes obvious. It doesn't feel like guessing. It feels like reading. The work is not in the decision. The work is in everything that comes before it." — Karol Wiszowaty

Yes But…

A conversation helps us get ideas on the table, but…

… how do we create and assess options?



Braiding the threads of the situation into a question and curiosity

Image by Nick Sousanis

Image from: https://twitter.com/Nsousanis/status/1354209403172319234

Options

The following pages identify some of the ways we envisage and explore options.

Aside: Watch for "yes, but" because it indicates that folk still have concerns, and ideas about things. Sometimes⋯ we need to move forward, and give them opportunities to learn more as we go. Sometimes⋯ we need to learn more about the consequences and obstacles that they are seeing, and giving us a heads up about with "yes, but." Goodwill, like trust, takes time to build, and can be damaged in ways that are harder to rebuild from. So it's good to not project our beliefs and conclusions, and even stereotypes, onto those who are not 'just coming along"⋯

*"The architect thinks it is their job to arrive with the answer. Sometimes it is. But sometimes **the job is to arrive with the question**, clearly stated, so the people who will live with the consequences can make the call."*

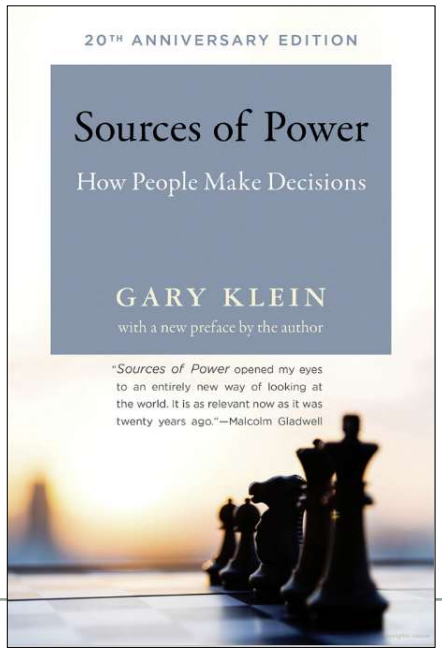
— Karol Wiszowaty

Klee said: art does not render what is visible, but renders visible (via Dan Klyn).
Visual design (diagrams, models, descriptions) renders visual.
Puts thought into the world.

How Decisions are Made

"the sources of power that are needed in natural settings are usually not analytical at all—the power of intuition, mental simulation, metaphor and story telling"

— Gary Klein



"In many cases, the problem isn't about having or noticing insights; it is about acting on them. The organization lacks the willpower to make changes."

—Gary Klein

Sources of Power

Gary Klein (*Sources of Power: How People Make Decisions*) studied decision making in settings that he characterizes as naturalistic decision making:

> "Features that help define a naturalistic decision-making setting are time pressure, high stakes, experienced decision making, inadequate information, ill-defined goals, poorly defined goals and procedures, and dynamic conditions."

Here, rather than deductive analysis and statistical methods, other "powers" were used:

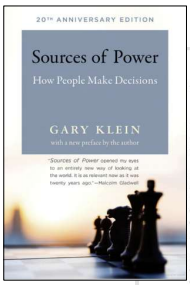
> "The power of intuition enables us to size up a situation quickly. The power of mental simulation lets us imagine how a course of action might be carried out. The power of metaphor lets us draw on our experience by suggesting parallels between the current situation and something else we come across. The power of storytelling helps us consolidate our experiences to make them available in the future, either to ourselves or others."

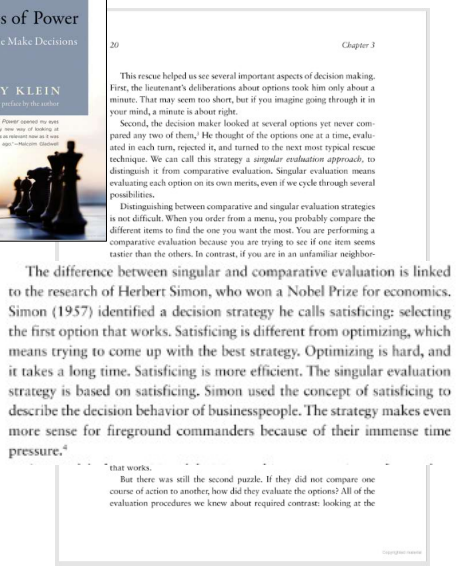
Experience lets us recognize situations and identify actions to take, intuitively

Experience

"Their experience let them identify a reasonable reaction as the first one they considered, so they did not bother thinking of others. They were not being perverse. They were being skillful."

– Gary Klein




20 Chapter 3

This rescue helped us see several important aspects of decision making. First, the lieutenant's deliberations about options took him only about a minute. That may seem too short, but if you imagine going through it in your mind, a minute is about right.

Second, the decision maker looked at several options yet never compared any two of them.[3] He thought of the options one at a time, evaluated in each turn, rejected it, and turned to the next most typical rescue technique. We can call this strategy a *singular evaluation approach*, to distinguish it from comparative evaluation. Singular evaluation means evaluating each option on its own merits, even if we cycle through several possibilities.

Distinguishing between comparative and singular evaluation strategies is not difficult. When you order from a menu, you probably compare the different items to find the one you want the most. You are performing a comparative evaluation because you are trying to see if one item seems tastier than the others. In contrast, if you are in an unfamiliar neighbor-

The difference between singular and comparative evaluation is linked to the research of Herbert Simon, who won a Nobel Prize for economics. Simon (1957) identified a decision strategy he calls satisficing: selecting the first option that works. Satisficing is different from optimizing, which means trying to come up with the best strategy. Optimizing is hard, and it takes a long time. Satisficing is more efficient. The singular evaluation strategy is based on satisficing. Simon used the concept of satisficing to describe the decision behavior of businesspeople. The strategy makes even more sense for fireground commanders because of their immense time pressure.[4]

that works.

But there was still the second puzzle. If they did not compare one course of action to another, how did they evaluate the options? All of the evaluation procedures we knew about required contrast: looking at the

TECHNICAL DECISIONS

Experts Tend to Make Good (Enough) Decisions, Based on Experience

"The standard advice for making better decisions is to identify all the relevant options, define all the important evaluation criteria, weight the importance of each evaluation criterion, evaluate each option on each criterion, tabulate the results, and select the winner. In one form or another, this paradigm finds its way into training programs the world over. Again and again, the message is repeated: careful analysis is good, incomplete analysis is bad. And again and again, the message is ignored; trainees listen dutifully, then go out of the classes and act on the first option they think of. The reasons are clear. First, the rigorous, analytical approach cannot be used in most natural settings. Second, the recognitional strategies that take advantage of experience are generally successful, not as a substitute for the analytical methods, but as an improvement on them. The analytical methods are not the ideal; they are the fallback for those without enough experience to know what to do."

"*Intuition* depends on the use of experience to recognize key patterns that indicate the dynamics of the situation. This is one basis for what we call intuition: recognizing things without knowing how we do the recognizing." "If you want people to size up situations quickly and accurately, you need to expand their experience base."

Satisficing: "selecting the first option that works. Satisficing is different from optimizing, which means trying to come up with the best strategy. Optimizing is hard, and it takes a long time. Satisficing is more efficient." — Gary Klein, *Sources of Power*

"decision makers can satisfice either by finding optimum solutions for a simplified world, or by finding satisfactory solutions for a more realistic world"
*— Herbert Simon**

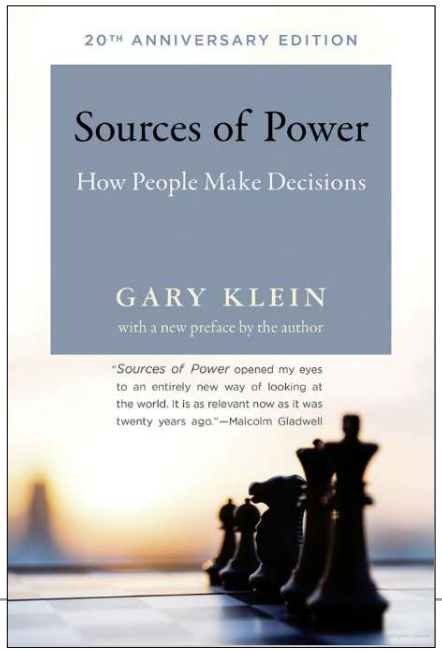
* in his Nobel Prize in Economics speech

Analogy and Metaphor

"Metaphor does more than adorn our thinking. It structures our thinking. It conditions our sympathies and emotional reactions. It helps us achieve situation awareness. It governs the evidence we consider salient and the outcomes we elect to pursue [..]

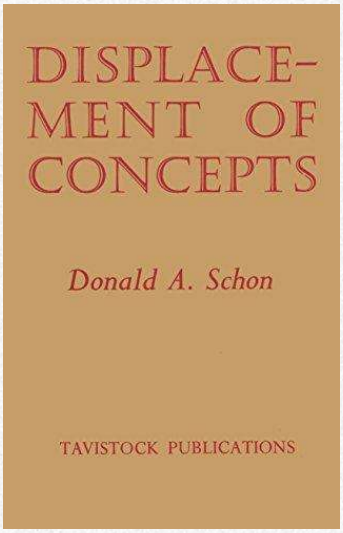
Analogical reasoning can also suggest options. "

— Gary Klein



"Analogues provide the problem solver with a recommendation about what to do."

— Gary Klein



Using Analogies To Solve Problems

"If we did not want to use analogical reasoning for tasks like these, we would be stuck. We would not know enough to construct formulas or to use them or have enough hard information to proceed. By using analogues, we are tapping into the same source of power for stories. We are applying an informal experiment, using a prior case with a known outcome and a semi-known set of causes to make predictions about a new case."

"First, we learned that they do not select analogues just based on similarity. [..] You would select an analogue that shares the same dynamics [..] If you do not have enough experience to take causal factors into account, you can get into trouble. The engineers we studied were all knowledgeable.

Second, we learned that some causal factors are easy to adjust for, and others are not.

Third, we learned that the logic of reasoning by analogy is similar to the logic of an experiment: to draw a conclusion without having to know all of the important factors operating."

— Gary Klein, *Sources of Power*

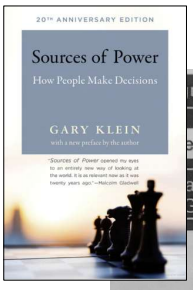
Analogies help us shape the problem (what we're addressing, and how we conceive of it) and get ideas for solutions (how we approach it).

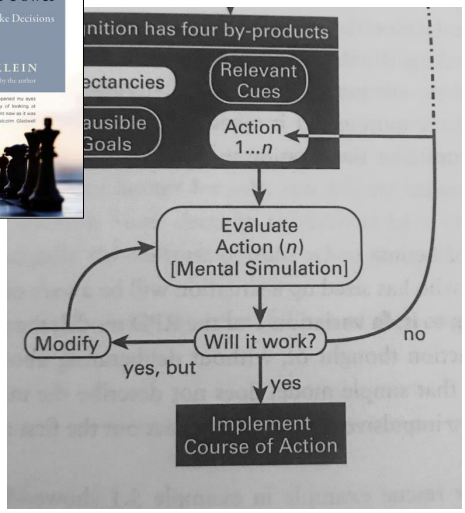
Mental Simulation

'In the middle of the meeting, the man stood up, walked over to the door, and closed it. Then in a hushed voice he said, "To be a good fireground commander, you need to have a rich fantasy life."

He was referring to the ability to use the imagination.'

– Gary Klein





TECHNICAL DECISIONS

Image: from *Sources of Power*

Thought Experiments

The quote in the slide above, is in the introductory paragraph in Gary Klein's chapter on mental simulation (in *Sources of Power*). It is a reminder that we tend too much to treat soft skills, and imagination in particular, as less than professional – less than "rational" or "objective" reasoning. Klein reminds us that we play out scenarios and alternatives in mind, to understand, to discover, to decide on courses of action.

Foresight is the application of imagination, of anticipating. I used to annoy my kids (don't judge me; repetition to the point of absurdity is the stuff of humor) when they'd say "I didn't mean to" and I'd respond "You've got to mean not to" – meaning, we need to try to anticipate the likely or even the possible, when it has bad consequences. Foresight is not a direct application of hindsight or learning from the past, but a willingness to take the risk of playing threads of the present forward, staying creative under uncertainty. Experience is valuable in giving us practice in recognizing cues and applying "muscle-memory" and tested-through-trial approaches, as well as in giving us the ability to anticipate, to "look ahead" and "look around" in an imaginative playing out of features and forces in a design or (other) decision moment. *Project premortems* (Gary Klein, Performing a Project Premortem, HBR, 2007; also Gary Klein, The Pre-Mortem Method, Psychology Today, 2021) asks us to imagine, during design, say, that a project *has* gone wrong, and to explore the reasons.

"*Code wins arguments*" (from Zuckerberg's "Hacker Way" letter to investors included in Facebook's IPO filing). Sure, but are all arguments worth having? Out beyond not valuing design/anticipation/etc. and not valuing making stuff, there is a field... (apologies to Rumi, etc.)

"To me, the real challenge is getting teams to slow down for a moment and think about what's going to be built, why, what the risks are, and what might change." — Phillip Johnston

"mental anticipation.. pulls the future into the present"

— Erich Jantsch

Try This

- Consider the following problem
  - One morning, exactly at 8 A.M., a monk began to climb a tall mountain. The narrow path, no more than a foot or two wide, spiraled around the mountain to a glittering temple at the summit. The monk ascended the path at varying rates of speed, stopping many times along the way to rest and to eat the dried fruit he carried with him. He reached the temple precisely at 8 P.M.

    The next day, he began his journey back along the same path, starting at 8A.M. and again walking at varying speeds with many pauses along the way. He reached the bottom at precisely 8 P.M.
  - I assert that there is at least one spot along the path the monk occupied at precisely the same time of day on both trips.
  - Is my assertion true? How do you decide?

TECHNICAL DECISIONS

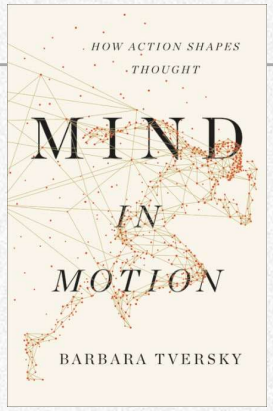
Source:

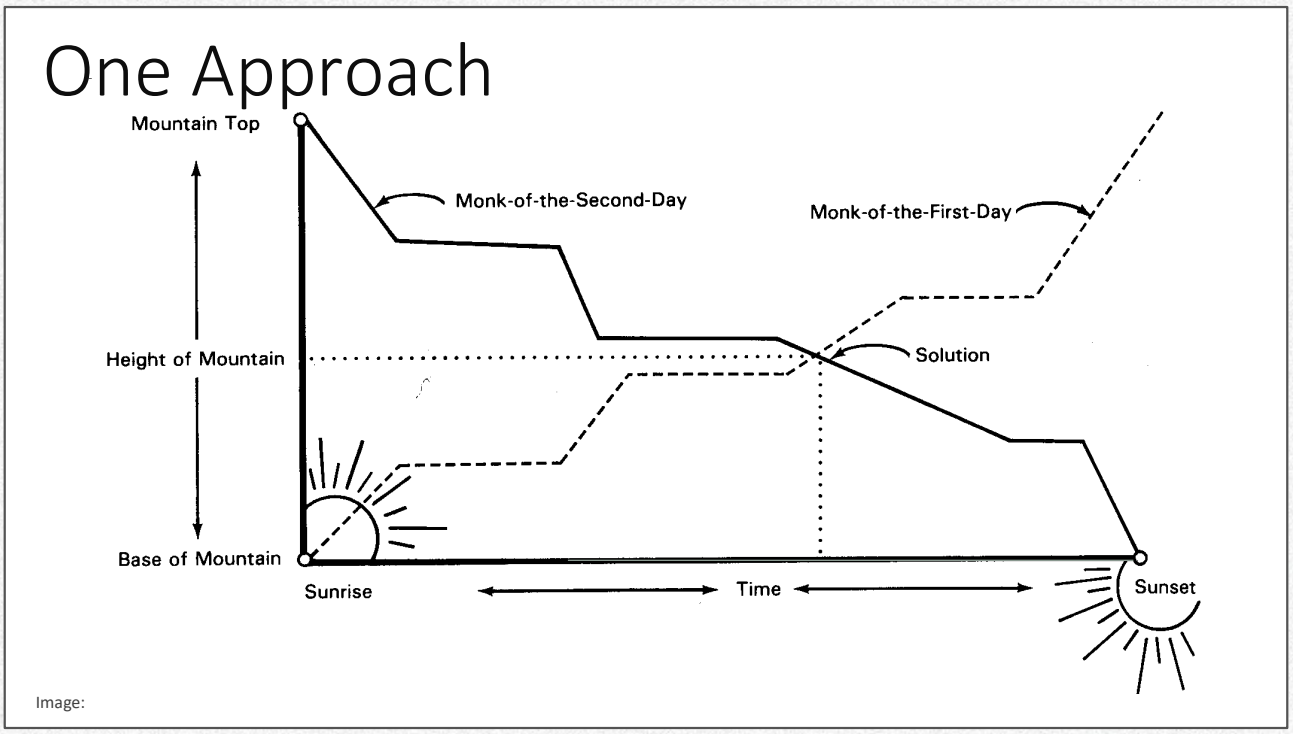
Exercise: Read the slide

And don't turn the page until you've had a chance to think about it.

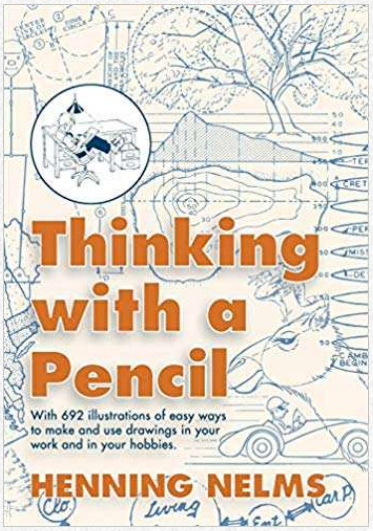
"Principle of Correspondence: The content and form of the representation should match the content and form of the targeted concepts"

— Barbara Tversky





(Mental) Simulation Illustration

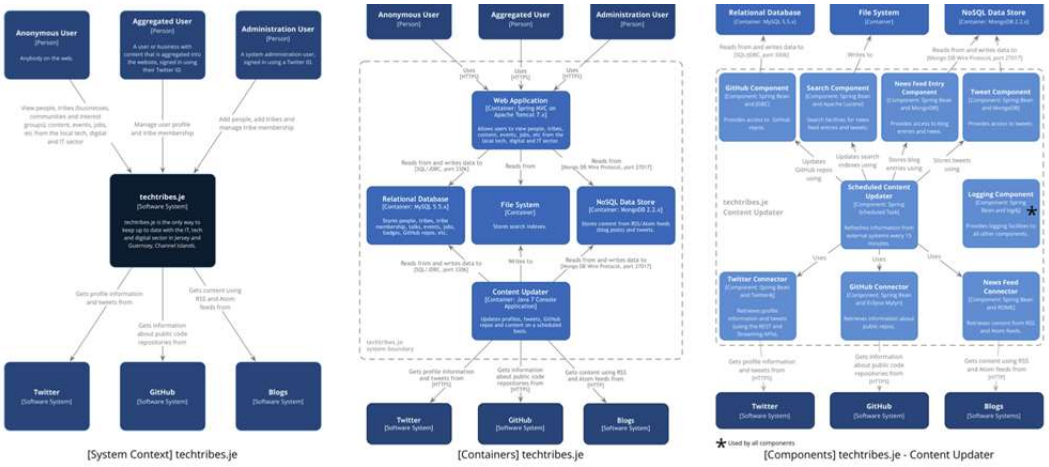


One way to think, is to draw, and the diagram illustrates that we can say yes. Another way to think about it is each of my hands is the monk on the two days, and one hand will move along the path in one direction, and the other hand is the monk starting at the other end of the path, and moving in the other direction on the same path. My hands have to meet at some point, at the same time. As important as the illustrations are to the point that we can put something in the world to help us think, it's also illuminating that some people will still not see it, and these people are important too. We can try to illuminate the solution different ways, but our perspectives differ, we're looking for a catch, and trust and credibility may factor, etc.

