
From the sky we look so organized and brave
Flying Over Water
Jason Isbell

The Wisdom of Nature

The Hubris of the Engineer

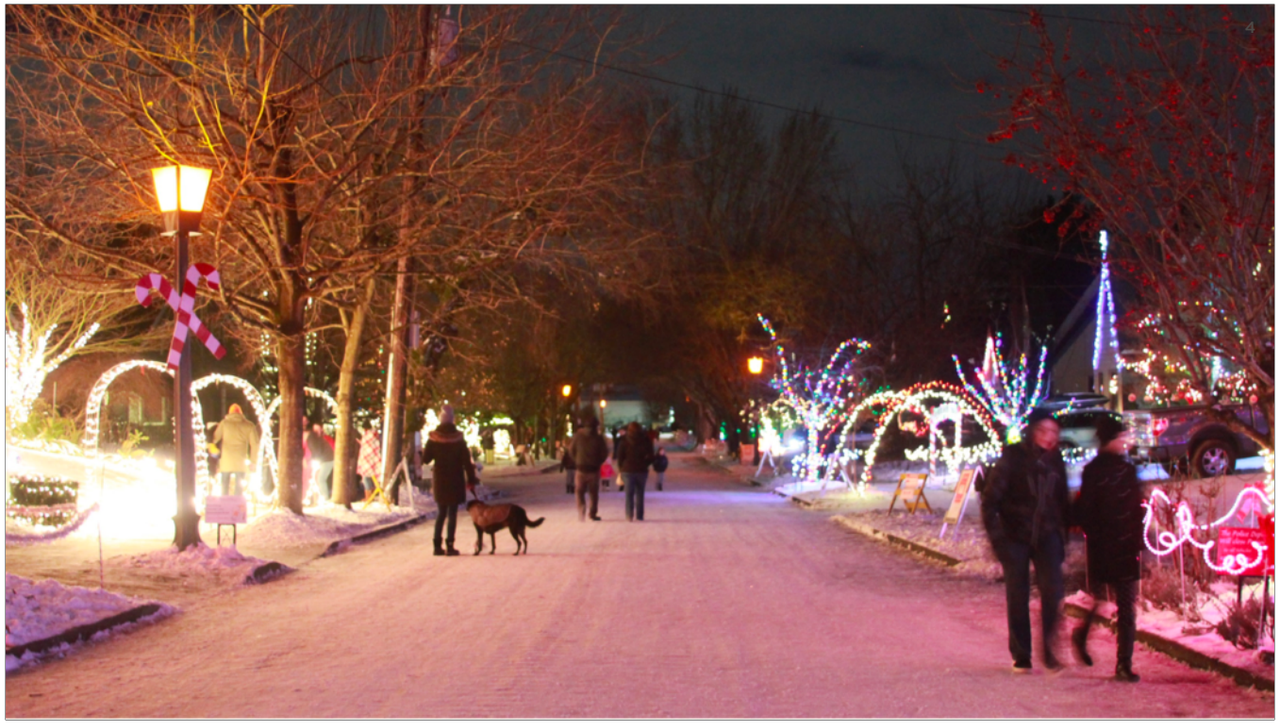
0111 Conference 2020
Aaron Longwell
https://adl.io

Thank you, Etienne. One of the things I love about Etienne is how brave he is.
Brave in this case to gift me an hour of your time to talk about things of which I am no expert.
The central idea in this talk comes from evolutionary biology. I majored in Public Policy and Media... I'm not sure I even took a class in the natural sciences in college.
Many of you might know much more than me about these topics. I welcome the opportunity to learn from you, so please take time in the QA to correct me as you see fit.

This talk starts with a story. Let me set the stage. This is 8 years ago. My CEO is at headquarters in Denver.
The company collects real estate data, and has a proprietary algorithm for valuations.

If we founded it today, we'd have called it a "Data Science" company, but that word wasn't in common usage at the time.

Our team was impressive. We'd recruited our senior engineers away from much larger companies, including Google. The head of our analytics department was a West Point educated applied mathematician who had previously led risk assessment for the Joint Chiefs of Staff at the Pentagon. The vast majority of the development team went on to CTO, VP or Principal Engineer roles. 2 of them are now founder/CTOs of their own startups. By every measure this was a top notch team.

