TECHNICAL LEADERSHIP MASTERCLASS

by:
Ruth Malan and Dana Bredemeyer
Bredemeyer Consulting

Helping teams succeed

September 13, 2019
INTRODUCTION

Leadership Essentials
Technical Leadership
Leadership Excellence

Note: This is a preview of selected material only.

This is a preview selection of material from the Bredemeyer Consulting Masterclass in Technical Leadership.

For more information, see: https://www.ruthmalan.com

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

complex (adj.): from com "with, together" (see com-) + plectere "to weave, braid, twine, entwine," from PIE *plek-to-, suffixed form of root *plek- "to plait."

To enroll in the next open enrollment Technical Leadership Masterclass, see: https://www.ruthmalan.com/Bredemeyer/Technical_Leadership_Masterclass_Overview.htm
## Decisions

<table>
<thead>
<tr>
<th>Leadership and decision making</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decisions constrain and enable</td>
<td>How do we make decisions?</td>
</tr>
<tr>
<td>When do we make them?</td>
<td>Significance</td>
</tr>
<tr>
<td>Forces and Tradeoffs</td>
<td>Communication</td>
</tr>
<tr>
<td>Mental Models</td>
<td></td>
</tr>
</tbody>
</table>

Bredemeyer Consulting
Thus, the demands on expertise and experience, increases 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.

So leaders make decisions that set context for further decisions. And decisions that cross contexts or boundaries, need leadership - - to help others understand the need and outcomes, and consequences, and what their role is in making the decision effective.

We need to make decisions.
Decisions Constrain

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

— Alicia Juarrero

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

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.

"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
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).

"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

Recommended video: Constraints that Enable Innovation - Alicia Juarrero

https://vimeo.com/128934608
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 Constraint”

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”

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

— Alicia Juarrero

We need to make decisions. But 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.”

And Jeff Atwood:
“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."

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

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

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 (whether 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), are going to cleave the design space, ruling some opportunities out.

“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.
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

No “one-size fits all” decision making

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

We need to make those decisions deliberately, attentively

Attending to Irreversible, Consequential Decisions

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]:

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

https://twitter.com/farnamstreet/status/1026105498372845571
Reversible Decisions

“But most decisions aren’t like that – they are changeable, reversible – they’re two-way doors.”

— 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 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.

- **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
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

Decisions Entail Tradeoffs and Tradeoffs

Don’t Stay Their Lane \(\_\(\_\)\_\)

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.

“[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”](https://www.pragprog.com/articles/the-art-of-tradeoffs)
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

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.

“Good engineering is less about finding the “perfect” solution and more about understanding the tradeoffs and being able to explain them.”
— Jaana B. Dogan
“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?**
Trade-offs: Dyads

Control     Autonomy
Global perspective     Local responsiveness

Control     Co-operation
More consistency     More flexibility

Co-operation     Autonomy
More synergy     More accountability

“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).
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

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
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

“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
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

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
**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.

‘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 (“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*
Forces in Dynamic Tension

Rasmussen’s dynamic safety model describes the feasible operating space for a sociotechnical system. Here, we’ve adapted the model to highlight the tension given economic and market forces, which puts pressure on code properties.

Dynamic Safety Model

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.

In the Cook and Rasmussen model (2005), a business operates within 3 boundaries (economic, workload and acceptable performance); failure occurs when the operating point moves outside this envelope. The economic boundary is formed by the zone outside which the organization will fail—make a loss, run out of investor willingness to fund the venture, etc. The workload boundary delineates where the workload becomes too much, the pressures on workers too high, etc. Because a business doesn’t want to fail, it pushes to lower costs and this puts pressure on the operating point, pushing it closer to the acceptable performance boundary (outside which the system fails). Exactly where this boundary is, is not known precisely—it is only discovered when an accident or failure occurs. Therefore, the organization needs a safety buffer or margin, so that it doesn’t cross the failure boundary. The dynamic nature is emphasized, as workers constantly strive to keep the system operating without serious failure. In the operations case, the performance boundary has, for example, to do with scalability and resilience—failures that cause loss and/or distress for customers and news vans to line up outside corporate headquarters.