"We have misfiled the significance of drawing because we see it as a professional skill instead of a personal capacity [..] This essential confusion has stunted our understanding of drawing and kept it from being seen as a tool for learning above all else." — D.B. Dowd

---

Modeling and Sketchprototypes



TECHNICAL DECISIONS

Image: Simon Brown's C4 Model https://c4model.com/

Thought Experiments, Sketchprototypes, and Heuristics

Whether you're using an ad hoc approach, or *Visual Architecting* with UML and/or C4 (from Simon Brown), or something else, diagrams, models, views of the system, are ways to explore "decisions in formation" — sets of related decisions, as well as formative ideas — to probe and assess them.

We take a guess as a starting point, and improve on it: model, and run thought experiments across it. For example, take use cases or user stories or focus on one property, then another, etc., and "run" (imagine and talk through) behavior across the structure models, to flush out component responsibilities we overlooked in our initial guess. Lists of responsibilities (for elements of a system — technical, strategic or organizational) are a powerful and largely overlooked/under used tool in the architect's toolbelt. If the responsibilities don't cohere within an overarching responsibility, or purpose, that should trip the architect's boundary bleed detectors. Interactions at the boundaries are essential to making a system more than the sum of its parts, but introduce coupling and (inter)dependencies.

As we do this exploration with the aid of models (just as we do when doing design in the medium of code), we're applying heuristics we've developed through experience, and exposure to other people's work (books, and such). Heuristics don't take away the need to think, to reason and try things out. They help us identify what to think about, as we do so, and may suggest how to go about it (better).

"Heuristics offer plausible approaches to solving problems, not infallible ones." — Rebecca Wirfs-Brock

To illustrate, let's turn to Parnas and his criteria (heuristics) for decomposing, and hence coping with complex systems despite our bounded rationality:

> "[begin] with a list of difficult design decisions [..] Each module is then designed to hide such a decision from the others ."

Be deliberate and deliberate all the things"
— Dawn Ahukanna

Experiment (on paper, too)

Models help us try out or test our ideas — in an exploratory way when they are just sketches, and thought experiments, where we "animate" the models in mind and in conversation. Just sketches, so less is invested. Less ego. Less time.



Image: by me, with apologies to Escher's Reptiles

TECHNICAL DECISIONS

Experiment, On Paper Too

Fred Brooks wrote "Plan to throw one away. You will, anyway." I'd say: that too, but plan to throw several away — on paper. It's quick and cheap. We sketch-prototype to come up with alternatives and try them out in the cheapest medium that fits what we're trying to understand and improve. We seek to probe, to learn, to verify the efficacy of the design elements we're considering, under multiple simultaneous demands. We acknowledge we can misperceive and deceive ourselves, and hold our work to scrutiny, seeing it from different perspectives, from different vantage points but also with different demands in mind. We make trade-offs and judgment calls. We bring in others with fresh perspective to help us find flaws. We simulate. We figure out what to probe further, what to build and instrument.

We need to come up with and try out alternatives in the cheapest medium we can learn more in; sometimes that's code, but not if a sketch will do. We don't learn what we learn in the medium of code, but we can at least start to try ideas out, and explore and bat at them, investigate how they could work, in sketch-driven-dialog.

Three possibilities (Gerald Weinberg)? For everything? That smacks of BDUF FUD (fear, uncertainty and doubt)?? Can't we just YAGNI that? Well, remember, these are make or break decisions. Game shapers and game changers.

"Thought happens not only inside the skull but out in the world, too; it's an act of continuous assembly and reassembly that draws on resources external to the brain. For another: the kinds of materials available to "think with" affect the nature and quality of the thought that can be produced."

— Annie Murphy Paul

Circumstances matter — is this ordered and routine, or novel and unpredictable?

Again: Cynefin and Context

"All too often, managers rely on common leadership approaches that work well in one set of circumstances but fall short in others. Why [..]? The answer lies in a fundamental assumption of organizational theory and practice: that a certain level of predictability and order exists in the world. This assumption, grounded in the Newtonian science that underlies scientific management, encourages simplifications that are useful in ordered circumstances. Circumstances change, however, and as they become more complex, the simplifications can fail."

– David Snowden and Mary Boone

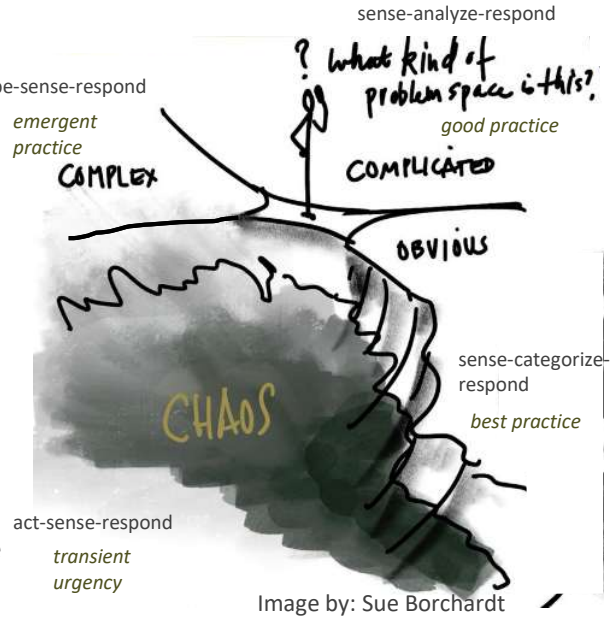


Image by: Sue Borchardt

Liz Keogh's Introduction to Cynefin

Liz Keogh has a useful introduction to Dave Snowden's Cynefin framework (and the extracts below are from Liz Keogh's post). Cynefin introduces four domains – obvious, complicated, complex and chaotic:

Obvious

Obvious problems are ones that either children can solve, or, if they do require expertise, the solution is obvious. In the obvious domain, there's normally one good way to solve the problem – a "best practice".

Complicated

As things become more and more complicated, the solution requires more and more expertise. A watchmaker knows how to fix your watch. The outcome is still predictable, but now it takes an expert to know how to get there. Both the Obvious and Complicated domains are called ordered. Ordered problems have repeatable solutions; the same process applied to the same problem will always work.

Complex

Complex problems are ones in which the solution, and the practices which lead to it, emerge. While it's possible to think of examples of what a solution might look like, attempting to create that solution usually creates unexpected side-effects; other problems or unintended consequences that might need to be solved. Cause and effect are only correlated in retrospect; you can see how you got there, but you couldn't possibly have predicted it. This is the domain of "wicked" problems that tend to resist being easily solved with expertise. In the complex domain, we have to probe the problem.

"A Quick Introduction to Cynefin ," by Liz Keogh

Chaos

Chaos is a transient domain; it resolves itself quickly, and not necessarily in your favor. It's dominated by urgency and the need to act, and act fast.

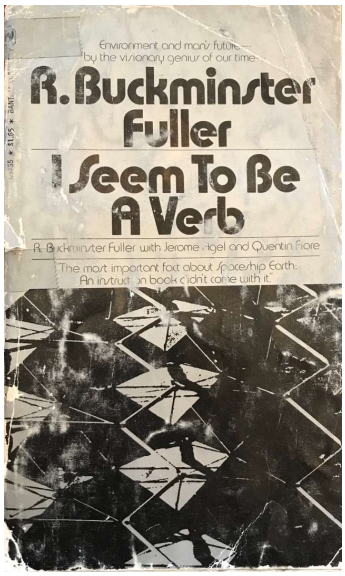
"Cynefin, pronounced ku-nev-in, is a Welsh word that signifies the multiple factors in our environment and our experience that influence us in ways we can never understand"

— David Snowden and Mary Boone

The Leadership Moment

"I always say to myself, what is the most important thing we can think about at this extraordinary moment."

– Buckminster Fuller



TECHNICAL DECISIONS

What at this Moment?

It's a lot? Sure. We need to continually be asking ourselves the orienting question: "what at this extraordinary moment, is the most important thing for me, and for us, to be attending to?" The discipline is in the question. We still need to do mundane things, supporting things, etc. However, we're guiding our own attention, and, as a leader, potentially others'.

Leadership is associated with vision, or at least, we tend to attribute absence of leadership when clear, shared vision is lacking. Is vision what we need to be attending to? Is it threat, and inhibitors to success? Is it decisions and shared understanding of the outcomes and our chosen approach to reaching them?

Who *at this Moment?*

We need to be aware of, think consciously about, who is involved. Diverse perspectives, born of different backgrounds and experience sets, are important to understanding the (various) contexts of use, development, and operations, surface ethical considerations, and understanding alternatives and impacts.

Asking who, is also about understanding that with too many involved, we can slow the process down (reaching agreement, decisions by committee, etc.). And we need to balance this with creating shared understanding and insight into the constraints and forces taken into account. We can get some of these benefits, involving others in reviews, etc.

Minimalist Discipline

Adopt a minimalist orientation:

Leaders work across; any decisions we make (or decisions we guide in the making), ought to be those that have substantive consequence and impact on system outcomes, and implications in different contexts.

Minimalist: does the decision need to be made by me? Scope? Timing? Impact? No? Then don't make it!

Strengthening the Decision

Recap: Characterize the decision

- Clarify outcome sought
- Understand the context(s), shaping forces, constraints and trade-offs

Recap: Determine the response

- Next: *Strengthen the proposed approach*

> *"Doing the right thing is a matter of wisdom, doing the thing right is a matter of knowledge and understanding."*
>
> — *Russ Ackoff**

TECHNICAL DECISIONS

* Via Trond Hjorteland

Strengthening Our Decisions

So far, we've been considering why we make decisions and what kinds of decisions we make more intentionally and when. And what goes into making technical decisions. Not prescriptively, for it's a judgement call, or rather, sets of judgment calls.

To inform the decision, we seek to understand the context (or circumstances (Diana Montalion)), and seek to identify the best possible approach, or solution, or resolution.

If the decisions we're making are make or break, it's worth considering how might we strengthen our reasoning underlying the decision. What do we need to learn, to increase confidence (our own, and stakeholders') in our approach? What constrains us? What are we not willing to compromise on? And yes, this can take time. We acknowledge uncertainty and incompleteness and seek to improve our understanding of impact and fit and options and opportunities.

Not that we want bullets, but to play with the "no silver bullets" (Brooks in MMM) expression: we know that objectivity isn't a thing we reach, but in attempting it, we employ the practices of science and experiment, to try to get to sounder decisions. To discover. To adapt. Where it matters.

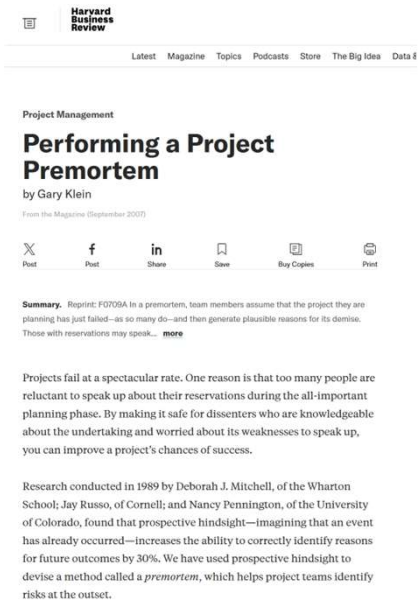
> *"My first round I tried appealing to rationality. Then I ran smack bang into bounded rationality."*
>
> — *Abdul Gani*

> *"look at reasons in terms of:*
> - *fact: is it so?*
> - *inference: does it follow?"*
> - *weight: does it matter?"*
>
> — *Diana Montalion*

Premortem

"A typical premortem begins after the team has been briefed on the plan. The leader starts the exercise by informing everyone that the project has failed spectacularly. Over the next few minutes those in the room independently write down every reason they can think of for the failure—especially the kinds of things they ordinarily wouldn't mention as potential problems, for fear of being impolitic."

— Gary Klein, Performing a Project Premortem



Harvard Business Review

Latest Magazine Topics Podcasts Store The Big Idea Data

Project Management

Performing a Project Premortem

by Gary Klein

From the Magazine (September 2007)

Post Post Share Save Buy Copies Print

Summary. Reprint: F0709A In a premortem, team members assume that the project they are planning has just failed—as so many do—and then generate plausible reasons for its demise. Those with reservations may speak... more

Projects fail at a spectacular rate. One reason is that too many people are reluctant to speak up about their reservations during the all-important planning phase. By making it safe for dissenters who are knowledgeable about the undertaking and worried about its weaknesses to speak up, you can improve a project's chances of success.

Research conducted in 1989 by Deborah J. Mitchell, of the Wharton School; Jay Russo, of Cornell; and Nancy Pennington, of the University of Colorado, found that prospective hindsight—imagining that an event has already occurred—increases the ability to correctly identify reasons for future outcomes by 30%. We have used prospective hindsight to devise a method called a *premortem*, which helps project teams identify risks at the outset.

"Next the leader asks each team member, starting with the project manager, to read one reason from his or her list; everyone states a different reason until all have been recorded. After the session is over, the project manager [or leader] reviews the list, looking for ways to strengthen the plan."

— Gary Klein

Exercising Anticipation, to Strengthen

A powerful way to bring weakness in the decision or its rollout into view, is to use our imagination, past experience, and awareness of others experience in the industry, etc., to imagine that it failed, and do what is very similar to an incident review, tracing the network of causal effects to identify how we can strengthen our approach and avoid or prepare for hurdles ahead. We can do this kind of "project ourselves into the future" at different scales – for a low impact decision, a personal thought experiment might suffice.

Source:_https://hbr.org/2007/09/performing-a-project-premortem

Writing to Think (Better)

Title: short noun phrase
Context: desired outcomes and the forces at play (probably in tension)
Decision: describes our response to these forces
Status: proposed, accepted, deprecated or superseded
Consequences: describes the resulting context, after applying the decision



— Michael Nygard, Documenting Architecture Decisions, Nov 2011

"Here are [..] reasons that radiating intent is better than begging forgiveness:
Radiating intent gives a chance for someone to stop you before you do a thing, in case it's truly harmful
Radiating intent gives people who have information, or want to help, an opening to participate
Radiating intent leaves better evidence of your good will"
— Elizabeth Ayer

Source:_https://medium.com/@ElizAyer/dont-ask-forgiveness-radiate-intent-d36fd22393a3n

Write It Down — No Really, Do It!

Writing our thinking down helps us to see what we're thinking, so we can improve it – the thinking, or the way we're communicating it. It creates externalized reasoning and memory that others can access, understand, and help improve.

We've mentioned keeping a systems and decision journal as a way to be more reflective and intentional about learning from our system(s) and decisions, and what is shaping up in the context. Design pattern templates and ADRs (covered earlier) help direct our attention, so that in our conversations and in formulating and writing our reasoning down, we have a well-trod path to follow in the form of template structure.

Various architecture decision templates have been published, including by Jeff Tyree and Art Akerman then at Capital One (in IEEE Software, so this template and discussion gained exposure and influence), and Olaf Zimmerman at IBM. But Michael Nygard's simplified (and well-described) Architecture Decision Record template caught on as a just enough version for documenting architecture decisions in an Agile context.

See Nat Pryce's ADR tools on Github:
https://github.com/npryce/adr-tools
See also Upmo and Wisen Tanasa's examples:
https://opensoftwaredesign.com/upmo/decisions/

There are various templates for Architecture Decision Records; of note: Indu Alagarsamy's brevity-oriented template

Variations

Highlighting Impacted Stakeholders: stakeholders (those impacted by the decision) have different perspectives and orientations and concerns… and agendas… (and these unfold and evolve)… Noting who is impacted, reminds us who to involve.

[Product / Architecture] Decision Record

Status : [Draft / Proposed / Adopted / Retired]

☐ Any action items

1.	**Question to Decide On**	[What is the question you are trying to answer?]
	Context	[Brief description of the context]
2.	**Recommended Decision**	[Brief description of the decision]
	Supporting Arguments	[What are the reasons which led to this decision?]
	Consequences / Constraints	[What are the ramifications of this decision, both positive and negative]
3.	**Other Options Considered**	[Were there other options? If so, list each option and the pros and cons of each option. [Option 1: option title] [Pros:] - Tk tk [Cons] - Tk tk *Duplicate this for each option considered*
	Impacted Stakeholders	Stakeholders impacted by this decision: tk tk Stakeholders (names and roles) of who participated/signed off in this decision making process: tk tk
	Related References	[Link to any meeting notes, Slack threads, Figjam / Miro board links, etc]

From: Indu Alagarsamy, *Document your product and software architecture decisions,* https://domainanalysis.io/p/document-your-product-and-software

Architecture Decision Records

Joel Parker Henderson has collected various resources around Architecture Decision Records, from templates and guidance, to links to the ADRs of various organizations, on github.

https://github.com/joelparkerhenderson/architecture-decision-record

Examples: https://github.com/joelparkerhenderson/architecture-decision-record/tree/main/examples

The Application Logging ADR of the HHS/Head-Start-TTADP project might be contrasted with the Metrics, Monitors and Alerts ADR (as an exercise, noting that the team's context and judgment factors):

- Application Logging https://github.com/HHS/Head-Start-TTADP/blob/main/docs/adr/0004-application-logging.md
- Metrics, Monitors and Alerts https://github.com/joelparkerhenderson/architecture-decision-record/tree/main/examples/metrics-monitors-alerts#metrics-monitors-alerts

The ADR template is vital. Still, too much of a good thing sinks itself under its own weight, so there's the matter of which decisions to record and how much of the reasoning to persist.

Many, but not all, ADRs focus on technology choices.
Example of a team oriented decision: https://github.com/joelparkerhenderson/architecture-decision-record/tree/main/examples/high-trust-teamwork

README.md

Decision Record for High Trust Teamwork

Date: [Insert Date]

Participants: [Insert Names of Participants]

Decision Reach: [Unanimous/ Majority decision/ Individual Decision]

Decision Description

The decision to adopt High Trust Teamwork as a core value and belief in our team and organizational culture.

Alternatives Considered

We considered alternative approaches to teamwork, including low trust or no trust teamwork, as well as other frameworks such as cognitive diversity teams, centralized decision-making teams, and agile teams.

Benefits and Risks

The benefits of high trust teamwork include improved collaboration, better decision-making, increased motivation and engagement, higher job satisfaction and retention rates, and enhanced innovation and creativity. The risks associated with high trust teamwork include the potential for trust breaches, conflicts, and misunderstandings due to miscommunications or not adhering to the high trust principles.

Decision Outcome

It was unanimously agreed upon to adopt high trust teamwork as the core value and belief in our team and organizational culture. We acknowledged the importance of building and maintaining trust through transparency, honesty, integrity, and respect, and recognized that this approach would help us to establish a positive and rewarding work environment.

Action Items

To implement high trust teamwork, we will take the following steps:

1. Define high trust principles and practices and communicate these to all team members
2. Establish regular feedback mechanisms and communication channels to keep the team updated on progress, challenges, and opportunities
3. Attend training on high trust teamwork and actively participate in team-building activities
4. Encourage and recognize behaviors that demonstrate high trust principles in action
5. Develop a plan to address potential trust breaches, conflicts, or misunderstandings and have a process to handle such situations effectively

Power of Writing things Down

"Writing and sharing that writing with others creates accountability. It also almost always leads to more thorough decisions."

— Gergely Orosz



Scaling Engineering Teams via RFCs: Writing Things Down

I have recently been talking at small and mid-size companies, sharing engineering best practices I see us use at Uber, which I would recommend any tech company adopt as they are growing. The one topic that gets both the most raised eyebrows, as well the most "aha!" moments is the one on how the planning process for engineering has worked since the early years of Uber.

When working at large companies like Microsoft or smaller ones like Skyscanner, there have been two things related to planning that have always bugged me. First, the lack of visibility on others building or having built the same thing as my team. Second, the tech and architecture debt accumulated due to different teams building things very differently, both approach-wise and quality-wise.

https://blog.pragmaticengineer.com/scaling-engineering-teams-via-writing-things-down-rfcs/

Writing Things Down

"Usually, I see this being less straightforward. "This is not what I meant when we talked." or "What about this important edge case we forgot?" and "If we change this here, it could break this other part of the system" are things that often come up when writing the plan down. It's great to have these discussions before having the same realizations when the project is halfway done."

— Gergely Orosz, Scaling Engineering Teams via RFCs

"Understanding requires elaboration, so it's important for us to understand how we can elaborate better. Naturally, we could elaborate our findings by talking about them. Unfortunately in an oral presentation, we could get away with unfounded claims. A "you know what I mean", or a confident gestures may stop your peer from assessing what you have elaborated.

Instead of just using an oral communication, we should elaborate by writing too. There are many points in the process of writing where we would question and challenge the arguments that we have written when they don't work. If there are contradictions or gaps, our writing show that to us. Elaboration in writing also mean that we don't have to wait for anyone to be available to listen to us, we can do it at any time."