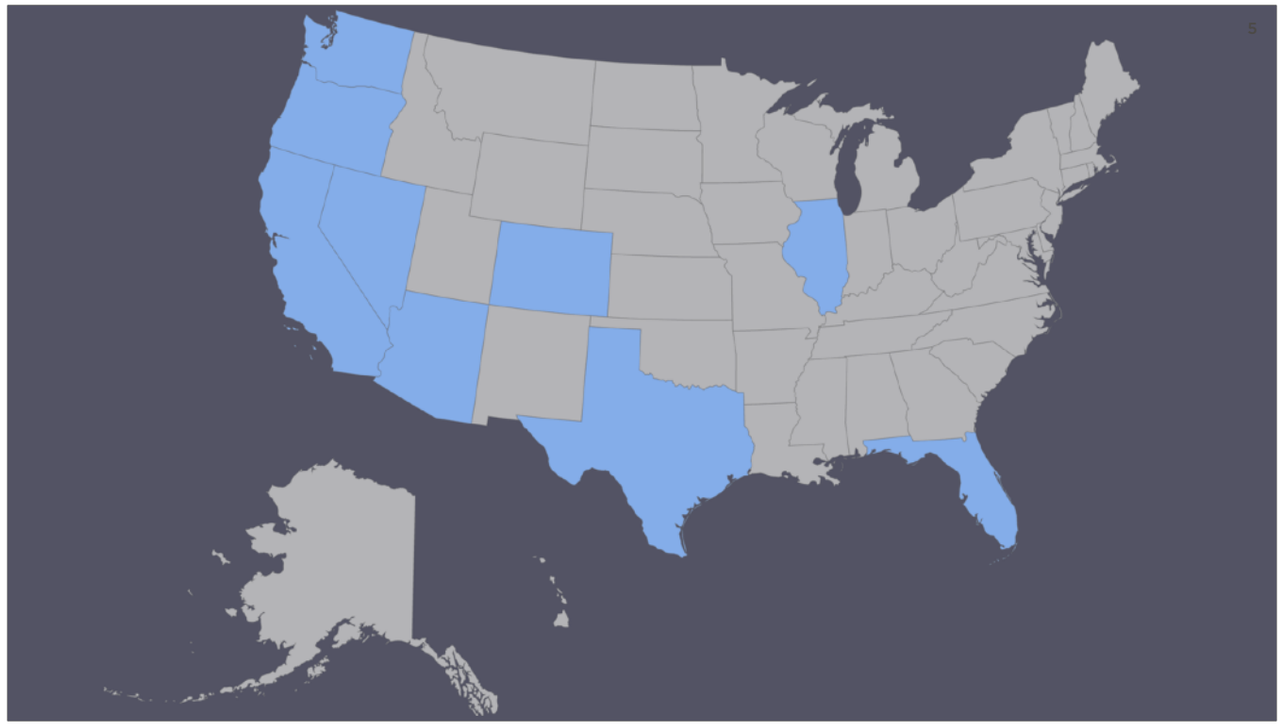


I was a remote CTO, so I wasn't in Denver. I was here. This is Peacock Lane. Portland's Christmas Street.
It's two weeks 'till Christmas, and I get a call from the CEO.

It's over. He's made the decision that we need to close the doors and begin layoffs.
The first batch of employees needs to go right away.... Scrooge would be so proud.

It wasn't a complete surprise. We were in the midst of a recession, and our lead investors didn't have deep pockets.

We'd been told by major banks that our data on their properties was better than their own
But didn't have enough runway to close enough sales.



We had also had discussions to be acquired by other real estate data companies. But there was concern about our limited coverage.
We started in Colorado, and we honed our architecture and business model there before expanding. But each state required lots of changes. Frustratingly, our velocity slowed with each state even as we were growing the team.