This model can be used to inform how we think about habitable zones and technical debt—in this case, we’re dealing with developer workload and code habitability. Within the boundary, the code is sufficiently changeable or adaptable; the boundary or failure threshold has to do with coupling and other forms of debt, that thwart adding features and raise cost of change. And we can relate the “performance boundary” to the edge of habitable zone for users and operators, beyond which, for example, security breaches and system outages cause failure. Code quality erosion can feed forward into economic failures (impacting reputation/trust and revenue), impact software properties and lead to workload (intolerable stress and cognitive load) failures.

"all systems are what emerges over its history of adaptation to stressors"
— David Woods
Context Matters

“Design quality is not a property of the code. It's a joint property of the code and the context in which it exists.”

— Sarah Mei

"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

Context Factors

“[system design] strives for fit, balance and compromise among the tensions of [stakeholder] needs and resources, technology, and multiple stakeholder interests” (Rechtin and Maier) There is no perfect solution. Eb Rechtin put it this way: “The essence of architecture is structuring, simplification, compromise, and balance.”

We joke about the two word answer to any question, that distinguishes the architect: “It depends.” But a good architect tells you what it depends on.

At a recent conference, Diana Montalion shared her definition of wisdom:

Wisdom = knowledge + experience + good judgment

According to this definition, wisdom is the ability to know what “it depends” on.
“Always design a thing by considering it in its next larger context.”
— Eliel Saarinen

**Always Consider**

“101 Things” is a book written for building architects, but has translatable lessons for software architects. In it, Eliel Saarinen* is quoted: "Always design a thing by considering it in its next larger context — a chair in a room, a room in a house, a house in an environment, environment in a city plan." We could amend Saarinen’s point to “always decide a thing by considering its context.” Decisions, any decisions, must take context into account too. From desired outcome(s) to forces that impinge, to side-effects, interactions and consequences, context factors. [*Also, related by Eero Saarinen in Time Magazine, "The Maturing Modern," 7/2/56, pp-50-57]

Christiane Floyd pointed out: “Design consists of a web of design decisions which, taken together, make up a proposed solution.”

This is true of any design – including organizational design, or UI design, and system design. As well as the design or formulation of initiatives and strategies.

Back to Christiane Floyd: ‘By design I understand the creative process in the course of which the problem as a whole is grasped, and an appropriate solution worked out and fitted into human contexts of meaning. In [Peter] Naur’s words: "Software development is an activity of overall design with an experimental attitude".’

“Software Development as Reality Construction” (by Christiane Floyd, 1992) is an exciting work, for it articulates software development as a co-evolving, dialogic process where we are learning what the design needs to be, even as we adapt both the system and its context. That is, it exhibits what Nora Bateson termed “symmathesy” (learning together).
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.

"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.
Formulating the Problem

Dancing with Systems

1. Get the beat
Before you disturb the system in any way, watch how it behaves.

2. Listen to the wisdom of the system
Aid and encourage the forces and structures that help the system run itself.

— Donella Meadows

Observe the System

“Before you disturb the system in any way, watch how it behaves. […] If it’s a social system, watch it work. Learn its history. Ask people who’ve been around a long time to tell you what has happened. If possible, find or make a time graph of actual data from the system. Peoples’ memories are not always reliable when it comes to timing.

Starting with the behavior of the system forces you to focus on facts, not theories. It keeps you from falling too quickly into your own beliefs or misconceptions, or those of others. It’s amazing how many misconceptions there can be. […]

Listen to the wisdom of the system. […] Don’t be an unthinking intervener and destroy the system’s own self-maintenance capacities. Before you charge in to make things better, pay attention to the value of what’s already there.” — Donella Meadows

“the problem itself is grasped in the course of the design process.”
— Christiane Floyd
Mental Models

‘these facets are incorporated into an internal representation that is sometimes called a "mental model"’

— SNAFUcatchers

Mental Models

We all have mental models. They are all imperfect. And they are all different. But they are the basis for our decisions — consequential decisions; decisions we may not be able to back out of, especially as other decisions become layered upon, and even entangled with, them.