— Wisen Tanasa, "Elaborate in writing to test your understanding"

One thing I like to add, is that in addition to getting our own thinking clearer, a good discussion (in writing, illustrated well, as relevant, too) of strategically and architecturally significant decisions, is an important venue for sharing design insight and engineering experience!

"Socialize. Pairing. Whiteboarding. Story-telling over lunch. We humans have used social methods for millennia to communicate our most sacred concepts."

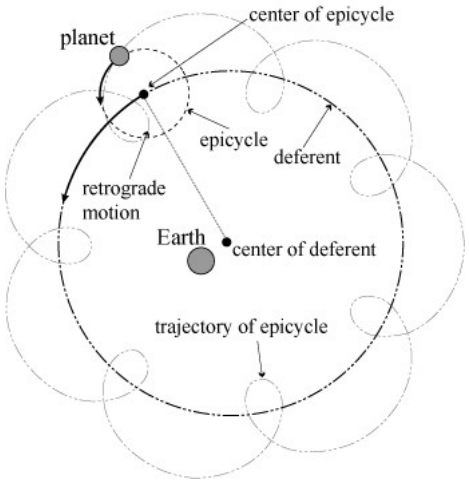
— Kent Beck, "The Documentation Tradeoff"

This is so important! *And* we also use writing. To get clearer. To explore. To convey (to others, and across time). To convene conversation around.

And the conversations with our externalized thinking, *and with others*, is important. It creates further avenues to gaining access to different perspectives, to surface and probe our assumptions.

Perspective, Illustrated

"what does it feel like, emotionally, to be wrong?"
Kathryn Schulz



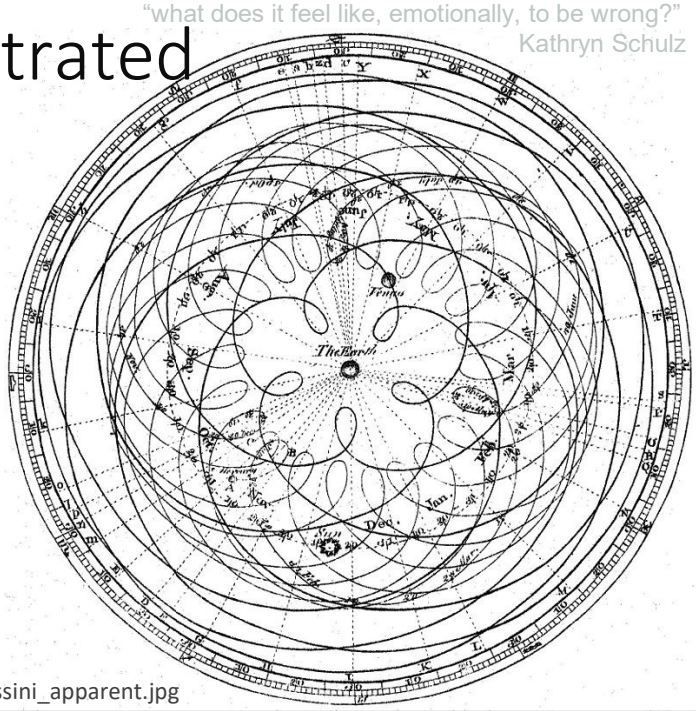


Image: wikimedia.org/wikipedia/commons/0/0e/Cassini_apparent.jpg

"Such assumptions appear so obvious that people do not know what they are assuming because no other way of putting things has ever occurred to them. With these assumptions a certain limited number of types of [..] systems are possible"

—Susanne Langer, Philosophy in a New Key

Perspective (Shifts)

Christin Gorman used this example in a wonderful talk recently: Before Copernicus, the earth was generally assumed to be at the center of the universe (geocentricism), and observations about the planets' movements were explained by a model that worked – its predictions were consistent with observations. Copernicus argued that the sun was at the center of the solar system, which resulted in a much simpler model, that explained and predicted the planets paths.

Christin uses this as an example of how we can be wrong. (And not know it. Until we know it.) Which brings in Kathryn Schulz's TED Talk: *On being wrong*. It is wonderful, and highly recommended, but here is an important (and funny, when she does it) take-away: When asked "what does it feel like, emotionally, to be wrong?" we answer things like "embarrassed," "awkward," (and if we're self-aware?) "defensive," … Kathryn points out: "That is answering a different question – namely, "what does it feel like to find out we're wrong?" She recalls how, when running off a cliff, Wile Coyote continues running -- on thin air, and only falls when he realizes there's only air beneath him. And she points out that *being wrong feels exactly the same as being right*. Sure, there's the old quip: "The first principle is that you must not fool yourself – and you are the easiest person to fool." But notice, that was talking about *you*. I'm kidding, but also not. It's key to Kathryn's talk.

Christin Gorman: Our architecture is a mess! Are you sure?, DevCon 2019

Kathryn Schulz: https://www.ted.com/talks/kathryn_schulz_on_being_wrong

Assumptions, and Perspective



A change in perspective helps make unstated assumptions (and other options) visible.

From Dawn Ahukanna: we need to actively surface not just assumptions, but our degree of confidence in them, and continually probe and update our understanding of the probability of the occurrence of assumptions that shape decisions, especially critical ones

TECHNICAL DECISIONS

'But the critical ones are what Jeff Bezos calls "irreversible" and therefore load bearing. If they are built on assumptions (with probability o) "Things fall apart; the center does not hold" (Yeats, quoted by Chinua Achebe in Things Fall Apart.)'

— Dawn Ahukanna

** referencing the Simons and Chabris Selective Attention Experiment.*

Repairing Blindspots

"A change of perspective " ⋯ Alan Kay reminds us to take a different vantage point, to see from a different perspective, use the lens of various views. We need to notice what is hard to notice from inside the tunnel of our own vision — where what we're paying attention to, shapes what we perceive and pay attention to. Another way to get a change of perspective, is to get another person's perspective. Our team can miss the gorilla*, so to speak, when our attention is focused on the design issues of the moment. Fresh perspective, and even just naive questions about what the design means, can nudge an assumption or weakness into view. And merely telling the story, unfolding the narrative arc of the architecture to fit this person or audience, then that, gets us to adopt more their point of reference, across more perspectives — in anticipation, and when we listen, really listen, to their response and questions.

We need to adopt the discipline of not just accepting our initial understanding, but rather seeking different understandings. This illuminates options, and gives us other things to try. These are the significant decisions, decisions about the important stuff, after all.

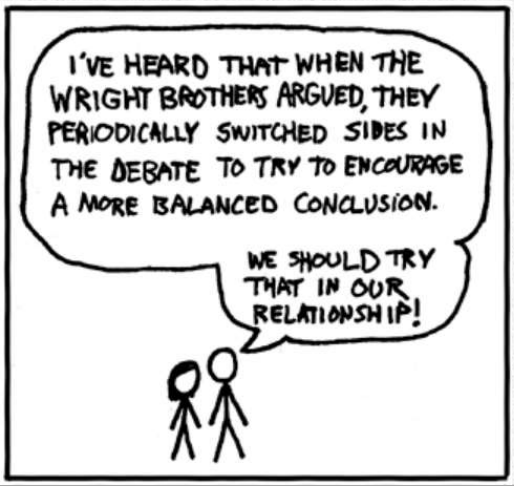


Image xkcd.com/106/

Don't ignore the social influences and the social forces around the decision

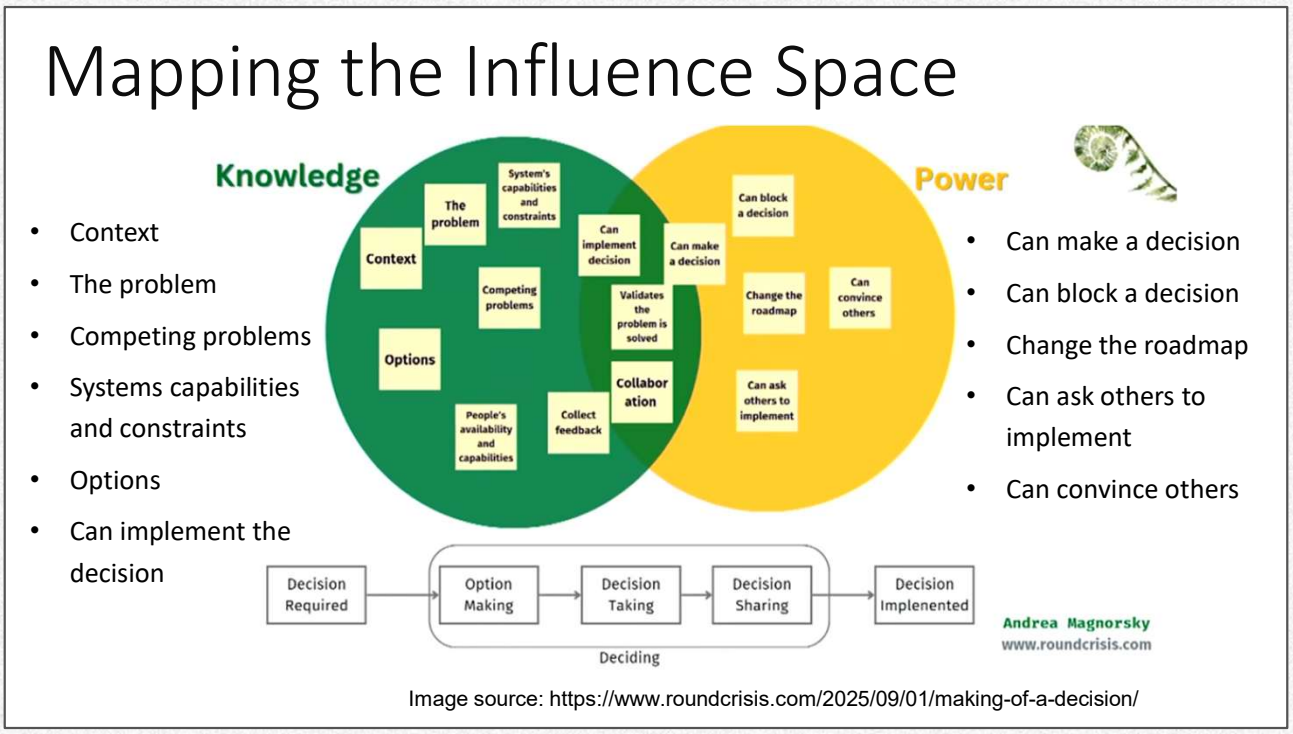


Image source: https://www.roundcrisis.com/2025/09/01/making-of-a-decision/

Mapping the Space Around the Decision

During the Advent of System Seeing in 2024, Michael (@mmby on mastodon) shared these various views of the org:

Maps like this are so useful for thinking about relationships and information flows, and influence spaces that result.

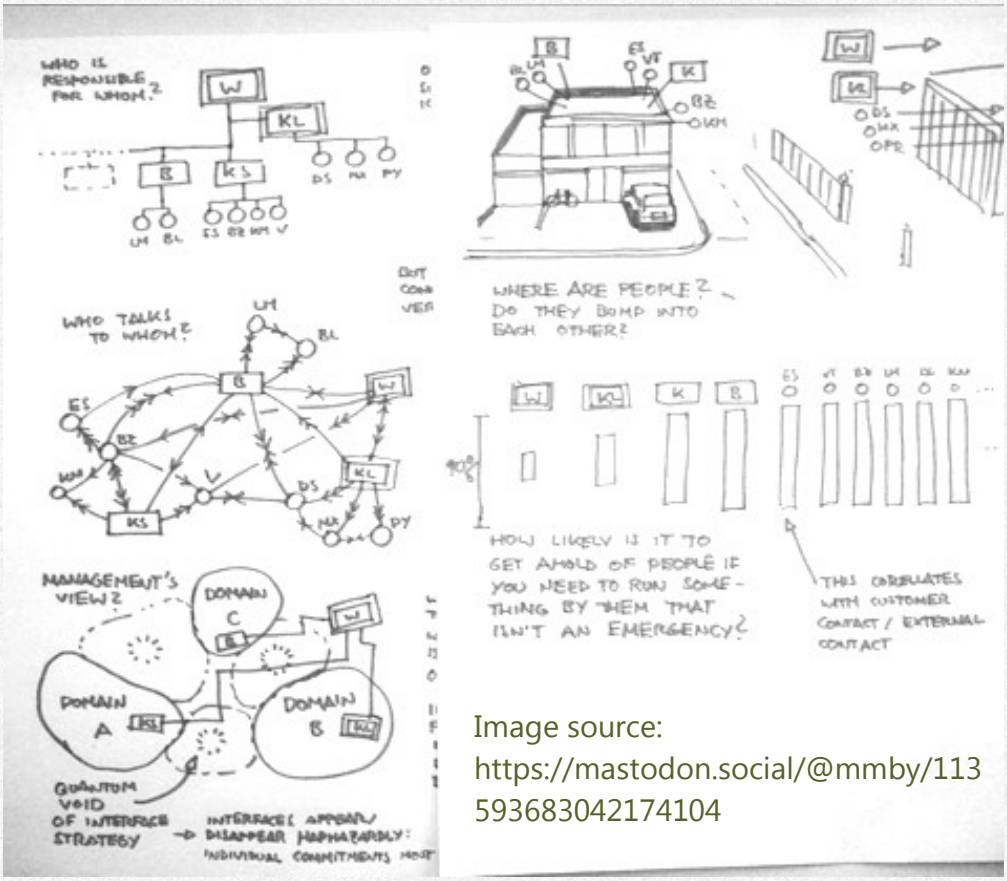


Image source: https://mastodon.social/@mmby/113593683042174104

We can take this in various directions. If different parts of the organization are exhibiting stances and orientations to a decision, we can use a Rich Picture to explore the differences in concerns across the space, or explore a set of causal influences/force field diagrams, to gain insight into the different experiences of the force fields and impacts across different groups. And so on.

The "big idea" here, is to use or *create* a diagram that helps foster and focus the conversations and insights that need to be explored, to understand the situation.

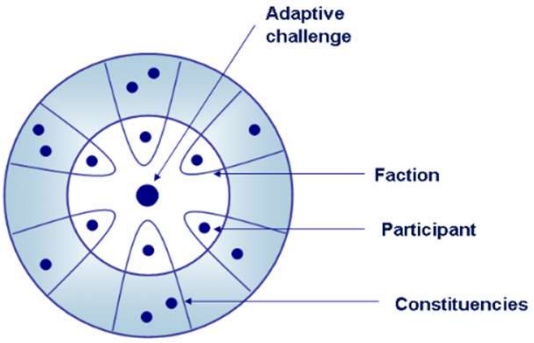
Stakeholder Cares and Concerns

i. put the challenge (that needs a decision) in the center of a circle

ii. add the stakeholders (who care about the outcome and consequences) around the challenge, within the circle

iii. add the constituencies those stakeholders represent outside the circle

Now, you can start to analyze what these various groups care about, and how they align, or don't (where there is conflict or tension)



stakeholder map from Ronald Heifetz and collaborators

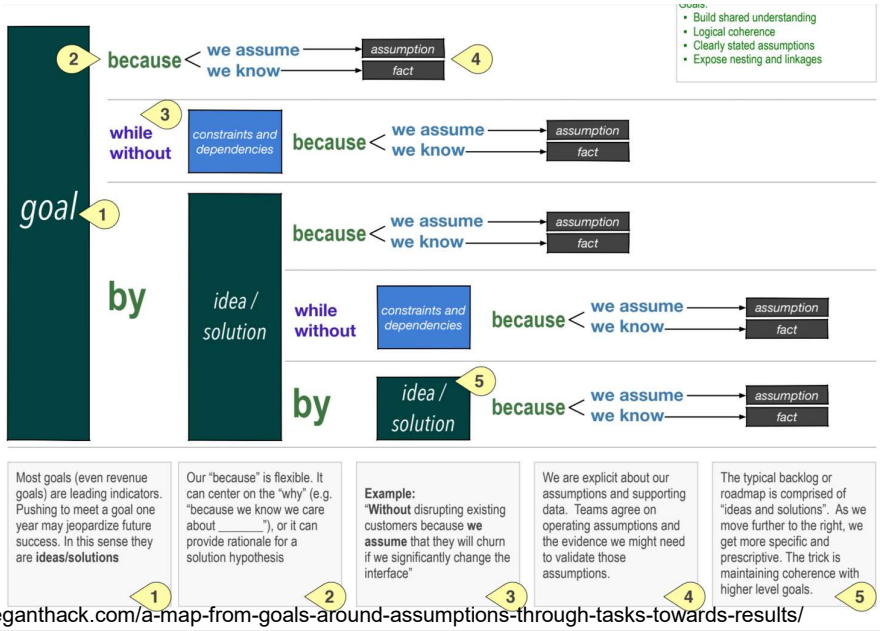
Stakeholders, well, Have a Stake!

Decisions with significant consequences have different impacts for different groups and individuals within the organization, and they may resist the decision and its changes for reasons that make a lot of sense to them, in their situation. Understanding their cares and concerns (for example, loyalties and ties to a particular segment of the customer base, or managing teams with different needs), helps to clarify the decision space and options (identify the forces and trade-offs), but also helps identify resistance and where that is coming from (what the concerns and orientations are). This is input to the decision (problem understanding and viability of options), and also impacts how we help the organization enact the changes required by the decision. For example, we might think about investments in infrastructure (building capabilities and tools and otherwise "paving the way" to make the change), to help ease transitions the decision calls for.

"Adaptive challenges are difficult because their solutions require people to change their ways. Unlike known or routine problem solving for which past ways of thinking, relating, and operating are sufficient for achieving good outcomes, adaptive work demands three very tough, human tasks: figuring out what to conserve from past practices, figuring out what to discard from past practices, and inventing new ways that build from the best of the past"

— Laurent Ledoux

Assumption and Constraint Mapping



Source: John Cutler, https://eleganthack.com/a-map-from-goals-around-assumptions-through-tasks-towards-results/

Goals as Fields and Enabling Constraints

So we have explored dynamics using influence maps and causal loop diagrams. Now, with the Technical Debt Exercise in mind, identify challenges we need to address. Pick a challenge. What is the goal that makes this challenge one we need to address? Explore how we might achieve the goal, using the template – see template on the slide (from John Cutler), and description on John Cutler's site at:

https://eleganthack.com/a-map-from-goals-around-assumptions-through-tasks-towards-results/

"I try actively to question myself and my certainties"

— Jérémie Zimmerman

"A lot of work is the work of noticing what work needs doing."

— Elizabeth Zagroba

"Design involves assumptions about the future of the object designed, and the more that future resembles the past the more accurate the assumptions are likely to be. But designed objects themselves change the future into which they will age."

— Petroski, To Engineer is Human

Argumentation Model

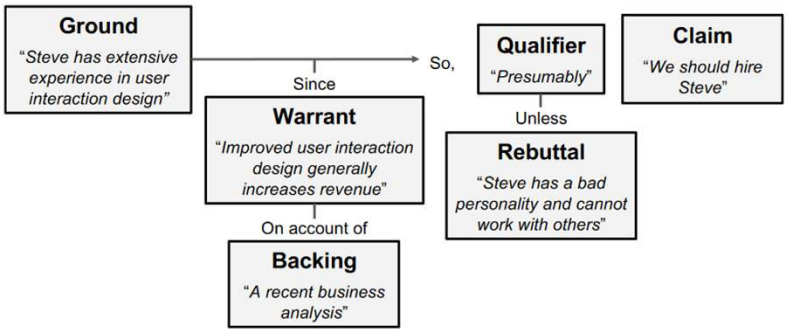


Figure 2. Relationship among the six components of the Toulmin's model of argumentation [59] with examples.

Image source:
https://www.researchgate.net/publication/341694611_ArguLens_Anatomy_of_Community_Opinions_On_Usability_Issues_Using_Argumentation_Models

Decision-based Evidence Making (Finding)

From Cameron Tonkinwise (on LinkedIn):

"The lead example in the article is [..] the Aeron Chair for which initial market research evidenced distaste. With this example, the article is defending how things other than positivist evidence often inform a decision, things like expert pattern detection. If an external context demands 'evidence' to backup those 'intuitions,' then it is justifiable, the article argues, to do 'evidence making' - though a better phrase is 'evidence finding' to get some distance from 'evidence fabricating.'

Even 'evidence finding' risks confirmation bias. Then again, any well-designed scientific experiment is structured around a lamp-post-lost-key hypothesis that it seeks, hopefully, to disconfirm.