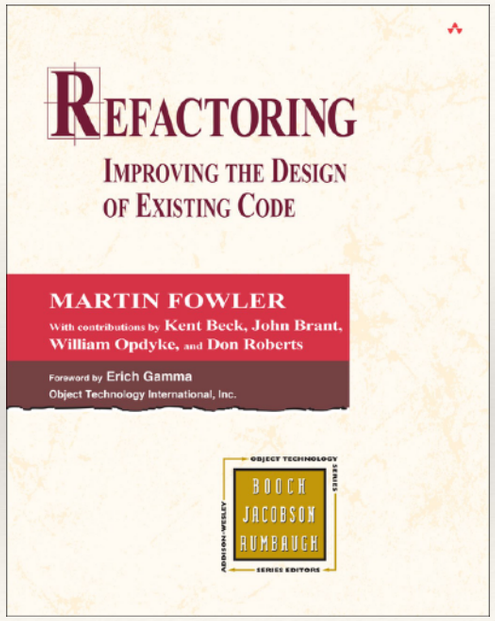
I think there's a lot to learn from failure, and I felt personally responsible. I wanted to learn what I had done wrong as the CTO.

I took some time off to think about that.

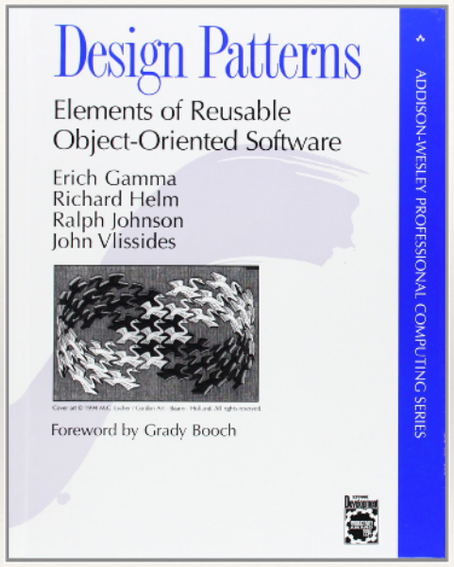
What makes something adaptable?

This is what I wanted to know. What did we do wrong that made our architecture inflexible? Why was it so hard to adapt to the new requirements of each state?

I did what I thought was logical.. I went to my bookshelf and dug into some of the most respected books in software history.



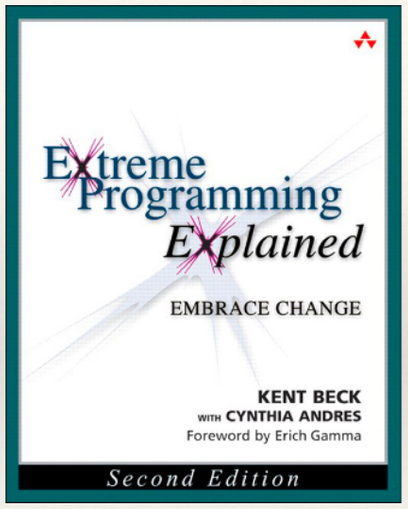
Books like this.


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

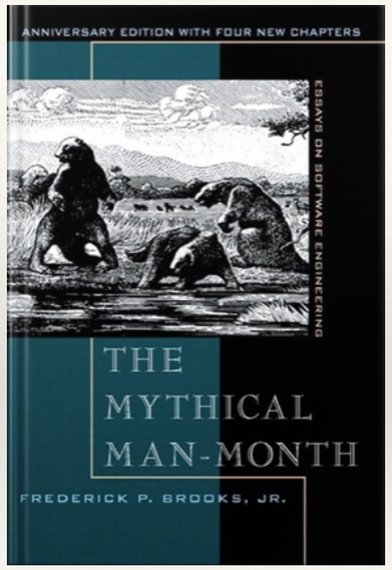
And this.

And newer ones like this.

I also read books on construction, process, and project management.

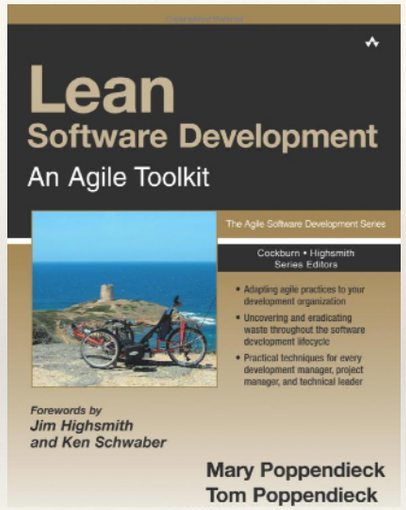


Books like this...



And this...

You folks in PDX1 know I read an annoying quantity of books.