“Our decision making relies on mental models

“what [@ri_cook] says is that there are multiple mental models inside an organization. And that is advantageous! It’s good that there are multiple models. And the reason it’s good, is because of the bounded rationality of each of these actors — no one person can hold the whole system in their head. The frames they make in their mind of what exists below the line, are what enable them to understand and make decisions about what’s happening. If they tried to put the whole thing in their head, they’d spend their whole time trying to load it in their head, and they’d never make any decisions.” – Jabe Bloom, Two Frames on Development and Operations, DevOpsCon

Woods’ Theorem: “As the complexity of a system increases, the accuracy of any single agent’s own model of that system decreases rapidly.”

— Christiane Floyd

Image by Dave Gray in “Liminal thinking The pyramid of belief”
Making decisions, as well as conveying decisions, is as much about what is relevant in the context as it is about the decision we make in response. If we want to make better decisions, and convey them well, we need to have a (good enough) conception of the desired outcome(s), the forces we need to weigh and constraints we need to take into account, as well as the significant consequences and side-effects of our decision. Where these goals and forces arise in various contexts — development and operations, the contexts of use, and the broader value network. We’re concerned with factors impacting developer experience, and forces arising from development constraints, capabilities, organizational forces like coordination mechanisms and costs (differing for collocated and distributed teams, say) and taking into account trends in the technology ecosystem. And so forth, also for operations engineers, management teams, security teams, as well as customers and users in various segments and at different points on the user path, as well as partners in the value network — channel partners, others adding value to our products, and more.

"What, at this extraordinary moment, is the most important thing for me to be thinking about?" — Buckminster Fuller

Of course, this is a lot. These contexts are all important, and to avoid being overwhelmed, we have to use judgment in determining what to pay attention to. What do we need to be paying attention to, now? And what do we need to table, but be mindful about returning to, later? And what is tantalizingly almost relevant, that is hovering in peripheral vision, that we might, at some point need to turn our attention to?
Decisions Across Boundaries

Decisions that impact across boundaries, need perspective across boundaries

System Design is Contextual Design

Recognizing that a system changes its contexts, means recognizing we’re designing the system-in-context — not just the system, but the context too. (While we have limited degrees of design freedom with respect to the context, everything the system takes on, impacts its (various) context(s), so we are redesigning at least some aspects of the containing socio-technical systems and broader context.)

Alternately put, to develop our “theory of the problem,” or to “load” the context into our mental models, so that we can uncover this multidimensional decision options and tradeoff space, we need to ask (not just) “what do users need?” but also “what do developers and testing need?” and “what do our operations and security teams need?” and “what do others in the value network need?”

“We need to ask: what does the code need?” — Michael Feathers

We architect across — across boundaries: across not just the code and the teams involved, but across the internal system design (architecture and code/tests) and design of the system-in-use or system-of-systems design (what our industry has tended to call “requirements”); across the different languages and concerns of these different spaces, the technical language of code and test and integration, deployment and operation, and the languages of the domains where the system is used; across the turfs and sense of ownership and decision responsibility; across views and perspectives; etc.

But of course, we can’t attend to everything, at least not all at the same time, in detail. We “zoom out,” as it were, to scan the ecosystem or value landscape, to identify opportunities and challenges that do warrant closer attention. To set framing for the problem, to understand the trends and forces that shape and constrain it. To get a bearing on the ecosystems that are or will be impacted.
Recap: Our systems, and organizations, are complex, or grow to be (Lehman’s Laws). Organization design, like other system design, entails a set of tradeoffs to weigh and balances to strike. Organizations have sub-entities because we organize to focus, to build and leverage capabilities, to get work done.

Communication costs – in terms of time but also in terms of focus of attention. Diverse perspectives are important to innovation; too many perspectives diffuses attention, increasing cognitive burden and demands on relationship fostering. Interdependencies cost in terms of potential for delays as well as interactions and relationships which need to be established and maintained.