This article had me thinking about Stephen Toulmin's approach to argument analysis. This points to the importance of being able to identify not just the claim and the evidence for the claim, but the warrant. This nice old word means something like 'the philosophical assumption behind why that kind of evidence is evidence of the claim' (hence police need a basis on which to search for evidence) [..]

All of this is a core problem in design: you can do research into a context and perhaps discover what problems people have, but that won't tell you how to best solve that problem, something that requires an abductive leap.

The thing to do is focus on the 'decision' side not the 'evidence' side. Make error-friendly, reversible decisions. 'Safe-to-learn' is a social context that privileging evidence avoids taking responsibility for."



"CONVERSATION, n. [..] each exhibitor being too intent upon the arrangement of his own wares to observe those of his neighbor.

DECIDE, v.i. To succumb to the preponderance of one set of influences over another set.

DELIBERATION, n. The act of examining one's bread to determine which side it is buttered on.

RATIONAL, adj. Devoid of all delusions save those of observation, experience and reflection.

REASON, v.i. To weigh probabilities in the scales of desire."

—Ambrose Bierce, The Devil's Dictionary

That's… a Lot…Which Decisions?

"The value of every decision we make depends on the context in which we make it. In *The Lord of the Rings*, Frodo's journey to destroy the ring is meaningful inside the context of Middle Earth. Otherwise, he's a short, hairy guy with apocalyptic hallucinations."

— Diana Montalion



Decisions are central, and it is a great template, but you can just hear the captain in the cockpit yelling "pull up, pull up" — we'll run into a veritable forest of decision trees if we speed too far too fast down that runway just now. Which decisions?

*'wisdom is the ability to know what "it depends" on' **

Which Decisions?

When I first read Martin Fowler's "Who needs an Architect?" column, I playfully summarized it as:

> Which decisions does the architect make?
>
> Architecturally significant decisions!
>
> What is architecturally significant?
>
> The architect decides!

Yes, it's a tautology. Yes, I repeat myself. To playfully indicate that architects discern what is architecturally significant, to determine where to focus architectural (*system* design) attention.

As systems become more complex, we mean "the architect*s* [plural!] decide", and note that decisions that prove to be architecturally significant aren't all known or knowable upfront. The system evolves. Many people are involved, making decisions that shape the system. Still, architects bring experience, context awareness and system understanding — and a *system* design purview — to judgments about what is architecturally significant.

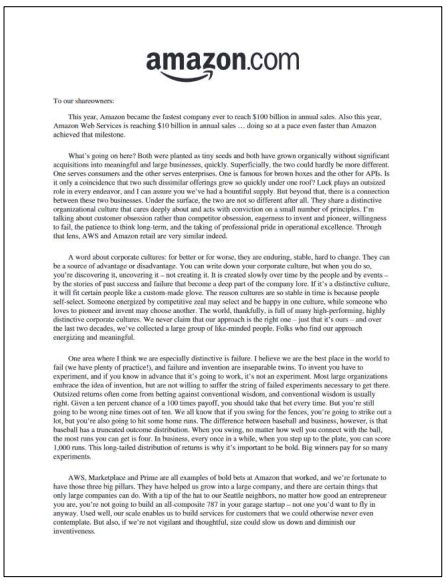
Judgment. Can we say more?

* https://twitter.com/vladikk/status/1134124637925892096

Irreversible Decisions

"Some decisions are consequential and irreversible or nearly irreversible [..] and these decisions must be made methodically, carefully, slowly, with great deliberation and consultation."

— Jeff Bezos



amazon.com

To our shareowners:

This year, Amazon became the fastest company ever to reach $100 billion in annual sales. Also this year, Amazon Web Services is reaching $10 billion in annual sales ... doing so at a pace even faster than Amazon achieved that milestone.

What's going on here? Both were planted as tiny seeds and both have grown organically without significant acquisitions into meaningful and large businesses, quickly. Superficially, the two could hardly be more different. One serves consumers and the other serves enterprises. One is famous for brown boxes and the other for APIs. Is it only a coincidence that two such dissimilar offerings grew so quickly under one roof? Luck plays an outsized role in every endeavor, and I can assure you we've had a bountiful supply. But beyond that, there is a connection between these two businesses. Under the surface, the two are not so different after all. They share a distinctive organizational culture that cares deeply about and acts with conviction on a small number of principles. I'm talking about customer obsession rather than competitor obsession, eagerness to invent and pioneer, willingness to fail, the patience to think long-term, and the taking of professional pride in operational excellence. Through that lens, AWS and Amazon retail are very similar indeed.

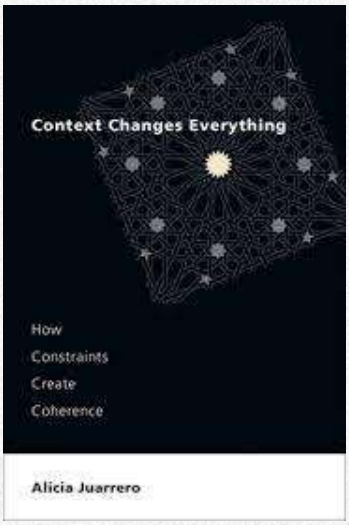
A word about corporate cultures: for better or for worse, they are enduring, stable, hard to change. They can be a source of advantage or disadvantage. You can write down your corporate culture, but when you do so, you're discovering it, uncovering it – not creating it. It is created slowly over time by the people and by events – by the stories of past success and failure that become a deep part of the company lore. If it's a distinctive culture, it will fit certain people like a custom-made glove. The reason cultures are so stable in time is because people self-select. Someone energized by competitive zeal may select and be happy in one culture, while someone who loves to pioneer and invent may choose another. The world, thankfully, is full of many high-performing, highly distinctive corporate cultures. We never claim that our approach is the right one – just that it's ours – and over the last two decades, we've collected a large group of like-minded people. Folks who find our approach energizing and meaningful.

One area where I think we are especially distinctive is failure. I believe we are the best place in the world to fail (we have plenty of practice!), and failure and invention are inseparable twins. To invent you have to experiment, and if you know in advance that it's going to work, it's not an experiment. Most large organizations embrace the idea of invention, but are not willing to suffer the string of failed experiments necessary to get there. Outsized returns often come from betting against conventional wisdom, and conventional wisdom is usually right. Given a ten percent chance of a 100 times payoff, you should take that bet every time. But you're still going to be wrong nine times out of ten. We all know that if you swing for the fences, you're going to strike out a lot, but you're also going to hit some home runs. The difference between baseball and business, however, is that baseball has a truncated outcome distribution. When you swing, no matter how well you connect with the ball, the most runs you can get is four. In business, every once in a while, when you step up to the plate, you can score 1,000 runs. This long-tailed distribution of returns is why it's important to be bold. Big winners pay for so many experiments.

AWS, Marketplace and Prime are all examples of bold bets at Amazon that worked, and we're fortunate to have those three big pillars. They have helped us grow into a large company, and there are certain things that only large companies can do. With a tip of the hat to our Seattle neighbors, no matter how good an entrepreneur you are, you're not going to build an all-composite 787 in your garage startup – not one you'd want to fly in anyway. Used well, our scale enables us to build services for customers that we could otherwise never even contemplate. But also, if we're not vigilant and thoughtful, size could slow us down and diminish our inventiveness.

TECHNICAL DECISIONS

"One common pitfall for large organizations – one that hurts speed and inventiveness – is "one-size-fits-all" decision making."
– Jeff Bezos



No "One Time to Rule Them All" Decision Making

So, strategy and architecture are about scope and impact, and not something that is simply determined by being done upfront — that is, by timing. Rather, the other way round. If it's strategically or structurally significant, we want timing to factor in decision making. Is this something we need to pay attention to now? Why?

We're using judgment to decide on the timing of decisions. And one way to inform this judgment, as pointed out by Sidharth Masaldaan, is to consider risk. What is highest risk and needs our (scarce!) expertise, perspective, attention and time now? And what do we need to enable (by deciding and building)? Yes, in the sense of enabling constraints.

No "One-Size Fits All" Decision Making Either

In his 2015 letter to Amazon shareholders, Jeff Bezos made this important distinction between irreversible and reversible decisions, emphasizing that consequential irreversible decisions need to be made with great deliberation and consultation.

Source: https://www.sec.gov/Archives

Not all decisions are equal. What differences make a difference?

Irreversible Decisions

"If you walk through and don't like what you see on the other side, you can't get back to where you were before."

— Jeff Bezos



TECHNICAL DECISIONS

Attending to Irreversible, Consequential Decisions

We need to make those decisions deliberately, attentively

Shane Parrish collected together a useful series of decision making heuristics in a twitter thread. Here are several (the numbers are Parrish's) that we've selected for their bearing in the case of more consequential decisions [and we've added a few notes]:

17. Put things on a reversibility/consequence grid — irreversible and high consequence decisions likely require more time. The rest of the time you can usually go fast.

Source: https://twitter.com/farnamstreet/status/1026105498372845571

10. The rule of 5. Think about what the decision looks like 5 days, 5 weeks, 5 months, 5 years, 5 decades.

11. Let other people's hindsight become your foresight. [Do the research; draw on expertise.]

13. Ask what information would cause you to change your mind. If you don't have that information, find it. If you do, track [it] religiously.

22. Walk around the decision from the perspective of everyone implicated (shareholders, employees, regulators, customers, partners, etc.)

26. Ask yourself "and then what?" [and "what if?" and "what else?"]

Source: Shane Parrish (@farnamstreet), on twitter, 5 Aug, 2018

'Legacy code is often defined as "code that makes more design decisions than the team working on it".'

— Ángel Siendones Sillero

Reversible Decisions

"But most decisions aren't like that – they are changeable, reversible – they're two-way doors."

— Jeff Bezos



TECHNICAL DECISIONS

Reversible Decisions

It's worth highlighting two takeaways from Bezos's insights here:

- where we can, make decisions reversible — reduce the cost of change.
- pay particular attention to consequential irreversible decisions — attend to those that have high cost of change

"If you're good at course correcting, being wrong may be less costly than you think" — Jeff Bezos

Reversibility Approaches

In *Taming Complexity with Reversibility*, Kent Beck outlines several approaches used at Facebook for making changes smaller, and getting feedback more rapidly, so decisions can be tried out and assessed, and reversed if they don't pan out well (enough), before they become entangled in other decisions, expectations and habits. These have include:

- *Development servers*. Each engineer has their own copy of the entire site. Engineers can make a change, see the consequences, and reverse the change in seconds without affecting anyone else.
- *Code review*. Engineers can propose a change, get feedback, and improve or abandon it in minutes or hours, all before affecting any people using Facebook.
- *Internal usage*. Engineers can make a change, get feedback from thousands of employees using the change, and roll it back in an hour.

Source: Kent Beck, *Taming Complexity with Reversibility*

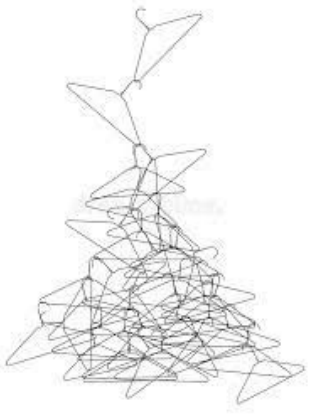
In part, these satisfy the second of Palchinsky's Principles:

"when trying something new, do it on a scale where failure is survivable" — Peter Palchinsky

(ir)Reversible Decisions

A significant decision that may be reversible to begin with, tends (over time) to become "entangled" with other decisions, and less readily backed out of.

The system isn't just coupled, but coupled to past assumptions.



TECHNICAL DECISIONS

(ir)Reversible Decisions

We have this interplay between decisions made early or next, to bring the benefit of those decisions forward, and decisions deferred to retain options. Even decisions about where we start, have consequences. We canalize pretty quickly. That is, we reduce the space of designs that are reachable. We gain an identity, internally and in the market. That shapes in ways we notice and don't — we make assumptions about value to customers and users, about system capabilities we're creating and so need to build in our teams, and so on. Sure, we (or the market) test(s) our theory of value — in so doing, shoring up the assumptions we proceed on. As users integrate our system into their workflows and systems, they build up expectations or assumptions about how things work, and ought to. As do we. There are a myriad ways our systems become coupled and resist change. Pretty soon, we call them "legacy systems" in that wry sense of a legacy we both value (or at least depend on) and regret. Change becomes costly (in various terms, including economic, but also factors such as capabilities and expertise we've built and not longer need which has technical, social and emotional impacts that require substantive organizational/political will) and hard. It rends and ripples through the systems of interconnected, inter-reliant systems.

Leading is about creating conditions for decision courage (we need to make some, now and soon) and decision scrutiny (we need to bring impacts into view and figure out what to do). Judgement and discernment factors. Including discernment about the scale and scope of the decision and its impacts. We're talking here about decisions of substantive consequence and non-local impacts.

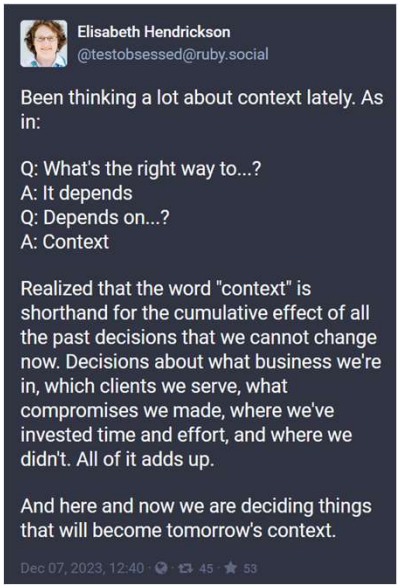
The challenges of "modernizing" aging systems go well beyond "the technology" — the entanglements that create inertia are not just in "the code" and "the stack"… They run deep — into assumptions about our very identity, even.

(Ir)reversible Decisions

"Realized that the word "context" is shorthand for the cumulative effect of all the past decisions that we cannot change now. Decisions about what business we're in, which clients we serve, what compromises we made, where we've invested time and effort, and where we didn't. All of it adds up.

And here and now we are deciding things that will become tomorrow's context."

— Elisabeth Hendrickson



Adaptive Capacity … and Entanglements

Software is highly mutable. Humans lend adaptive capacity to our sociotechnical systems, allowing us to evolve them into astonishingly complex, and useful systems. One characterization of legacy systems:

> "Legacy systems are valuable because they continue to exist; they wouldn't continue to exist if they weren't valuable." — Kevlin Henney

Nonetheless, our systems tend to canalize – internal structures are adapted to fit shifts in context, but that fit comes at a cost, including becoming embedded in other systems that rely on them, and resist change. The 737 MAX story is illustrative of forces in tension⋯

> "So when Boeing designed the 737 MAX, they were trying to balance two conflicting requirements. [Accounts differ: bigger engines for fuel efficiency or for range.] The other was to keep the design sufficiently similar to the existing 737 aircraft that pilots wouldn't need a new type rating [which aircraft pilots are allowed to fly]. But it turns out those new engines on the 737 MAX were actually so big they wouldn't quite fit under the wings. They couldn't redesign the airframe to make the wings higher, otherwise it wouldn't have been a 737 any more, so instead they mounted those new engines a little further forward and a little higher than the old ones. And this is where it gets complicated. That new engine placement introduced handling problems – it meant that when you open the throttle, the aircraft had a tendency to stick its nose up in the air. And that's bad, because if the nose goes up too high the plane is going to stall. And so the solution to this was software. Specifically, a software system called the MCAS – the Maneouvering Characteristics Augmentation System." — Dylan Beattie, The Cost of Code, 2019

Continually adapting, but the possibility envelope is shaped by prior decisions.

"The law of stretched systems:

every system is stretched to operate at its capacity; as soon as there is some improvement, for example in the form of new technology, it will be exploited to achieve a new intensity and tempo of activity.

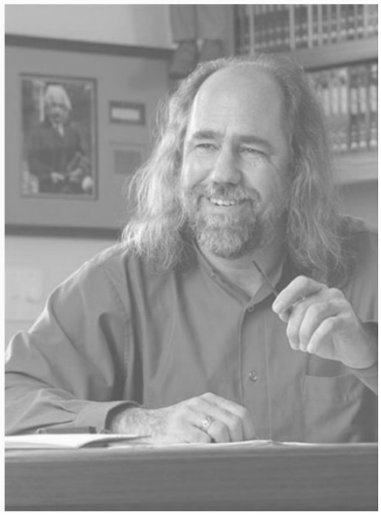
— David Woods

David Woods, Laws that Govern Cognitive Work, 2002

(Ir)reversible Decisions

"Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

— Grady Booch



(ir)reversibility of decisions

low(er(ed)) cost of change

high cost of change

SOFTWARE ARCHITECTURE

The Point?

When we talk about cost of change in the context of architecture, we're typically thinking about the cost to make changes to the system. But cost of change plays in, in different ways. As the system takes shape, other systems develop expectations of our system – some implicit and some explicit, some critical and constraining, others not so much. As our system becomes embedded in expectations and commitments and reliance on its role in the broader ecosystem, it becomes hard to "reverse" or back out of shaping decisions. We might want to relate this to "sheering layers," but its at least good to recognize that we seek stabilities, even as we seek to adapt and evolve.

We have this interplay between decisions made early or next, to bring the benefit of those decisions forward, and decisions deferred to retain options. Even decisions about where we start, have consequences. We canalize pretty quickly. That is, we reduce the space of designs that are reachable. We gain an identity, internally and in the market. That shapes in ways we notice and don't — we make assumptions about value to customers and users, about system capabilities we're creating and so need to build in our teams, and so on. Sure, we (or the market) test(s) our theory of value — in so doing, shoring up the assumptions we proceed on. As users integrate our system into their workflows and systems, they build up expectations or assumptions about how things work, and ought to. As do we. There are a myriad ways our systems become coupled and resist change. Pretty soon, we call them "legacy systems" in that wry sense of a legacy we both value (or at least depend on) and regret.

"A good architecture reduces disruptive change. For example, if a on-the-wire protocol has version support you can do this. If it was forgotten in the architecture then the change is more disruptive or very inefficient."

— Martin Thompson

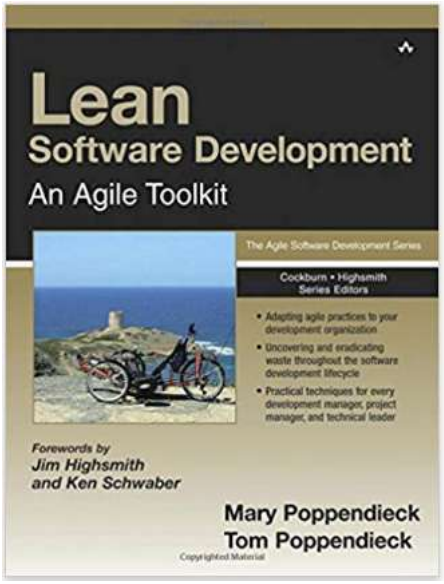
"When reversibility is important to you, that's part of your context. The decision section should state what you're doing in light of that context. (Pilot project, wrapping interface, abstraction layer, etc.)

— Michael Nygard

When? Last Responsible Moment

"the last responsible moment [:] the moment at which failing to make a decision eliminates an important alternative."

— Mary and Tom Poppendieck



Last Responsible Moment

Jeremy Miller on delaying decisions until the last responsible moment:
"The key is to make decisions as late as you can responsibly wait because that is the point at which you have the most information on which to base the decision."

And Jeff Atwood:
"Deciding too late is dangerous, but deciding too early in the rapidly changing world of software development is arguably even more dangerous. Let the principle of Last Responsible Moment be your guide."

Source: https://blog.codinghorror.com/the-last-responsible-moment/

Eb Rechtin and Mark Maier:
"Build in and maintain options as long as possible in the design and implementation of complex systems. You will need them."

Some wry? @nonspecialist@aus.social got you:

- "any decision you make now will be wrong
- you have to make a decision now, or things will be worse
- if you don't make a decision, it will be made for you and you'll have to live with the consequences"

*"delay commitment until the **last responsible moment, that is, the moment at which failing to make a decision eliminates an important alternative**. If commitments are delayed beyond the last responsible moment, then decisions are made by default, which is generally not a good approach to making decisions."*

— Mary and Tom Poppendieck

YouArentGonnaNeedIt (often abbreviated YAGNI, or YagNi on this wiki) is an ExtremeProgramming practice which states:

> "Always implement things when you *actually* need them, never when you just *foresee* that you need them."