Lean
Software Development
An Agile Toolkit
The Agile Software Development Series
Cockburn • Highsmith
Series Editors
• Adapting agile practices to your development organization
• Uncovering and eradicating waste throughout the software development lifecycle
• Practical techniques for every development manager, project manager, and technical leader
Forewords by
Jim Highsmith
and Ken Schwaber
Mary Poppendieck
Tom Poppendieck

And this.

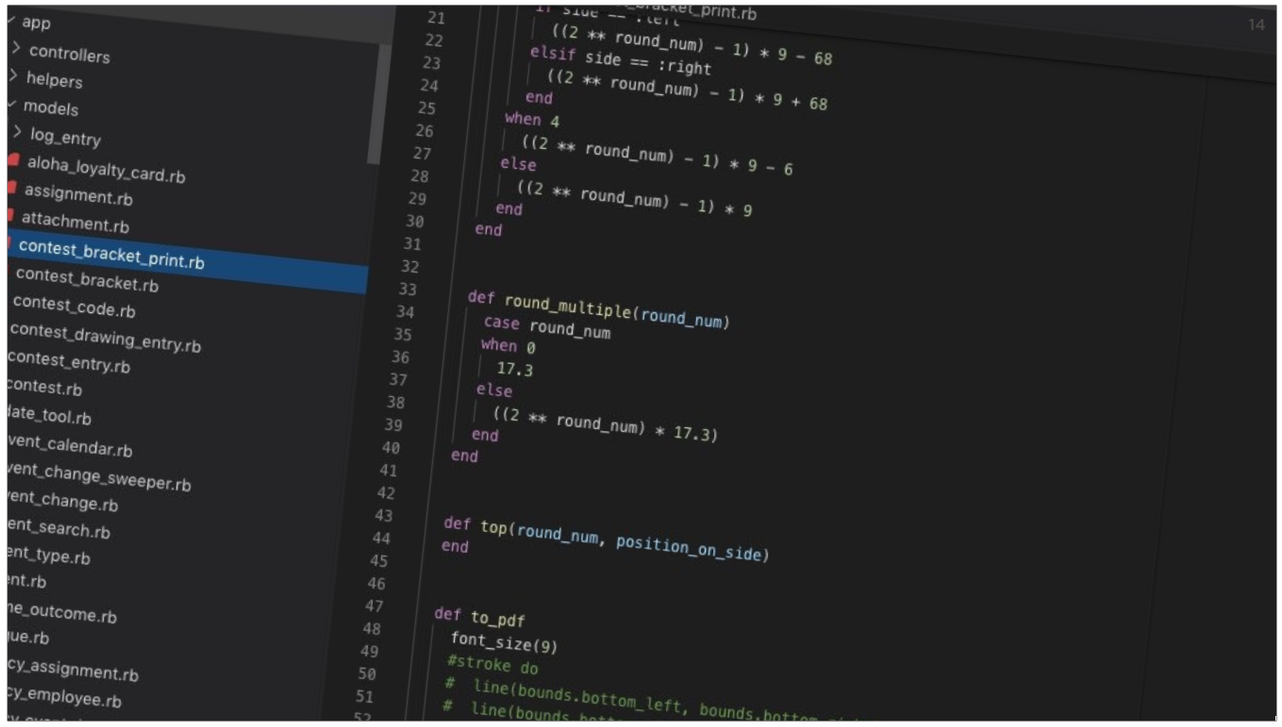
But really... how does adaptation work?

I got to the end of this process and was ultimately disappointed. Not because these books weren't good.

They're all excellent. I recommend every single one.

But they didn't really answer my question. We had largely been following this advice. We weren't perfect by any means, but I didn't see huge gaps in the quality of our code or our processes...

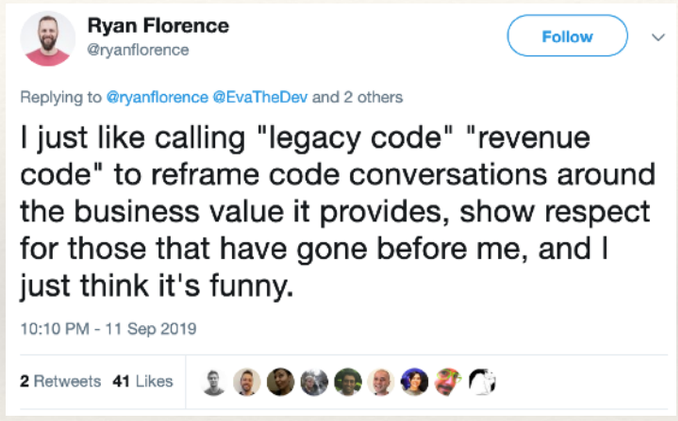
I had read many of these books before, and the wisdom of the others are so thoroughly believed that the wisdom has diffused into the culture. You already know the gist without having read them.