So we seek to identify responsibility boundaries so teams can be more independent. And yet we want to create systems with structural and design integrity – that is, conceptual integrity, as well as robustness where it matters, and resilience or adaptive capacity. And organizational integrity (matters of ethics, and social and environmental responsibility).

Decision making in strategy and architecture is about setting direction and context, so that decision making and work at more narrow scope, produces something coherent at broader (system or system-of-systems) scope. Without these decisions, we have piecemeal contributions which fail to add up to a system with integrity. These decisions have impact across boundaries (and their associated arenas of responsibility), and it takes organizational will (determination, because they are hard and other things compete for attention), and a commitment to understanding the decision and its ramifications, to follow through.

Participation in decision making helps build understanding and a sense of priorities, but broad participation in every decision doesn’t scale. So “higher level” decisions (decisions that impact across boundaries) need to be attended to and made in a smaller decision setting (a few people), but advocated for and shared in a way that brings others along, so that impacted work is consonant with these decisions. That is, decisions that impact others’ work across boundaries, entail leadership across boundaries.

*As quoted in Rechtin and Maier

**Decisions Across Boundaries**

“It is inadequate to architect up to the boundaries or interfaces of a system; one must architect across them.”

– Robert Spinrad*

---

**Systems: Cohesion, Integrity and Leadership**

Wait. Why are we talking about decisions?
I can’t point to why Gary Klein calls his book *Sources of Power.* (He studied decision makers in high uncertainty, stressful situations.) But this is the sense I make: Making good decisions in high risk, high outcome situations, is a source of power. If people look to you to make critical outcome-shaping decisions, in a socio-dynamic sense, that’s leadership. It’s not all of leadership. But it factors.

In many situations, it’s associated with being in roles associated with leadership. Yes, there’s the Peter Principle (a person will rise in the hierarchy, stopping when they reach the level at which they are incompetent), but that’s cynical. Working at broader scope, across domains, changes at each scope change (or level, in a hierarchy). The demands of the role differ, the context shifts, the relationships are different. Expertise needed shifts. Some people don’t make the transition, others need mentors or coaches and time to learn. Just like anything else. But largely, it’s about relationships, and expectations, and trust or confidence in decisions.

Now, an inversion of the structures and dynamics of power comes of drawing on others to share in the decision making. “This typically requires an intersection of the right technical knowledge, a thorough understanding of your organization’s goals, authority to make the decision, and responsibility for the consequences of the decisions made.” — David Marquet (*Turn the Ship Around*)
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 storytelling”

— 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.”

“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
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

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

* in his Nobel Prize in Economics speech
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

**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, HBR, 2007) asks us to imagine, during design, say, that a project has gone wrong, and to explore why.

“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 8 A.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?

Source: Visual Thinking by Rudolf Arnheim

Sometimes Don’t Means Don’t — or Else!

Or else you run a risk, like spoiling the surprise, or lesson 😊
Thinking with a pencil, by example

One Approach

(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.
Thinking with a Pencil

"But thinking is nothing but talking to yourself inside."

"Oh yeah?" Bernie said. "Do you know the crazy shape of the crankshaft in a car?"

"Yeah, what of it?"

"Good. Now tell me: how did you describe it when you were talking to yourself?"

So I learned from Bernie that thoughts can be visual as well as verbal."

— Richard Feynman

"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
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
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

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).
The Leadership Moment

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

– Buckminster Fuller

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?”

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!
**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.

**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

“A Quick Introduction to Cynefin ,” by Liz Keogh
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.

**Compensate for blindspots, taking a different point of view**

** Repairing Blindspots

"A change of perspective is worth 80 IQ points" (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.

* referencing the Simons and Chabris Selective Attention Experiment.

—— Dawn Ahukanna

---

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

“If you haven’t thought of three possibilities, you haven’t thought enough.”

— Jerry Weinberg

---

**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. Use rich pictures, use case and component diagrams and play over behavior of interest — repeat. Use other views. Do this at system level early to clarify direction worth taking/starting out in. But continue to do this (with just enough sketches and modeling focused on the concerns at hand) as challenges present themselves; some of these arise in the use context; some are internal “life sustaining” mechanisms that the system needs for it to be/become the kind of system it is (meet its persistence and dynamic data delivery/consistency needs; meet its scaling demands for spikes and growth; etc.).