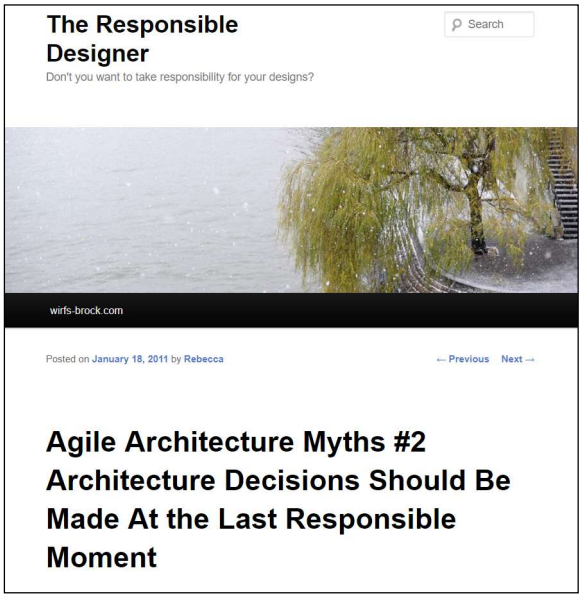
http://c2.com/xp/YouArentGonnaNeedIt.html

Earliest Responsible Moment

"I prefer to make decisions when they have positive impacts. Making decisions early that are going to have huge implications isn't bad or always wasteful. Just be sure they are vetted and revisited if need be."

— Rebecca Wirfs-Brock

"I prefer calling that opportune moment of when it is reasonable to decide, the Most Responsible moment ... as it is based on your judgment of the context, the situation, the risks, and everyone impacted by that decision." — Rebecca Wirfs-Brock



Creating Ground Under the Feet

Strategy and architecture decisions create context for further decisions, establishing relationships, and reducing the decision space. This is good. It reduces the overload of overwhelming ambiguity and uncertainty, by narrowing the space and putting stakes in the ground. Now we can probe and test, to see how we're doing. We make certain key decisions early, to "put ground under our feet." Huh? Ground? Metaphorically speaking, but to be able to move forward, we have to start to shape the space, gain traction. More metaphors.

We have to decide what we are going to do (next, and at all, and if we want to be proactive about cohesive and concerted action, where we are headed), and how. We may make ad hoc decisions implicitly on the fly without considered reflection, but some of our decisions are going to cleave the design space, ruling some opportunities out. This will be true whether they are implicit or explicit, considered, reasoned and probed, or made on the fly on guesses or without even knowing there were other choices we could have made. Better, if we anticipate they'll be highly consequential, if well considered.

"I believe that you can and should look ahead. And that most developers, given half a chance, are pretty good at incorporating past experiences and making anticipatory design choices."

– Rebecca Wirfs-Brock

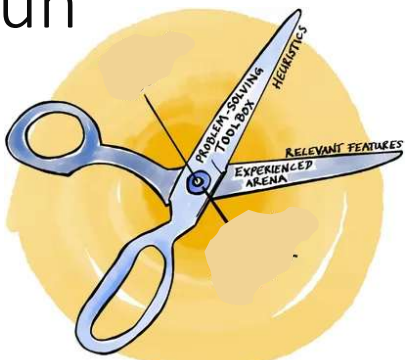
You know the adage: "What's the best time to plant a tree? 20 years ago. What's the second best time? Now." Well, that's true, unless we don't need a tree. And there isn't something more critical to do now. But the point is important too — trees can't be moved so they constrain and set context for other landscaping decisions and they take a long time to grow, so to have the benefit of a bigger tree, we need to start as soon as we can.

As Mayoor Salva pointed out, opportunity cost is a useful concept to draw on here.

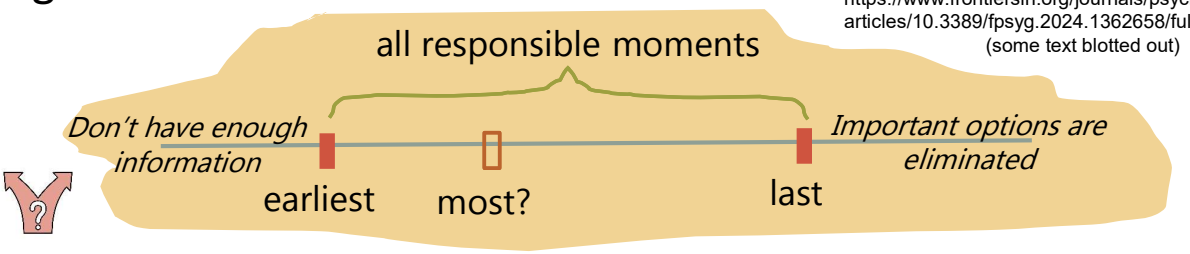
When? Judgment, again huh

The point isn't that we *know* what is earliest, last, and most

It's that we explore what we gain and risk



Simon's scissors Image source: Jaeger et al https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2024.1362658/full (some text blotted out)



When? Think About It

If there is some time frame in which we can "responsibly" make (significant) design decisions, there is some "earliest" and some "last" "responsible moment" — conceptually, anyway. We don't *know* where those points are, but the point is more about (for significant decisions) exploring (just enough) the tradeoffs… of earlier benefit from the decision (being put into play) versus knowing more later, and retaining degrees of freedom longer. What depends on the decision, and is held up? What is risky to defer, or to move forward on without learning more?

"What skills would we need in order to *not* have to make this decision until later?" (Kent Beck)

And! What should we bring forward, and for what reason? And some of those reasons are engineering reasons and some are market/user facing reasons. So what skills do we need to develop, to think strategically about the difference that makes a difference here? (Where "strategically" is relative to the scope at which we are designing.)

"To make sense of such an ill-defined and open-ended world — in order to survive, thrive, and evolve — the organism must first realize what is relevant in its environment. It needs to solve the problem of relevance." — Johannes Jaeger et al, 2024 (https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2024.1362658/full)

What do we pull forward because it underpins, and what do we push out, because we need to learn more, etc….

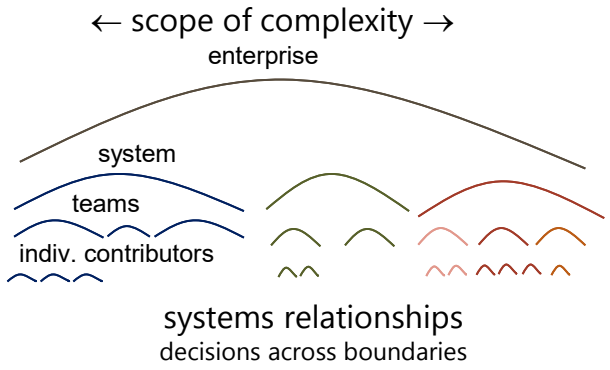
"Oh yeah, this is a golden year for least responsible decisions."

— Einar W. Høst

Significant: Impact Across Groups

Strategically or architecturally significant:

Enables something critical (to what value we deliver and how we do so over time)



TECHNICAL DECISIONS

Leadership Across (Scopes of Complexity)

The management hierarchy is an accountability hierarchy in the contractual, fiduciary and financial sense. It manages people but also resources, like getting funding early on, and allocating budgets across priorities, including new business creation, later on. Obvious, and yet we can overlook both the importance, work and attention required, and the stresses involved, in being responsible for keeping salaries paid, investors and boards satisfied, and making choices where outcomes may only be fully visible years ahead.

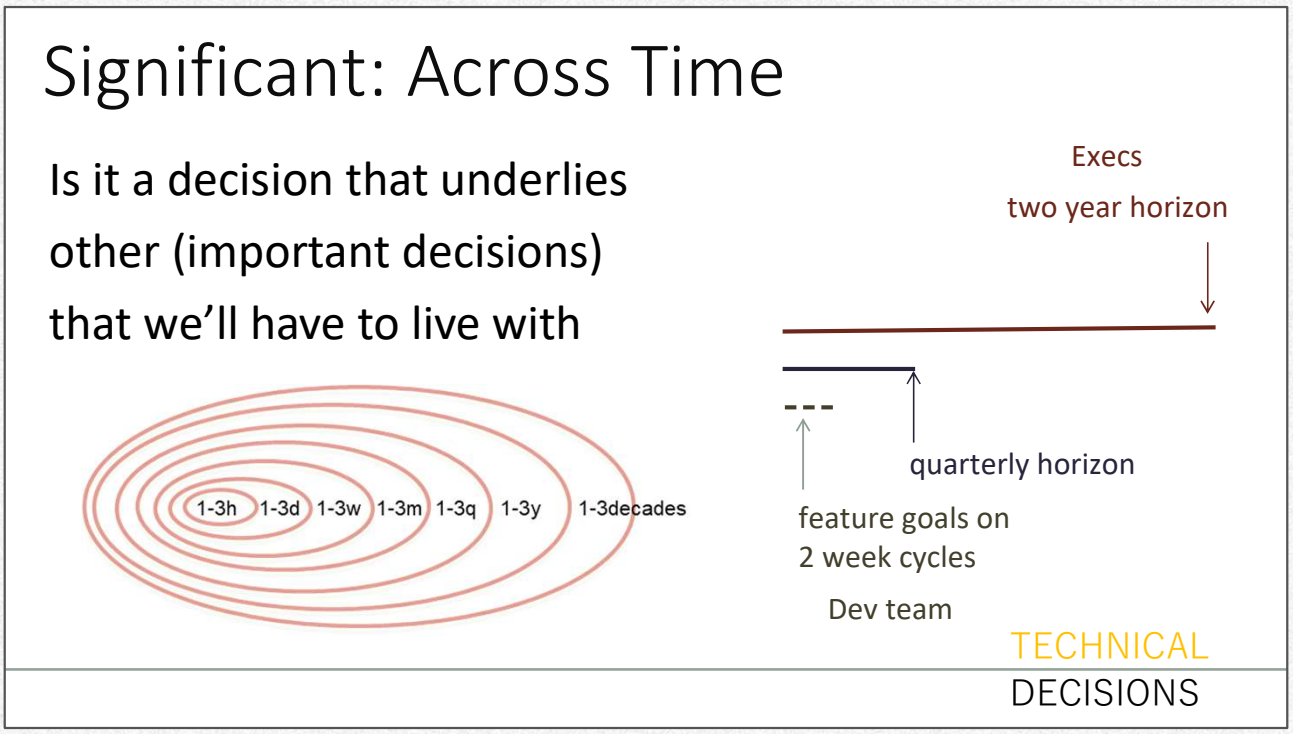
It is also a part of the (broader) communication and co-ordination network. Influence networks, or informal relationships, facilitate communication, creating alternate pathways in the organization, and can help to get cross-boundary things done with less bureaucracy. They may be largely invisible (the kind of thing where it would take many interviews to map the influence network out, and still miss much) until they kick into higher gear to effect or impede change.

There's also the network of relationships in place to get work done. We're going to focus on complex systems built, evolved and operated by several, or even many, teams. Some of the system spanning work is reflected in the management hierarchy; some in technical roles that span, like architect roles; some is (ad hoc) "glue" work. As the span of responsibility increases in scope, from responsibility for some local part of a system, to responsibility across subsystems and systems, the compass (span) of complexity in technical and organizational terms increases, and the demands on mastery shifts.

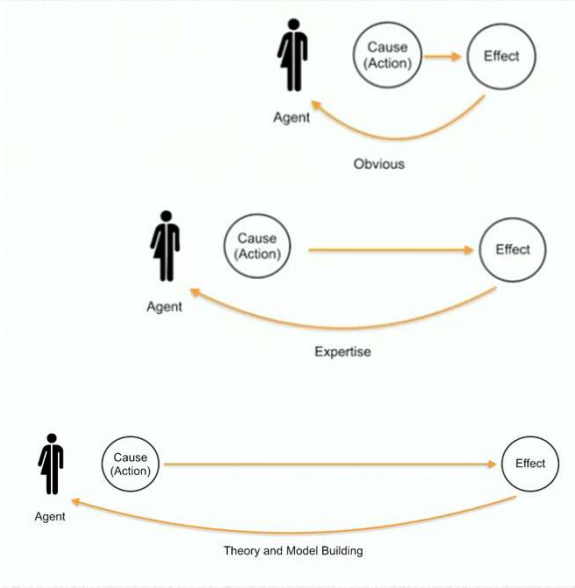
Back to hierarchy for a moment, and a couple of points from the Jo Freeman classic ("The Tyranny of Structurelessness"): "Contrary to what we would like to believe, there is no such thing as a 'structureless' group." and "The structure may be flexible, it may vary over time. It may evenly or unevenly distribute tasks, power and resources over the members of the group." An explicit hierarchy is visible, and hence can be worked on, to make it more inclusive and more about leadership (power-with rather than dominance and power-over).

> *"the scales of information, people, time horizons and information all changes. As a result so does the impact." — Nivia Henry*

Seniority relates to scope and time span of work, with increasing complexity and uncertainty associated with more seniority



Leadership and Time/Scope



Source: Jabe Bloom, Whole Work: Sociotechnicity & DevOps

Seniority generally comes with increasing scope of concern (across systems, of systems, and, with more seniority, more impact on ecosystems). Increases in scope mean we're with dealing with greater complexity, and need expertise and experience that is rooted in the technical but is increasingly strategic and organizational. And we're dealing with longer time horizons, so more uncertainty.

Elliott Jaques' concept of time span of discretion/span of complexity provides a way to talk about roles and decision span. Those with shorter time span of discretion (and more narrowly scoped decision frames) are making decisions with more immediate impact and conscribed decision autonomy (e.g., the time horizon for completion of work made visible to others on the team or management, may be days or weeks). More senior roles are paying attention to longer term outcomes, across more of the organization. More hinges on what decisions are made, and not. All of these different scopes of concern take attention and cognitive bandwidth, and demand experience and expertise, but the focus shifts from more immediate observable effects, to making judgment calls under greater uncertainty and complexity. (That said, the essentialism aspect of Jaques work is⋯ hard pass.)

While "time span of discretion" flavors the concept with what decisions we have discretion over or power to effect, Yvonne Lam draws attention to what timespan infuses our work and so draws/shapes our focus: "different entities (orgs, roles, etc.) have a span of time in which they can effect change, so that's the span of time to which they tend to pay attention." What I'm attending to, shapes what I perceive and attend to.

"thought about it as time travel: the higher up you go, the more you live in the future. As a senior eng you live 1-2 sprints out. A manager, 1-3 months. A director 3-9 months and so on. " — Danielle Leong

System Knowledge

"understanding of complex systems is distributed"

— Chris McDermott

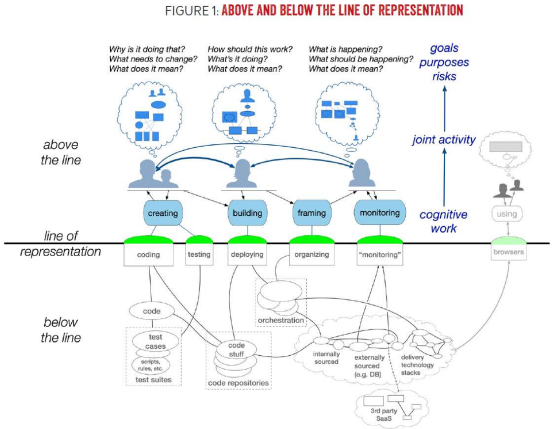


Image by: Richard I. Cook, in ACM Queue, 2020 (and many talks, the STELLA Report, etc.)

Image source: Above the Line, Below the Line, by Richard I. Cook, https://queue.acm.org/detail.cfm?id=3380777

Imperfect and Incomplete

Indeed, when it comes to complex systems, as Chris McDermott observed in a talk a while back: "understanding of the system is distributed." None of us understand it all, in breadth and depth, and even if we did or could, circumstances are ever changing, the environment is changing, the system is changing. And so is understanding. David Woods observed (and it is known as Woods' Theorem): "As the complexity of a system increases, the accuracy of any single agent's own model of that system decreases rapidly." (STELLA Report)

"The people engaged in observing, inferring, anticipating, planning, troubleshooting, diagnosing, correcting, modifying and reacting to what is happening are shown with their individual mental representations. These representations allow the people to do their work -- work that is undertaken in pursuit of particular goals. To understand the implications of their actions requires an understanding of the cognitive tasks they are performing and, in turn, an understanding of what purposes those cognitive tasks serve.

The green line is the line of representation. It is composed of terminal display screens, keyboards, mice, trackpads, and other interfaces. The software and hardware (collectively, the technical artifacts) running below the line cannot be seen or controlled directly. Instead, every interaction crossing the line is mediated by a representation. This is true as well for people in the using world who interact via representations on their computer screens and send keystrokes and mouse movements.

A somewhat startling consequence of this is that what is below the line is inferred from people's mental models of The System.

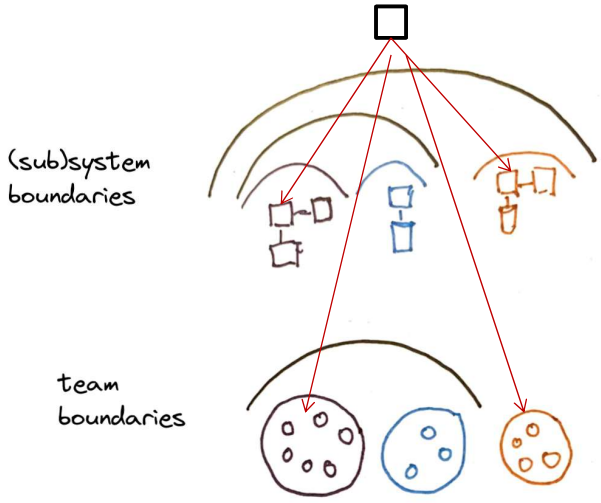
This is not to say that what is below the line is imaginary. But the artifacts there cannot be perceived or manipulated directly. Instead, people use mental models of what, although hidden, they infer must be there to interpret what they see on the screens and to predict what the effect of typing a character or clicking a mouse will be." from The STELLA report, https://snafucatchers.github.io/

Decisions via Advice Process

The Rule: anyone can make an architectural decision.

The Qualifier: before making the decision, the decision-taker must consult two groups: The first is everyone who will be meaningfully affected by the decision. The second is people with expertise in the area the decision is being taken.

— Andrew Harmel-Law,
in "*Scaling the Practice of Architecture, Conversationally*"



See also: The Advice Process, https://reinventingorganizationswiki.com/en/theory/decision-making/

Conversations and Learning

We have talked about Architecture Decision Records as a device for leading our thinking – directing us to consider contexts, impact and consequences. That is, implicitly drawing on and refining our Theory of the Problem, Theory of the Solution, and Match between.

We (as an industry) have also seen something of a return to the RFC (request for comment), as a mechanism to involve more people in architecture decision making. This may take the form of an ADR adding a space for comments (and in-flight status). One of the ways to build expertise, is to draw on expertise, perspectives, alternative points of view and experience.

Andrew Harmel-Law has pushed this practice further, working as an architect with Thoughtworks clients and in an article he wrote, hosted on Martin Fowler's site, and a book that is in early release. In what he calls the Advice Process, anyone can make an architectural decision, but they must get advice from 1. those affected or impacted by the decision, and 2. those who have expertise relevant to the decision.

This is a process to make better decisions, and foster and build system understanding and design expertise.

Architecture decisions impact across boundaries.

In Andrew's words: "while decision-takers are in no way obliged to agree with the advice the folks in these two consulted groups give them, they must seek it out, and they must listen to and record it. We are not looking for consensus here, but we are looking for a broad range of inputs and voices."

And later, "It will come as no surprise to learn that consequently, a series of ADRs, and their surrounding conversations provide an excellent learning ground for people wanting to begin to take on the task of decision-taking; everything is out in the open, including the dissent and compromise-making. Less experienced practitioners of architecture can peruse the history of what went before them quickly and easily, see good (and quite likely less-good) examples, and see decisions being taken (and perhaps also being revoked when circumstances change / the team learned more). They are almost a thinking and decision lore for a set of software, written in the hand of those who contributed most to it."

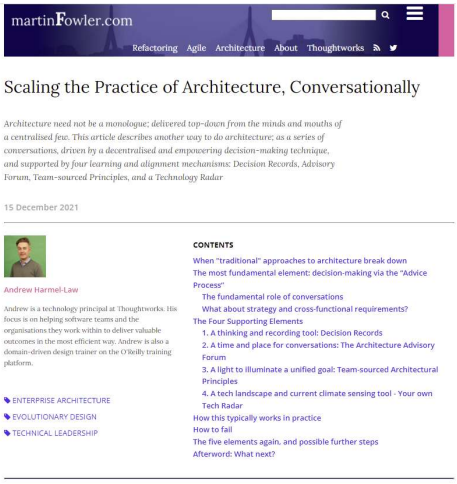
— Andrew Harmel-Law, Scaling the Practice of Architecture, Conversationally, 2011

Note: Where we say architecture, we could also add in other strategically significant decisions; that is, decisions that impact identity and value.