Around this time, I got another call. This time from an old consulting client. They were calling about software I'd written for them in Rails 1.0, when I didn't know what I was doing, and when they didn't know what they'd wanted. Like Frankentstein's monster, it'd rear its ugly head about once a year with an urgent problem.

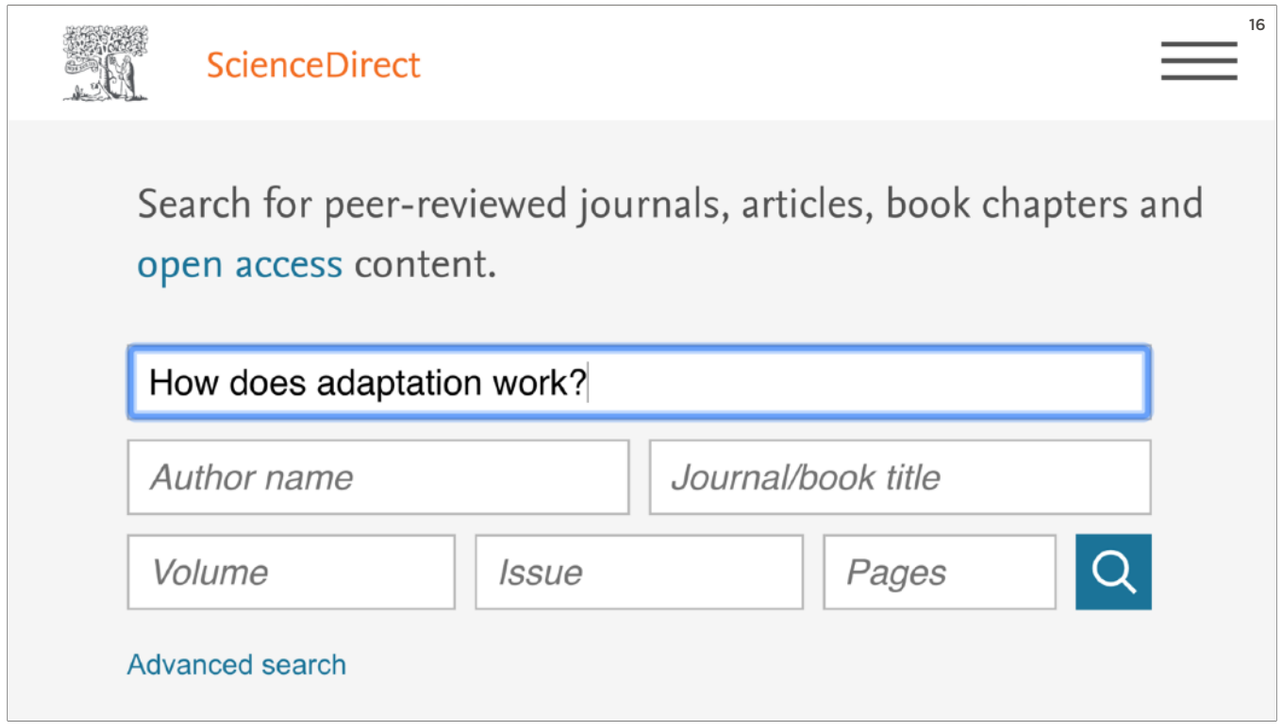
I am not proud of that code. Look at it. Not a comment in sight. Method names don't make sense. You've got more than a dozen magic numbers on a single screen. There's even a method there at the bottom with no implementation at all.

But you know what... as far as I can tell, this ugly code is is running in production right now, almost 10 years later.



I saw this Tweet recently. I like it. That code has survived because it has provided real value to the business. It's not technical debt, it's revenue code. Terrible, terrible *revenue* code.