At any rate; “plan to throw some away,” needs to include sketchprototypes. We need to 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 at the granularity we learn when we learn in the medium of code, but we at least start to try ideas out, and explore and bat at them, investigate how they could work, in sketch-driven-dialog.

Three possibilities? 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.

---

**Characteristics of architects:**

“High tolerance for ambiguity”

“The willingness to backtrack, to seek multiple solutions”

— Eb Rechtin
And

“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 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 interface design and to address 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 architecture and key decisions. We still need to write significant decisions down (and describe architectural models and their implications) and talk about them.

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
Communicating the Decision

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

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 memory that others can access, understand, and help improve.

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 (yet well-described) Architecture Decision Record template caught on as a just enough version for documenting architecture decisions in an Agile context.

The Architecture Decision Record documents decisions in terms of the statement of the decision, the outcome sought and the forces weighed in the making of the decision, along with consequences or implications of the decision.

The Tyree/Akerman and Zimmerman versions also keep track of alternatives considered but ruled out, and this is valuable too.

See Nat Pryce’s ADR tools on Github:
https://github.com/npryce/adr-tools
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 bring repeatedly until they literally ask you to stop.”

— Nivia Henry

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.

A leader is a storyteller, setting context, connecting-dots to higher level needs and decisions, and coaching on how we reached decisions, and why.

“Don’t ever stop talking about the system”

— Eb Rechtin
Quoted in this module

Dawn Ahukanna: @dawnahukanna Jaana Dogan: @rakyll
Jeff Atwood: @codinghorror Sarah Mei: @sarahmei
Abeba Birhane: @abebab Diana Montalion: @dianamontalion
Kent Beck: @KentBeck Shane Parrish: @farnamstreet
Grady Booch: @Grady_Booch Mary Poppendieck: @mpoppendieck
Melvin Conway: @conways_law Robert Smallshire: @robsmallshire
Richard Cook: @ri_cook Rebecca Wirfs-Brock: @rebeccawb

“There is a Zulu phrase, ‘Umuntu ngumuntu ngabantu’, which means ‘A person is a person through other persons.’
— Abeba Birhane

References

Bateson, Mary Catherine, “How To Be a Systems Thinker,” Edge Conversations, 4/17/18

Birhane, Abeba, “Descartes was wrong: ‘a person is a person through other persons’,” aeon, April 2017

Cook, Richard and Jens Rasmussen. “Going solid”: a model of system dynamics and consequences for patient safety. Qual Saf Health Care, 2005


Juarrero, Alicia, Dynamics in Action: Intentional Behavior as a Complex System, 1999

Kahneman, Daniel, Thinking, Fast and Slow, 2013

Keidel, Robert, Seeing Organizational Patterns, 1997

Keogh, Liz, A Quick Introduction to Cynefin, in “Cynefin for Everyone,” on www.lizkeogh.com
Quoted in this module

Sue Borchardt: @contemplatethis  Phillip Johnston: @mbeddedartistry
Michael Feathers: @mfeathers  Liz Keogh: @lunivore
Nivia Henry: @lanooba  David Snowden: @snowded

References


Kanter, Elizabeth M., “Power Failure in Management Circuits.” HBR, July 1979

Lewin, Kurt, *Field theory in social science*, 1951


Woods David, STELLA: Report from the SNAFUcatchers Workshop on Coping With Complexity, Ohio State University, 2017

Young, Indi, *Mental Models*, 2008

“We know from everyday experience that a person is partly forged in the crucible of community.”
—Abeba Birhane

Shoulders we stand on
Attribution

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

For more:

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

Duarte Slidedocs®

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

Quotes and Photos

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

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

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

“Act always so as to increase the number of choices.’
— Heinz von Foerster
“The most essential (and satisfying) work of the leader is to create more leaders.”
— Mary Parker Follett