Architecture Advisory Forum

"the invitees to the AAF are those typically affected / possessing relevant expertise."

— Andrew Harmel-Law


martinFowler.com

Refactoring Agile Architecture About Thoughtworks

Scaling the Practice of Architecture, Conversationally

Architecture need not be a monologue; delivered top-down from the minds and mouths of a centralised few. This article describes another way to do architecture; as a series of conversations, driven by a decentralised and empowering decision-making technique, and supported by four learning and alignment mechanisms: Decision Records, Advisory Forum, Team-sourced Principles, and a Technology Radar

15 December 2021

Andrew Harmel-Law

Andrew is a technology principal at Thoughtworks. His focus is on helping software teams and the organisations they work within to deliver valuable outcomes in the most efficient way. Andrew is also a domain-driven design trainer on the O'Reilly training platform.

ENTERPRISE ARCHITECTURE
EVOLUTIONARY DESIGN
TECHNICAL LEADERSHIP

CONTENTS

When "traditional" approaches to architecture break down
The most fundamental element: decision-making via the "Advice Process"
The fundamental role of conversations
What about strategy and cross-functional requirements?
The Four Supporting Elements
1. A thinking and recording tool: Decision Records
2. A time and place for conversations: The Architecture Advisory Forum
3. A light to illuminate a unified goal: Team-sourced Architectural Principles
4. A tech landscape and current climate sensing tool - Your own Tech Radar
How this typically works in practice
How to fail
The five elements again, and possible further steps
Afterword: What next?

From: Andrew Harmel-Law, Scaling the Practice of Architecture, Conversationally, 2021

Structured Conversations and Learning: Architecture Advice Forum

"The second supporting element in this alternative approach exists to make all the conversations supporting this advice-seeking easier: a weekly, hour-long Architecture Advisory Forum ("AAF").

Fundamentally, this is a regular and recurring place and time for conversations. Your ideal attendees are delegates from each team as well as your key representatives from your Advice Process checklist. However, the invite should remain completely open to encourage transparency and openness. The timeliness and quality of the conversations which take place is a key indicator of success, but equally important is the breadth and diversity of views shared, and the same goes for the contributors. If architecture is being "done" here, and lessons shared and learned, then you're winning."

"Firstly, the Advice Process reigns. Decisions taken to the AAF are still owned and made by the originators. The only thing other attendees can do is offer advice, or suggest additional people to seek advice from. Hence the name.

This brings us to the second key difference. Given the Advice Process qualifiers, the invitees to the AAF are those typically affected / possessing relevant expertise. This means those typically present include representatives from each feature team (and not just the lead; BAs/POs and QAs are frequently present), people from other programmes of work, UX, Product, Operations, and occasionally senior execs."

— Andrew Harmel-Law

Leadership is very much about ensuring that conversations that need to be happening are happening — not always initiating them, nor always helping to focus or navigate them, but ensuring they do happen and guiding when needed

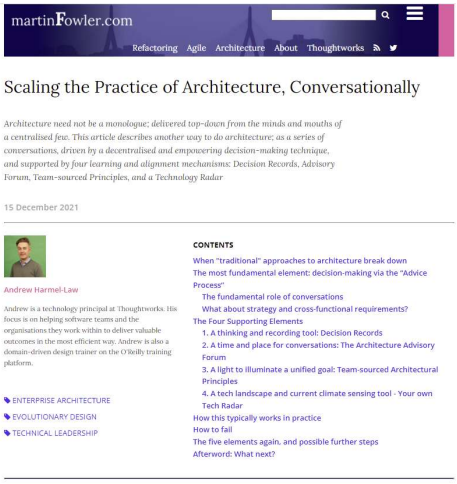
Of note: in this model, architects also use the advice process

Bring more voices and experience to the decision process, through advice gathering

Architecture Advisory Forum

"When [the conversations] take place in an AAF there is an audience, so many people can listen and everyone can learn. The amount of organisational, domain, legacy, and experiential information and architectural skill-deployment shared at these sessions is unlike anything I have ever seen"

— Andrew Harmel-Law


martinFowler.com

Refactoring Agile Architecture About Thoughtworks

Scaling the Practice of Architecture, Conversationally

Architecture need not be a monologue; delivered top-down from the minds and mouths of a centralised few. This article describes another way to do architecture; as a series of conversations, driven by a decentralised and empowering decision-making technique, and supported by four learning and alignment mechanisms: Decision Records, Advisory Forum, Team-sourced Principles, and a Technology Radar

15 December 2021

Andrew Harmel-Law

Andrew is a technology principal at Thoughtworks. His focus is on helping software teams and the organisations they work within to deliver valuable outcomes in the most efficient way. Andrew is also a domain-driven design trainer on the O'Reilly training platform.

ENTERPRISE ARCHITECTURE
EVOLUTIONARY DESIGN
TECHNICAL LEADERSHIP

CONTENTS

When "traditional" approaches to architecture break down
The most fundamental element: decision-making via the "Advice Process"
The fundamental role of conversations
What about strategy and cross-functional requirements?
The Four Supporting Elements
1. A thinking and recording tool: Decision Records
2. A time and place for conversations: The Architecture Advisory Forum
3. A light to illuminate a unified goal: Team-sourced Architectural Principles
4. A tech landscape and current climate sensing tool - Your own Tech Radar
How this typically works in practice
How to fail
The five elements again, and possible further steps
Afterword: What next?

From: Andrew Harmel-Law, Scaling the Practice of Architecture, Conversationally, 2021

Conversations with "the System in the Room"

The Architecture Advisory Forum convenes conversations about key system decisions with "the system in the room" – if not the full system, folk who bring in the perspective of relevant, or impacted parts of the system. (User research will go beyond this, too.)

Over time, these conversations increase the repertoire of, or at least exposure to, a variety of system design elements and approaches, and develop and nuance each person's system understanding, as various people grapple with the interaction between their perspectives, assumptions, concerns and understandings, and those of others across the various sociotechnical system spaces involved.

I again want to point out that I am referring to Andrew Harmel-Law's work (which is specifically in the architecture context) here, but we might consider extending advice gathering to other strategically important areas (such as strategic aspects of product design, or SRE and platform engineering, usability, etc.).

Now, I want to return to André Henry's point (in the quote alongside). Some of our decisions have impacts and consequences and implications outside our of our extended team's collective experience (even as we strive to be more inclusive). For decisions of consequence to the people and ecosystems that our system impacts, we need to extend our consequence scanning and advice gathering, so that we are drawing on broader demographic and geographic, experience and can anticipate adverse impacts. Further,

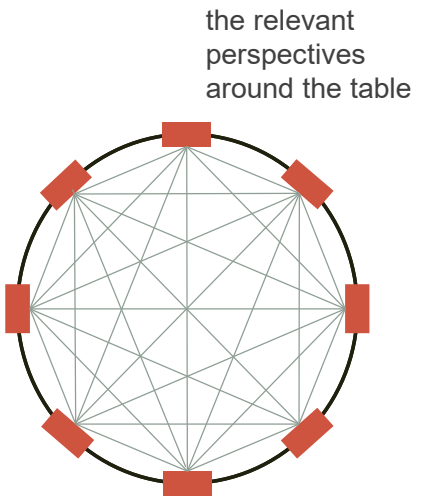
"you also have to get to market and this is where the experience comes in" — André Henry

"So much of this comes from experience, and seeing the consequences of decisions and learning from others."

— André Henry

Sense Making Round Table

Much of what we do, requires understanding (sensing and sense making) that none of us individually hold. And our perspectives (formal role related and individual) interact.



Sense Making Round Table Source: Dana Bredemeyer

*"information is a flow of messages, while knowledge is created and organized by the very flow of information, anchored on the commitment and beliefs of its holder. This understanding emphasizes an essential aspect of knowledge that relates to human action." — Ikujiro Nonaka**

"Access to information and limiting access to information is a very old strategy for trying to push forward one political agenda or one agenda or another"

— John Green

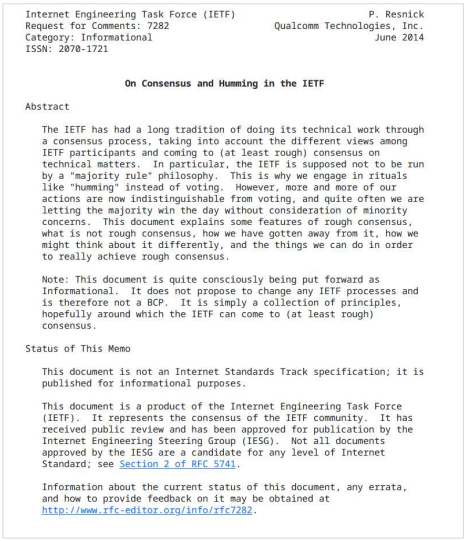
Understanding, Diagnosing and Acting as Joint Activity

Complexity (in our work, our organizations, and the various contexts we're impacting with the systems we're evolving) means we're seeking to act in contexts that are mixes of stable and shifting, even dramatically. Where knowledge applies, but is distributed – among different people, and in relevance. Yet we need to act, and bring about requisite coherence (in our organization, and systems, and systems in contexts). One way to do that, is to bring those who have relevant experience and perspective together to jointly make sense of the situation, and response. So "leadership is about creating contexts" is about fostering contexts for joint (situation) understanding or sense making, and in so doing fostering understanding that informs decisions and gives collaborative impetus to action.

* Ikujiro Nonaka, *Dynamic Theory of Organizational Knowledge Creation*

Consensus?

"Engineering always involves a set of tradeoffs. Its almost certain that any time engineering choices need to be made, there'll be options that appeal to some people, but [not] others. In determining consensus, the key is to separate choices that are simply unappealing from those that are truly problematic. If at the end of the discussion some people have not gotten the choice they prefer, but they have become convinced that the chosen solution is acceptable[..]they have still come to consensus." — P Resnick



Internet Engineering Task Force (IETF) P. Resnick
Request for Comments: 7282 Qualcomm Technologies, Inc.
Category: Informational June 2014
ISSN: 2070-1721

On Consensus and Humming in the IETF

Abstract

The IETF has had a long tradition of doing its technical work through a consensus process, taking into account the different views among IETF participants and coming to (at least rough) consensus on technical matters. In particular, the IETF is supposed not to be run by a "majority rule" philosophy. This is why we engage in rituals like "humming" instead of voting. However, more and more of our actions are now indistinguishable from voting, and quite often we are letting the majority win the day without consideration of minority concerns. This document explains some features of rough consensus, what is not rough consensus, how we have gotten away from it, how we might think about it differently, and the things we can do in order to really achieve rough consensus.

Note: This document is quite consciously being put forward as Informational. It does not propose to change any IETF processes and is therefore not a BCP. It is simply a collection of principles, hopefully around which the IETF can come to (at least rough) consensus.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7282.

https://datatracker.ietf.org/doc/html/rfc7282

Consensus

"Having full consensus, or unanimity, would be ideal, but we don't require it: Requiring full consensus allows a single intransigent person who simply keeps saying "No!" to stop the process cold."

"Lack of disagreement is more important than agreement"

"But _determining_ consensus and _coming to_ consensus are different things than _having_ consensus."

"the group must have honestly considered the objection and evaluated that other issues weighed sufficiently against it. Failure to do that reasoning and evaluating means that there is no true consensus."

Source: https://datatracker.ietf.org/doc/html/rfc7282

Another great read: Feminist Perspectives on Argumentation by Catherine E. Hundleby https://plato.stanford.edu/entries/feminism-argumentation/

Better Decisions

Through:

- Identifying the decision factors, explicitly considering the various contexts (use, operations, development, .., ecosystem)
- Writing and clarifying the decision reasoning using an RFC, ADR, etc.
- Getting advice and input via an advice or RFC process
- Conversation with the "system together in the room" in Advice Forum or similar

which over time, across decisions, deepens system understanding

What else?

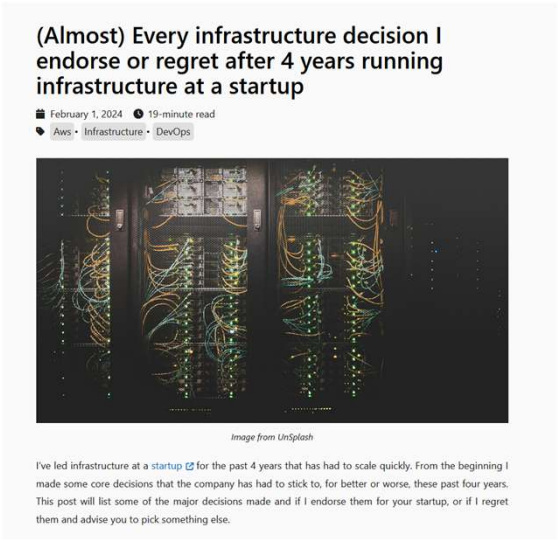
So Far

To recap and reinforce, we have looked at better decisions through the frames of decision "anatomy," directing our attention at what we're taking into account, and in particular reminding us that forces are not just forces in the technical context, shaping developer experience and technical outcomes. We need to look outwards to effects and side-effects, to forces and desired outcomes, arising in the user space and other partnering parts of the organization, and beyond.

Documenting decisions with some version of an ADR shares this reasoning, and an RFC or advice process encourages input and advice. The better the decision process – where it gathers input, what it seeks to understand in terms of impact and consequences, the options created and deliberated, and so forth – the more the decision record itself serves to teach design reasoning and increase the design repertoire of decision makers and decision readers and implementers.

An Advice Forum, or similar, brings "the system" (representative parts of it) together in conversation, and over time builds and deepens understanding.

Do This! (Reflect on Decisions)



(Almost) Every infrastructure decision I endorse or regret after 4 years running infrastructure at a startup

February 1, 2024 · 19-minute read

Aws · Infrastructure · DevOps

Image from UnSplash

I've led infrastructure at a startup for the past 4 years that has had to scale quickly. From the beginning I made some core decisions that the company has had to stick to, for better or worse, these past four years. This post will list some of the major decisions made and if I endorse them for your startup, or if I regret them and advise you to pick something else.

EKS

Endorse

Unless you're penny-pinching (and your time is free), there's no reason to run your own control plane rather than use EKS. The main advantage of using an alternative in AWS, like ECS, is the deep integration into AWS services. Luckily, Kubernetes has caught up in many ways: for example, using external-dns to integrate with Route53.

EKS managed addons

Regret

We started with EKS managed addons because I thought it was the "right" way to use EKS. Unfortunately, we always ran into a situation where we needed to customize the installation itself. Maybe the CPU requests, the image tag, or some configmap. We've since switched to using helm charts for what were add-ons and things are running much better with promotions that fit similar to our existing GitOps pipelines.

Process

Automating post-mortem process with a slack bot

Endorse

Everyone is busy. It can feel like you're the "bad guy" reminding people to fill out the post-mortem. Making a robot be the bad guy had been great. It streamlines the process by nudging people to follow the SEV and post-mortem procedure.

It doesn't have to be too complex to start. Just the basics of "It's been an hour of no messages. Someone post an update" or "It's been a day with no calendar invite. Someone schedule the post-mortem meeting" can go a long ways.

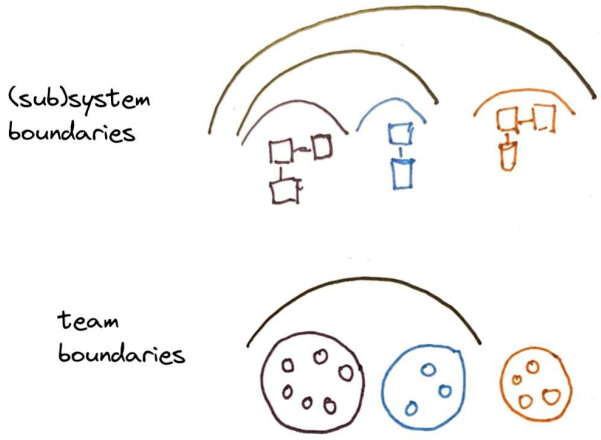
Source: Jack Lindamood
https://cep.dev/posts/every-infrastructure-decision-i-endorse-or-regret-after-4-years-running-infrastructure-at-a-startup/

Reflect on Decisions as a Way to Share Learning

Jack Lindamood's post reflecting on technology and process decisions (and giving them a regret/endorse/etc. rating) is useful (even if your experience may differ, given your context, etc.).

Local vs System

what we're paying attention to, shapes what we perceive and pay attention to



Local Concerns and Autonomy

So, this is good. Conversations are happening. And people are being brought into the decision informing, decision shaping, decision understanding process. Better decisions. Better socialization of the decisions, as now more people are part of the various conversations considering the decision, its approach, alternatives, tradeoffs and ramifications.

However. Focus builds focus. What we are paying attention to, shapes what we perceive and pay attention to.

At any rate, with focus and autonomy, we need to pay attention to, put work into, the system as a system, where decisions that impact what the system is and is becoming, are made with the system in mind.

Where "the" system is not quite so simple as "a" system. It's not just that it's a sociotechnical system, or a system of systems. We're creating systems for users (impacting their sociotechnical systems) within systems (the economic and sociotechnical systems of our design-development and operations). This brings into view the various boundaries that technical leadership and system design is spanning. Within our organization, and the various teams involved. As well as of our system and what capabilities it takes on, and shares, with users and how that impacts their value flows.

And working across these boundaries mean that we, as leaders in tech, are spanning organizational gaps – that would otherwise be gaps in perception and anticipation. We notice the need for, and create the context for, (participative ensemble) work that needs to be done at the system level.

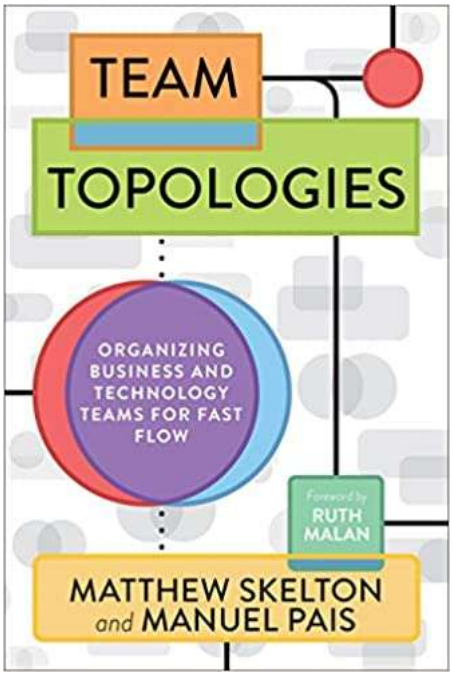
> *"Don't ever stop talking about the system."*
>
> *— Eb Rechtin*

Team Topologies

"The limit of what a team can do is cognitive capacity."

"If we stress a team by giving it responsibility for part of the system that is beyond its cognitive load capacity, it ceases to act like a high-performing unit and starts to behave like a loosely associated group of individuals, each trying to accomplish their individual tasks without the space to consider if those are in the team's best interest."'

— Matthew Skelton and Manuel Pais



Cognitive Load

"When we talk about cognitive load, it's easy to understand that any one person has a limit on how much information they can hold in their brains at any given moment. The same happens for any one team by simply adding up all the team members' cognitive capacities."

"When cognitive load isn't considered, teams are spread thin trying to cover an excessive amount of responsibilities and domains. Such a team lacks bandwidth to pursue mastery of their trade and struggles with the costs of switching contexts."

— Matthew Skelton and Manuel Pais

Jessica Kerr: "When a team's cognitive load is too high, coherence dissolves. You get a group of individuals instead"

'Cognitive load was characterized in 1988 by psychologist John Sweller as "the total amount of mental effort being used in the working memory."'

— Matthew Skelton and Manuel Pais

Coordination Costs

If there is interaction between (microservices, modules, ..) components (and hence an interface), work has gone into enabling that interaction. It may be communication and coordination between the teams involved, or may be interface design work that's been done, and that the teams agree to adhere to. It may be that the components publish and subscribe to events, but those are designed. Somewhere, that coordination cost is born as design attention and design communication (which may be code). If born by only one side of the interface, the other side bears the cost of accommodating to its constraints.