Being reminded of the code that wouldn't die made me wonder. What if we failed not because we didn't follow best practices... but because we did.... so I broadened my search.



I figured biology would be a good place to start. Who knows more about adaptation and evolution than biologists, right? My wife is a teacher and a lactation consultant, so I borrowed her keys to the walled garden of scientific knowledge... and started digging.

Complexity

Robustness

Evolvability
(variation)

Whitacre, 2010 Theoretically Biology and Medical Modeling

Right away I ran into some interesting things which contradicted the software books.

This paper from 2010 is interested in how adaptive systems like species, brains, and ecosystems work. It takes as a given that successful systems are defined by 3 things: Robustness, which for biologists means something slightly more specific than it does for us engineers... it has to do with consistency of traits over time, Evolvability, which biologists also call "ability to innovate". And then Complexity.

<Record Screech>

Wait, what... one of these things doesn't seem to fit. If we think in terms of goals for our software... of course we want it to be robust. And evolvability is obviously desirable. But complexity?

The Nature of Complexity

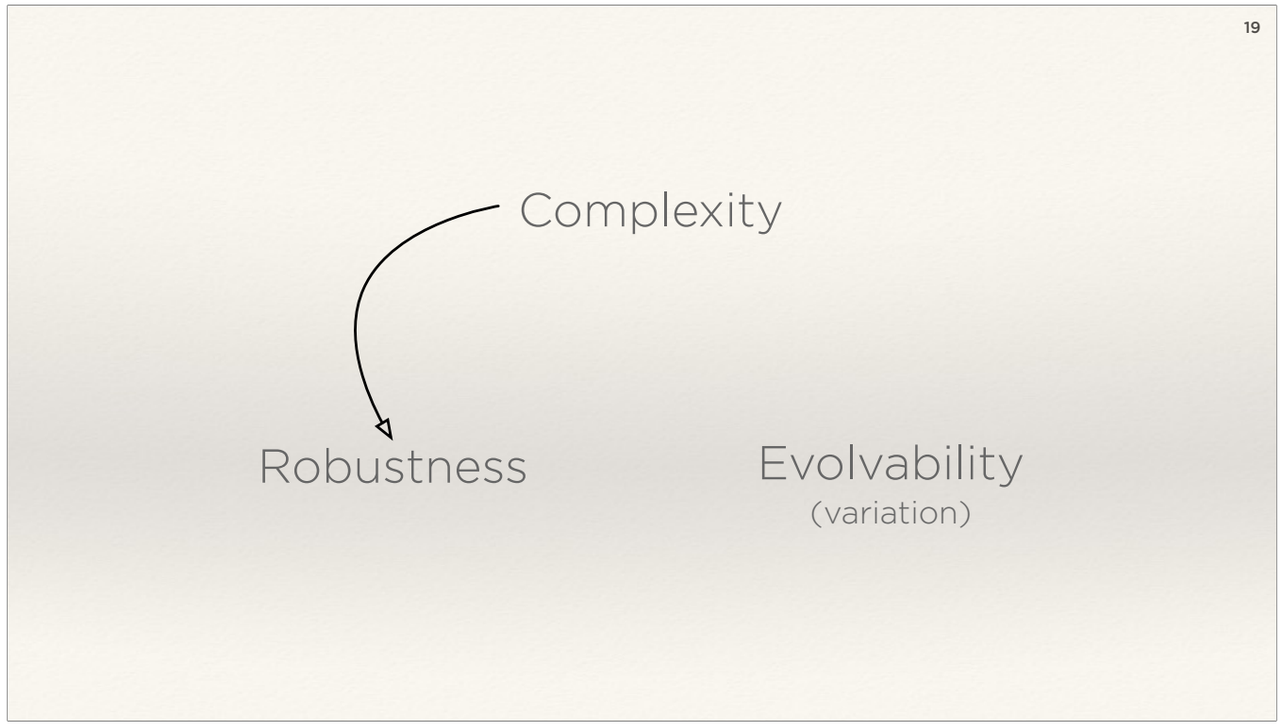
This book is about how to design software systems to minimize their complexity. The first step is to understand the enemy. Exactly what is "complexity"? How can you tell if a system is unnecessarily complex? What causes systems to become complex? This chapter will address those questions at a high level; subsequent chapters will show you how to recognize complexity at a lower level, in terms of specific structural features.