Coupling enables interactions to build capabilities, but comes with co-ordination costs. Communicating across contexts means understanding the needs and trade-offs across those boundaries, and the synthesis and emergence sought. This may mean context switching away from the focus of the team's work on the component(s) they are responsible for, increasing cognitive load which we seek to reduce.

Put Work into Common Ground

"requires continuing effort to sustain, extend, and repair common ground."

— Richard Cook



Image from: *Unflattening,* by Nick Sousanis

Common Ground Takes Effort

"Joint activity assumes a basic compact, which is an agreement (often tacit) to facilitate coordination and prevent its breakdown." — Gary Klein et al.

Gary Klein and collaborators, working on joint activity (and systems development and evolution is joint activity), indicate that joint activity depends on interpredicatability, and interpredictability depends on a "common ground" of shared knowledge, beliefs and assumptions. And Richard Cook (quote on slide), who you might recognize from the classic "How systems fail," is reminding us that common ground takes effort – effort to build, effort to sustain and extend, and effort to repair.

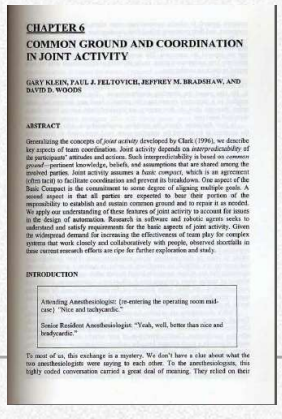
Byte-Size Architecture Sessions

Andrea Magnorsky created a practice area around something many of us maybe sorta do, but should do more, and more consciously. Those whiteboard (or Miro board) discussions we pull others into, to noodle on some part of the system? Those Architecture Community of Practice meetings? Extend/modify that idea to intentionally and regularly get folk talking about and modeling "the system" and architecture. Building understanding of this system, together. Addressing issues, together. And doing so at a regular cadence (like an hour a week) with just enough structure, to build shared understanding, to investigate and learn together, to explore options, and more. Make it an ongoing practice.

https://bytesizearchitecturesessions.com/

"Collective responsibility is not a matter of adding but of interweaving, a matter of the reciprocal modification brought about by the interweaving. It is not a matter of aggregation but of integration."

— Mary Parker Follett



CHAPTER 6

COMMON GROUND AND COORDINATION IN JOINT ACTIVITY

GARY KLEIN, PAUL J. FELTOVICH, JEFFREY M. BRADSHAW, AND DAVID D. WOODS

ABSTRACT

INTRODUCTION

(Co)Stewardship

"I got reminded [..] of the notion of stewardship, and we should probably think about our successful systems in these terms – with responsibility for the connections, continuity, and health of the system and the people impacted by it. And part of that stewardship should be – *needs* to be – an engagement in a sustained renewal of necessary expertise."

– Michael McCliment

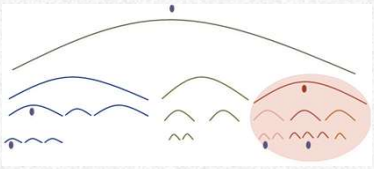
Source: https://twitter.com/cornazano/status/1292967631990018049

Stewardship

This brings to mind the notion of stewardship and the sustained building and renewal of the expertise it takes to be good stewards of the system – stewards who are convening conversations and growing expertise and contextualizing work so that we're bending the arc of our system more towards (structural, conceptual, design, organizational) integrity, and sustainability in technical, economic, social and environmental terms.

Roles that have "across" organizational unit (teams, teams of teams; managers, architects, managers of managers, senior SREs and principal engineers, CTOs, etc.) focus, each have a unique responsibility field, with a unique relationship and perceptual field, putting them in a position to develop understandings of the system (and interacting systems) at this scope. From this unique across-boundaries vantage point, we scan for and discern what needs to be understood and responded to, to sustain the system. Anticipatory awareness means paying ongoing attention to understanding the various contexts (use, development, operations, business, technology,⋯), the system, and the implications of changes and shifts. This is work. Collaborative work. Integrative work. Still, work. As is co-shaping and responding to shifts.

Anticipatory awareness and intentional design, both as ongoing processes as the system and its contexts co-evolve, mean doing hard things that have outcomes and consequences. And stewardship also means building the organizational will and wherewithal to confront, understand, cope with, and co-create and co-evolve towards better. Even as we seek to understand and shape what we mean by "better."



Each system, and its intertwingling in other systems, is unique and (co-)evolving.

We have a unique responsibility and opportunity to understand, to recognize, to draw out.

Ethics and Oughts



Jabe Bloom
@cyetain

Thinking about, thinking about, the differences between these...



"the designer, is concerned with how things ought to be - how they ought to be in order to attain goals, and to function. "
— Herbert Simon

We lead to enable things to be more the way they ought to be.

Intention, and Oughts

"The rocks on the right where (presumably) assembled by "objective" means, no human intervened, physical explanations should be able to capture how it "got like that. " [T]he rocks on the left... OTOH... can't VIOLATE physical laws... but their assembly includes a good deal of human judgment" — Jabe Bloom (@cyetain)

Just because we can, doesn't mean we should. But there is a lot to discern, and forces and pressures.

Consider rock stacking in river beds. "Rock stacking can be detrimental to the sensitive ecosystems of rivers and streams. Moving rocks from the river displaces important ecosystem structure for fish and aquatic invertebrates. Many of our Ausable River fish species lay eggs in crevices between rocks, and moving them can result in altered flows, which could wash away the eggs or expose the fry to predators. Salamanders and crayfish also make their homes under rocks, and rock moving can destroy their homes, and even lead to direct mortality of these creatures." (Ausable River Association website)

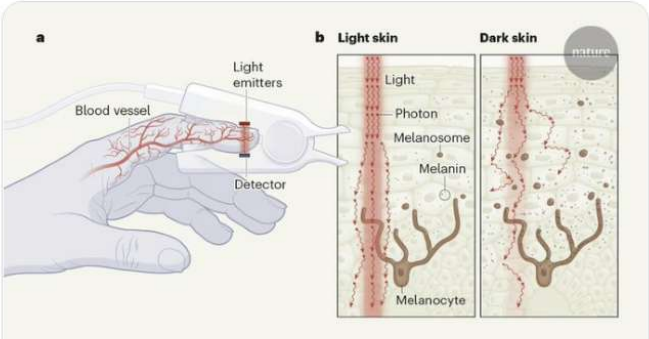
The point? Is that our impact adds up. How much should and could we think about? More than we have, typically? Yes, technology disrupts. Has positive and negative effects. Still, considerate design would have us think in terms of impacts.

Diversity on the team is important, as is doing the work to understand impact

Impact Blindspots

Blood oximeters failing to be accurate for people with darker skin, impacts diagnoses during Covid.

AirTags have been used to maliciously track women.



nature.com
Skin colour affects the accuracy of medical oxygen sensors
Nature - Two views on the pigmentation dependence of pulse oximetry.

Responsibilities and Outcomes

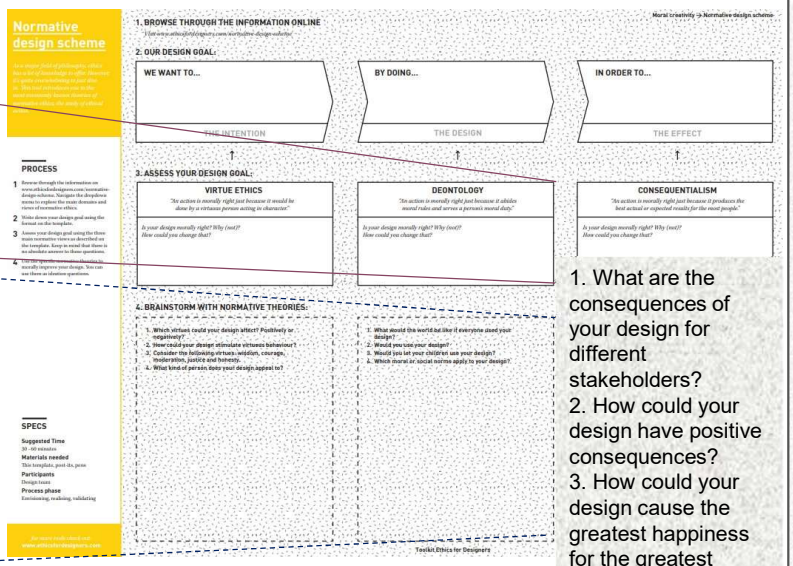
Lore has it that "The ancient Romans had a tradition: whenever one of their engineers constructed an arch, as the capstone was hoisted into place, the engineer assumed accountability for his work in the most profound way possible: he stood under the arch." (Michael Armstrong). Being around as a legacy system ages, is another way that we in software get to experience the impact of decisions made years, and even just days, ago… But our responsibilities aren't just to our future selves, in development and operations. They are to users, and communities, and other creatures and ecosystems. And it can be hard to anticipate. This is not a get out of responsibility free card. We need to foster diverse teams and partner with others to research options and impacts (to anticipate, and as we build out the system, to probe and test impacts and outcomes). It's work and part of this work is to explore for what our team and organization is set up to be blind to (e.g., strive to see from different points of view, and partner with impacted groups).

"Given any design team organization, there is a class of design alternatives which cannot be effectively pursued by such an organization because the necessary communication paths do not exist."

— Mel Conway

"Those of us developing software don't need to be told what a big impact it's had on humanity this century. I've long maintained that this places a serious responsibility on our profession. Whether asked to or not, we have a duty to ensure our systems don't degrade our society."
— Martin Fowler, in the introduction to a post on Thoughtworks' "Responsible Tech Playbook" at https://martinfowler.com/articles/2021-responsible-tech-playbook.html

Ethics for Designers



Deontology
Is your design morally right?
Why (not)?
How could you change that?

1. What would the world be like if everyone used your design?
2. Would you use your design?
3. Would you let your children use your design?
4. Which moral or social norms apply to your design?

1. What are the consequences of your design for different stakeholders?
2. How could your design have positive consequences?
3. How could your design cause the greatest happiness for the greatest number?

Source: https://www.ethicsfordesigners.com/normative-design-scheme

How Design Designs Us: Part 3 | The Ethics of Design

Leyla Acaroglu writes:

"With all this rapid post-industrial revolution growth and technological advancement, we are beginning to see the fall-out of the avoidance of a singular question: how does what we design, design us?"

"Nearly everyone I interviewed had, at some point, learnt about the systemic implications of rapid innovation and how to make better decisions; yet, most of them still passed off the responsibility of 'right' decision making to someone else. It was the boss's, client's, manufacturer's, government's, or consumer's choice that would solve the problem that their production would participate in. When everyone within a system plays this hands-off, 'that's not my problem' game, the system is very quickly riddled with externalities⋯ and a shit load of problems! This appears to be the case with the complex debate around the ethics of design and technology."

"Who is taking responsibility for the outcomes, externalities, and downright damaging impacts of our hyper-consumer, ever-changing landscape of new gadgets and virtual arenas that are coming on board at a lighting speed pace?"

— Leyla Acaroglu

Source: Leyla Acaroglu, How Design Designs Us: Part 3 | The Ethics of Design
https://medium.com/disruptive-design/how-design-designs-us-part-3-the-ethics-of-design-ca40e33f5842#.5ur28he4l

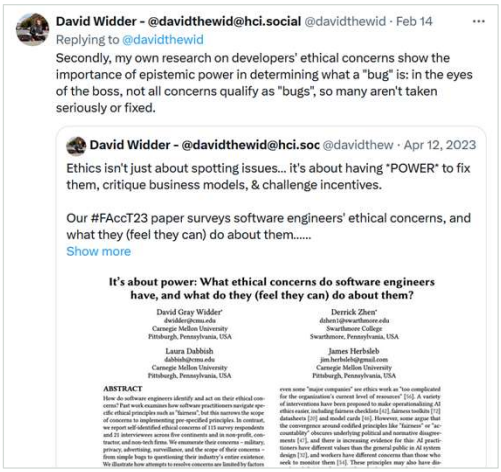
Some folk I value following, to learn more at the intersection of ethics and design: Abeba Birhane (@abebab); Alba Villamil (@albanvillamil)

Not deciding and not acting, just means other things are (implicitly) being decided and acted on …

Where Not to Act is to Act

"In the end, there are no perfect solutions to leadership and management problems [..] But because the essence of leadership is action and responsibility, one cannot *not* act."

— Peter Vaill



TECHNICAL LEADERSHIP

Hard Things Are Still Hard

Ethical matters often contain clear choices. And not so clear choices. And hard choices. A technology that takes away jobs in one area, may open up jobs or create solutions to a societal concern (treatments, etc.) in another. Not considering ethical choices, and their conflicts, doesn't make them go away (like some object permanence game).

Agnotology is culturally cultivated ignorance. Nescience is a term for constructed ignorance. Daniel R. DeNicola identified "five forms of nescience, distinguished by factors motivating a decision to barricade the boundary of knowledge: rational ignorance, strategic ignorance, willful ignorance, privacy and secrecy, forbidden knowledge." (PropCazhPM)

It can be useful to distinguish rational ignorance from strategic ignorance:

'This habit of feigning incompetence at a task, so as to make it someone else's responsibility, is called "weaponized incompetence," and can show up at work in a number of different ways.' (strategic ignorance)

'"rational ignorance" [is] ignorance we CHOOSE to retain. We make a decision that something isn't worth knowing, or the benefits of learning X would not outweigh the investment of learning X.'

— @PropCazhPM (on twitter)

"the ability to hold two opposing ideas in mind at the same time and still retain the ability to function. One should, for example, be able to see that things are hopeless yet be determined to make them otherwise."

— F. Scott Fitzgerald

Recommended:

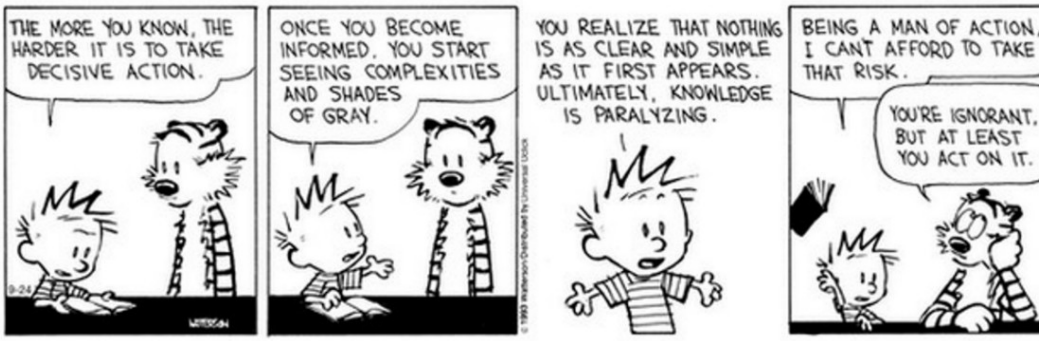
Daniel R. DeNicola, *Understanding Ignorance*

Philosophytube (Abigail Thorn) episode on *Ignorance and Censorship*

Shannon Mattern, Modeling Doubt, https://www.youtube.com/watch?v=b6ogLgWnpes&t=891s

Not deciding and not acting, just means other things are being decided and acted on ⋯

But What iF?



Watterson, Bill. *There's Treasure Everywhere: A Calvin and Hobbes Collection*. Kansas City Andrews and McMeel, 1996. Print.

TECHNICAL
DECISIONS

Hard Things Are Still Hard

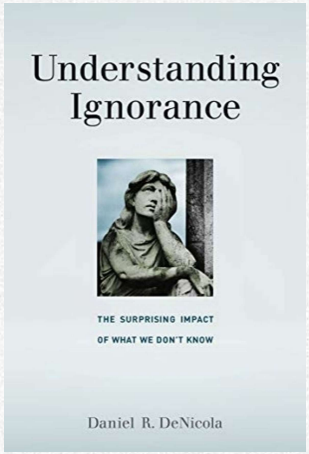
"Any adequate theory of knowledge or philosophy of education must incorporate an understanding of ignorance. Ignorance is neither a pure or a simple concept... In its house are many mansions. It is both an accusation and a defense.." [..]

"Ignorance is a scourge, but it also may be a refuge, a valve, even an accompaniment to virtue.. Ignorance is a many-splendored thing..."

— Daniel R. DeNicola, *Understanding Ignorance: The Surprising Impact of What We Don't Know*

"Now I believe I can hear the philosophers protesting that it can only be misery to live in folly, illusion, deception and ignorance, but it isn't — it's human."

— Desiderius Erasmus



Decisions, Decisions

So we have understood the context, the various interacting desired outcomes, the interacting forces, and implications, and we have developed alternatives that… all look good… 😵 NOW WHAT???

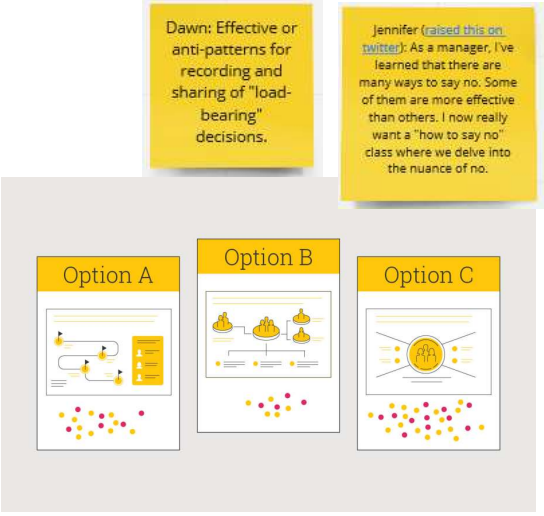


Image: https://x.xplane.com/dot-vote-method-card-download

TECHNICAL DECISIONS

Decisions Are About Choices

Sometimes we'll do this work, and the choice is clear. Or we pick the most intuitive path based on our experience and best sense of what to do with limited time, and attention and resources. Other times, it's hard — the choice may not be clear, or those involved may be split on what alternative is preferred. It is all the more difficult (and potentially fractious), if it's a crucial decision that much depends on.

Some ways to winnow the choice set: dot voting and attempts to converge; play "vote off the island," voting the least preferred out of the consideration set (and then considering factors like minimization of regret). A powerful approach, is to go back to the notion of reversibility, and consider how to make the decision testable in a short timeframe, before too much depends on it and while it is still reversible.

For those who disagree, we can ask: what would it take, to make this approach work for you? Or: if you partially agree, is this sufficient to try out, and what should we explore first to address your concerns?

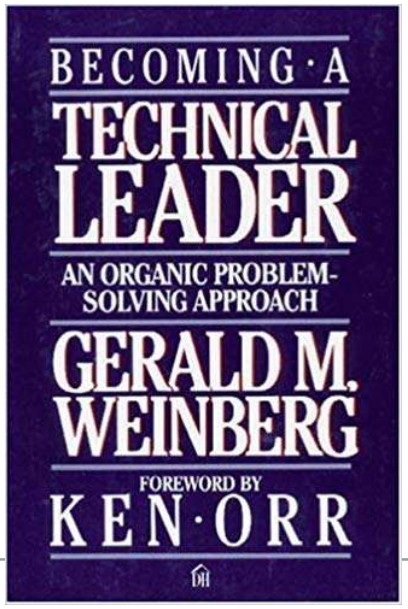
Hard Choices

When we're working across the system, strategically and structurally significant decisions need to be made from a system — not local (to a part) — perspective. So not only do they impact different people — stakeholders (the ones with stakes, to quote Tom Graves) — but they are things people care about, and have strong, but different, opinions about. They are seen from different vantage points where there are different pressure points, by different teams and their people, responsible for different pieces of the system. We work across the system. And across the turfs and charters of teams, and individuals, and functions. Well. All this means that people will see things differently. Care about them differently. And as leaders we need to make decisions to meet broader system or organizational goals. Decisions that will sometimes look suboptimal from the perspective of local goals. These decisions need to be communicated effectively, so that progress can be made even if there isn't uniform agreement.

Common Understanding

"Unless and until all members of a team have a common understanding of the problem, attempts to solve the problem are just so much wasted energy."

— Gerald M. Weinberg



Some Ways to Develop Common Understanding

We orient to working together, especially in ways that draw out assumptions and ideas, so that we can ask questions and probe (work them out, as well as instrument, to better assess) and respond to them together. Working together (ensemble design as well as ensemble programming) is a rich way to build common ground and shared understanding. But when teams (of teams) get too large for this to be particularly effective, we start to rely on a more fractal approach, with smaller teams. As much as we can, we involve team members to work collaboratively on addressing concerns that cut across teams. Still, proactive system identity and integrity defining work, needs perspective and leadership across boundaries. There's also the matter of building organizational will to do bigger things that impact various teams. At any rate, even where everyone can't be involved, drawing in some of those who'll be impacted, helps to bring ideas and concerns into the decision making, and builds understanding among those who can share it in their teams. In other words, working organically, through participation, broadens the set of those who can tell the story of the system, its architecture and key decisions. We still need to write significant decisions down (and describe architectural models and their implications) and talk about them.

"[The basic compact of joint activity] includes an expectation that the parties will repair faulty knowledge, beliefs, and assumptions when these are detected. Common ground is not a state of having the same knowledge, data, and goals. Rather, common ground refers to a process of communicating, testing, updating, tailoring, and repairing mutual understandings"

— Klein et al, Common Ground and Coordination in Joint Activity

If you are a good leader,
Who talks little,
They will say,
When your work is done,
And your aim fulfilled,
"We did it ourselves."

— Lao Tse

"He who establishes his argument by noise and command shows that his reason is weak."

– Michel de Montaigne

Communicate, Communicate

"The longer I'm a leader, the more I realize that communicating something once is the equivalent of not communicating it at all. Communicate the [th/b]ring repeatedly until they literally ask you to stop."

— Nivia Henry



Image source: Nivia Henry (@lanooba) with permission

TECHNICAL DECISIONS

Go Ahead, Repeat Yourself

I really have to remind myself that I not only get to repeat myself, but I MUST repeat myself -- for the benefit of others. Contrast this (apocryphal??) interchange:

"Simplify, simplify" — H. D. Thoreau

'One "simplify" would have sufficed' — Ralph Waldo Emerson

with Eberhardt Rechtin's:

"Simplify, simplify, Simplify"

And recall he also said:

"Communicate, Communicate, Communicate"

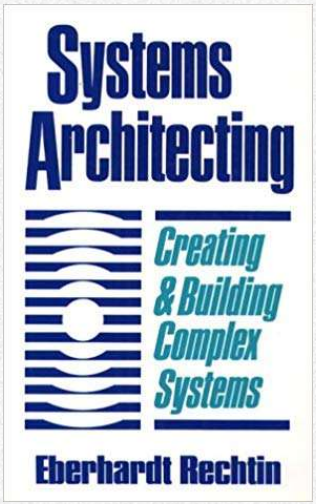
So much competes for attention, we miss things. And there are things we don't understand at first, and need to hear again, perhaps another way. Communicating helps increase awareness – of the decision and its ramifications, and implications we may not have been aware of. Conversations move understanding around. We need to keep having them, and drawing attention to what is important, or subtle or overlooked. We're helping to build shared understanding of critical shaping decisions, whether its architecture, product direction, strategy, or other matters of importance to system integrity and business outcomes.

"bring" was a typo (for thing), but I liked it — communicating what we're doing, why it matters and how that's important for the system (us/stakeholders/..).

In addition to our words, there's what is communicated by our actions.

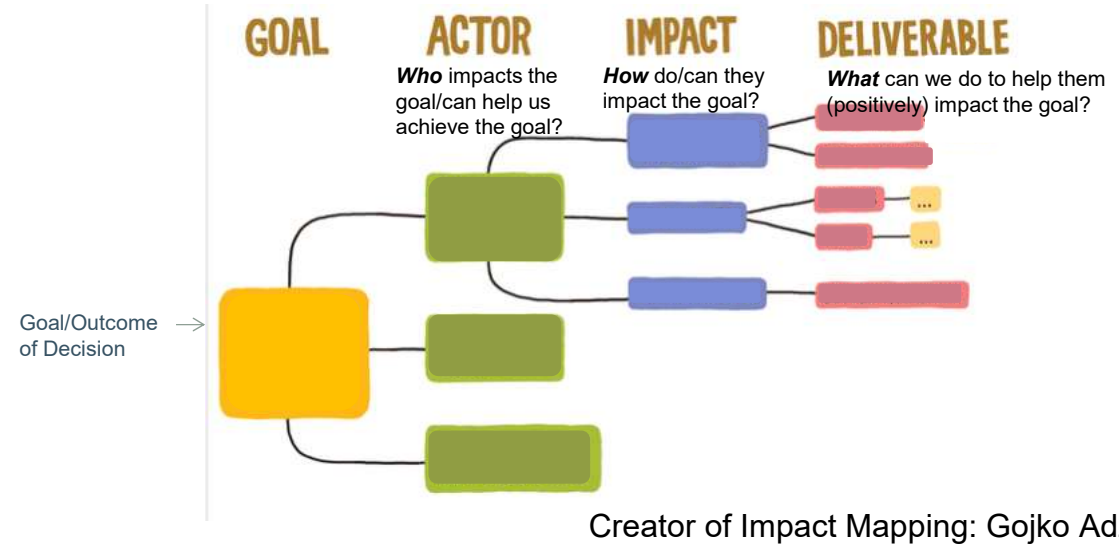
"Don't ever stop talking about the system"

— Eb Rechtin



Explore the decision in terms of who needs to be enabled and how

Impact Mapping and Enablement



Creator of Impact Mapping: Gojko Adzic
Image adapted from: https://www.impactmapping.org/example.html

Impact Mapping

We can use impact mapping to explore how to make the decision successful, organizationally. What information do teams need, in what format? What capabilities do they need to build and how can we help?

Impact Mapping Template

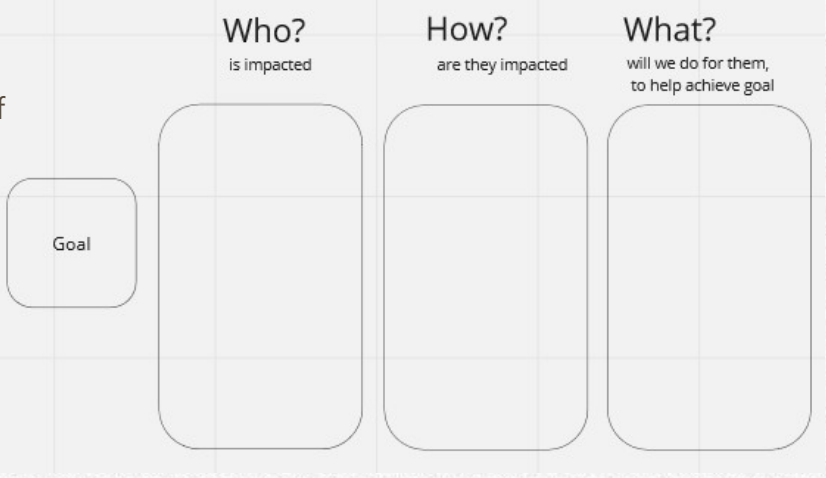
"*Goal*: The centre of an impact map answers the most important question: Why are we doing this? This is the goal we are trying to achieve.

Actors : The first branch of an impact map provides answers to the following questions: Who can produce the desired effect? Who can obstruct it? Who are the consumers or users of [our decision]? Who will be impacted by it? These are the actors who can influence the outcome.

Impacts: Next, we answer the following questions: How would our actors' behaviour need to change? How can they help us to achieve the goal? How can they obstruct or prevent us from succeeding? These are the impacts that we're trying to create.

Deliverables: Once we have the first three questions answered, we can talk about scope. The third branch level of an impact map answers the following question: What can we do, to support the required impacts?"

Source:
https://www.impactmapping.org/drawing.html

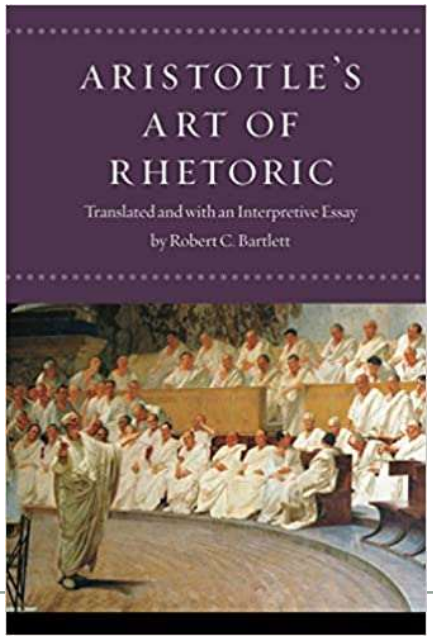


Persuasion

Rhetoric: persuasion through the use of oral or written language (term coined by Plato)

In *The Art of Rhetoric*, Aristotle defined three modes of rhetoric

- Logos: an argument from reason
- Pathos: emotional appeal, and
- Ethos: reputation and credibility



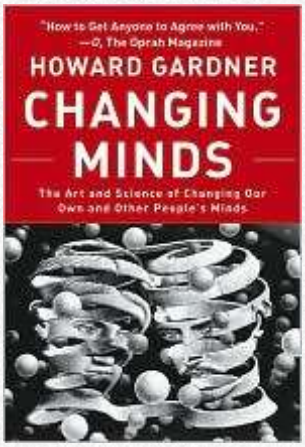
Logos, Ethos and Pathos

"There are 3 frogs on a log. One decides to jump. How many frogs are on the log?" (Cory Booker) Just deciding, doesn't mean anything gets done. As much as we eschew manipulation against anyone's interests or will, there's still the work of influence or persuasion and building sufficient common ground or understanding, so that others can and will act.

In engineering, we tend to use and appeal to logic as our principal tool for persuasion; logos, standing on the shoulders of ethos. We might try to avoid pathos, the appeal to the emotions, but then we neglect enthusiasm, and the importance of building it in ourselves that it may light in others; we neglect to tell stories that connect personally, appealing to the common history and aspirations of those we would persuade. We need to be aware of *epistemic injustice* (Miranda Fricker), in particular testimonial injustice, which is unfairness related to trusting someone's word.

In *Changing Minds*, Howard Gardner plays with the pun – changing minds not just in the sense of persuading, but in the sense of changing the information we have to work with or what we hold in mind, too. For example, presenting *reason*ed arguments where relevant factors are presented and weighed (the decision template does this); collecting and presenting relevant data and *research*; *resonance* which depends on style of audience and may be appeal to logic, or may be through stories or narratives; representing the information differently (think back to the monk and the mountain exercise and how we "simulated" the monk walking the path both days, and the visual representation) (what Gardner calls *representational redescription*); etc.

"People don't change because of data. They change because of what they value."
— Esther Derby



Performance Art

"When I taught a "user interface" class at the University of Virginia, I'd bring in a working VCR on the first day. I would put it on the desk in front of the room. I would pull out a sledgehammer. I would destroy the VCR. ... I sure got their attention. That's always the first step to solving an ignored problem."

— Randy Pausch



Persuasion and Performance

Culture is expressed and shaped; built, repaired and renewed; acted and reenacted. Some is in the doing, in the performance of work and conversations and what we encourage and discourage. Still, we might note the role of the performance of performance, whether it is in a setting like a talk or a meeting, or in ensemble programming, hallway conversation, etc. It might be to draw attention, as Randy Pausch did. As Steve Jobs reportedly did (e.g. tossing iPod prototype in aquarium: "Those are air bubbles," he snapped. "That means there's space in there.") — in a way that arguably eroded other important social elements, like psych safety, at least for some, or inclusion and diversity, at least for some…. Now, what both Pausch and Jobs were drawing attention to were, often, elements of quality or design excellence. That's the lesson, not any unkindness!

Meanwhile at Amazon…

> "All of our meetings are structured around six-page memos" … "When you have to write your ideas out in complete sentences and complete paragraphs, it forces a deeper clarity of thinking." — Jeff Bezos

"When I'm getting ready to reason with a man I spend one third of my time thinking about myself and what I am going to say — and two thirds thinking about him and what he is going to say."
~ "Abraham Lincoln" (apocryphal?)



Persuasion and Influence

Cialdini's 6 fundamental principles of persuasion:

Liking: People like those who are like them.

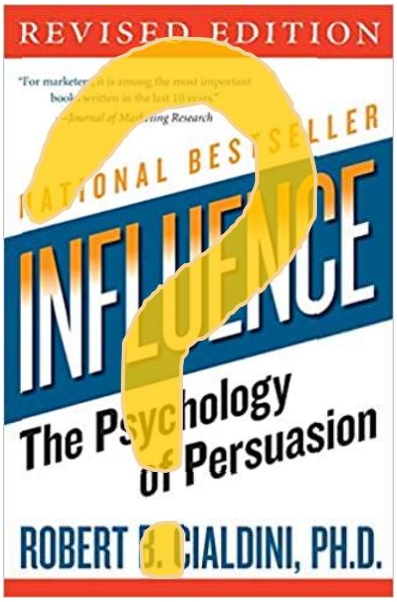
Reciprocity: People repay in kind.

Social Proof: People follow the lead of similar others.

Consistency: People align with their clear commitments.

Authority: People defer to experts.

Scarcity: People want more of what they can have less of.



Participation Persuades

Cialdini's principles of persuasion may be self-illustrating? In which case, how do we break out of the trap of self-fulfilling like-mindedness, and foster more diverse inclusive teams/organizations?

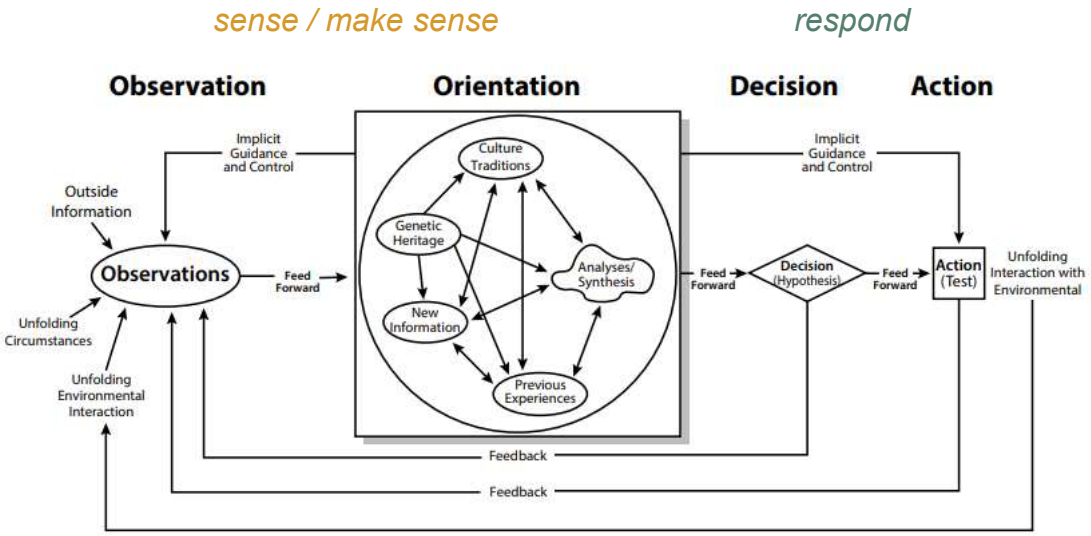
One thing to bear in mind, is that participation helps build understanding and common ground. By being there (for the discussions that surface assumptions and shape desired outcomes and surface and weigh tradeoffs, and make the judgment calls about simplifications, sometimes, and interweaving understanding across views and perspectives to take in more of the complexity at others, etc., etc.), we build up our understanding (mental models) and, frankly, feelings! We become invested – we "feel and take ownership." But we can't do everything "by committee" together in large, and then even larger, groups, as the system and organization grows and adapts to challenges. So we look to extend participation through "design crits" or "design labs" or reviews. By creating opportunities for areas of participation, understanding and contribution. And by writing about the decisions. But this may be more helpful for some stakeholders and not so much for others. Senior management, for example. Or very large teams. So we want to think about and work on bringing them along to, so they understand how they can support and enable, and what we are doing to support and enable them.

"Persuasion used to be a "good" word. It was the opposite of coercion. It was wrong to try to force people, you should try to persuade them. Now we tend to think that persuasion is, or rather may easily be, a kind of force."

*— Mary Parker Follett**

* Source: Leaders and Experts, Mary Parker Follett, 1927, in Dynamic Administration

Decisions Interact With Context



John Boyd, "A Discourse on Winning and Losing"

Decisions Orient; Decisions Interact with Context; It's Loopy!

The OODA loop is a simplification, for all its depiction of feedback loops. While we're making decisions, we need information and re-enter observe/orient or sense/make sense stances and activities. Our evolving understanding changes our perception of the context. Our decisions change actions, potentially already underway. Actions change the context. And so forth.

It's still hard, so practice self-empathy as well as self-awareness (being aware of our impact on others)



John Cutler
@johncutlefish

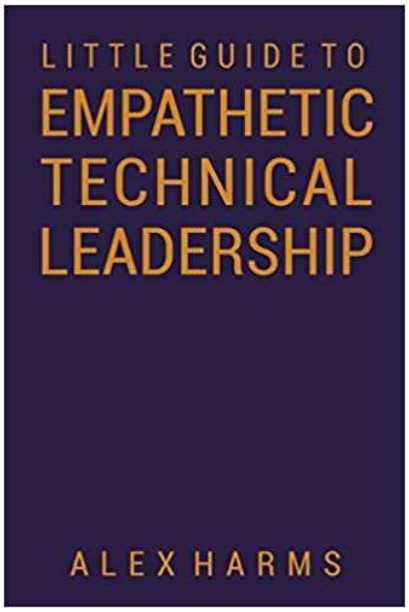
Some things I've learned over the years as a Systems (Over)Thinker

1. Take care of yourself. Your brain is working overtime—all the time. Practice "radical" recovery
2. You may spend a lot longer thinking about things than *most* people. Pace your delivery
3. If you go deep first, and then simplify...keep in mind that you don't need to show all of your work
4. Your default description of (almost) any problem will be too threatening/overwhelming
5. Do your deepest thinking with co-conspirators (not the people you're trying to influence)
6. Informal influence is often not formally recognized. Prepare mentally for this
7. The people you're trying to influence spend 98% of their day overwhelmed by business as usual
8. Remember to *also* do the job you were hired to do (if you don't you'll be easier to discount)
9. Seek "quick wins", but know that most meaningful things will take a while
10. Some things take ages to materialize. It is discontinuous, not continuous
11. Make sure to celebrate your wins. They will be few and far between, so savor the moment
12. The people who support you in private may not be able to support you in public. Accept that
13. Hack existing power structures—it's much easier than trying to change them
14. Consider becoming a formal leader. It's harder in many ways, but you'll have more leverage. What's stopping you?
15. In lieu of being a formal leader, make sure to partner with people who actually "own" the area of change
16. Watch out for imposing your worldview on people. Have you asked about what people care about?
17. You'll need a support network. And not just a venting network. Real support
18. "Know when to fold 'em". Listen to Kenny Rogers *The Gambler.* Leave on your own terms
19. Don't confuse being able to sense/see system dynamics, with being about to "control" them. You can't
20. Grapple with your demons, and make sure not to wrap up too much of your identity in change

Source: John Cutler, https://twitter.com/johncutlefish/status/1518361842342453248

Practice Self-Care

Leadership is hard. We try to accomplish bigger things than one person can alone, but we all have better ideas… and so does reality… We need to practice self-care (too; in balance).



Leading is Hard; It's Work

Leadership is hard. And even if we, as an organization and as individuals, embrace failures as key to learning and being responsive in changing, even volatile, contexts and situations, it takes a toll. We strive to get things more right, more the way they ought to be, and it's a load; there are cognitive, relational and emotional pressures, forces, currents, alignments and misalignments, ⋯ and it's a lot. A lot!

I've made points along these lines already, but it bears repeating:

> "Does your role involve being an "integrator" across teams?
>
> Supporting. Connecting. Cohering. Resolving. Balancing. Listening. Translating. Absorbing.
>
> If so, there is a good chance you have a view of your organization that no one has (not even the CEO).
> — John Cutler

This is important too:

> "You are taking on an emotional burden that most people don't take on. And you are shouldering an emotional burden that often goes unrecognized and unappreciated...and unpaid."
> — John Cutler

Be kind. Also to yourself.

Source: John Cutler, https://cutlefish.substack.com/p/tmb-4252-the-integrator-burden

Attribution

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

For more:

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



TECHNICAL
LEADERSHIP

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:

On LinkedIn

Mastodon:
@RuthMalan@mastodon.social

Web: ruthmalan.com

Workshops

- **Technical Leadership,** Nov 3 and 10, 2026 at 12pm – 3pm Eastern Time
- **System Design and Software Architecture,** Oct 5-7 and Oct 12-14, 2026 at 11 am - 3:30 pm Eastern Time (US/Canada)
- *More: https://ti.to/bredemeyer/*

"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 — 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