The ability to recognize complexity is a crucial design skill. It allows you to identify problems before you invest a lot of effort in them, and it allows you to make good choices among alternatives. It is easier to tell whether a design is simple than it is to create a simple design, but once you can recognize that a system is too complicated, you can use that ability to guide your design

I thought complexity was the enemy. Here's Chapter 2 of the Ousterhout's book.

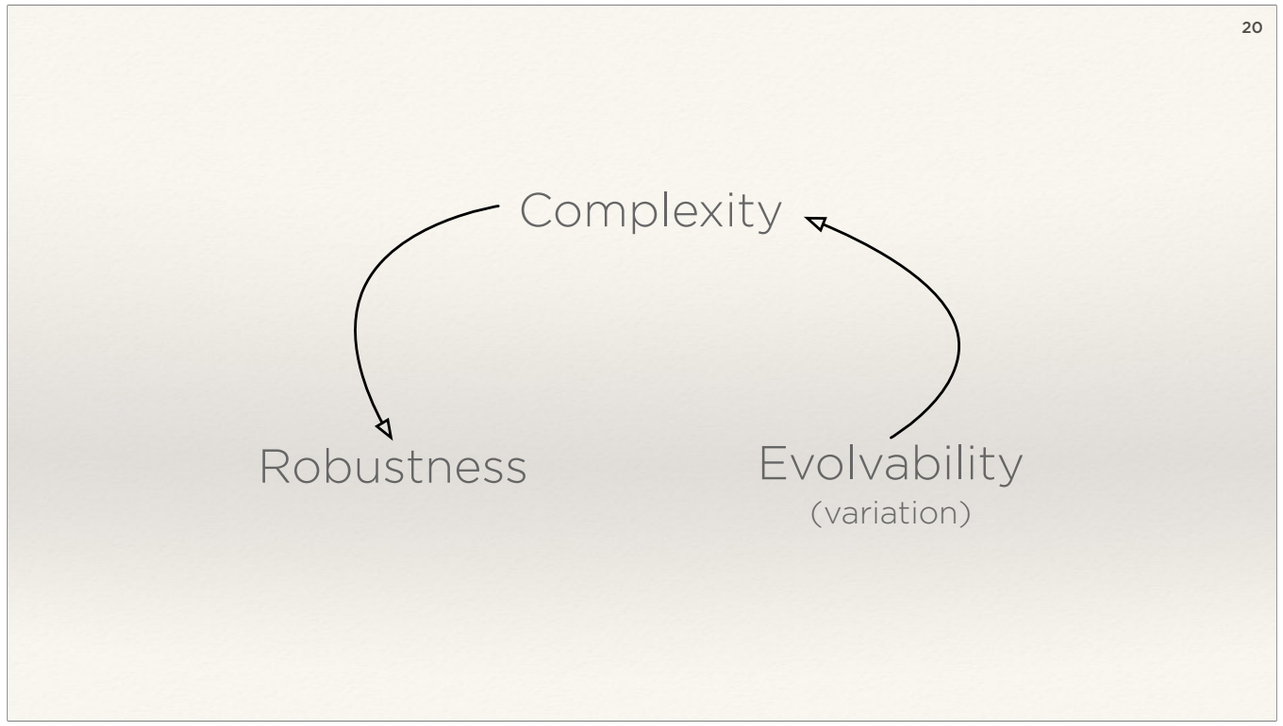
Chapter 1... also about complexity. It comes up again in Chapter 3 and in 4-21. And this book doesn't have a narrow focus. It aims to be a summary of a broad range of best practices of software. And John isn't alone... he's summarizing very succinctly what a lot of other experts are saying.

So we're on to something. Reasonable minds can disagree, and now that we have disagreement, we can learn something... let's see what the evolutionary biologists have to say.



It starts with a somewhat obvious observation. The environment isn't static. Being robust means being prepared for a wide variety of risks.

It's somewhat of a tautology, but it follows that if something has survived over a longer timeframe, then it has embedded solutions for a wider set of rarer threats. At a minimum, there is strong correlation between robustness and complexity such that robustness is *because of* complexity. If your tolerance for applied math is higher than mine, then you should look into something called the Law of Highly Optimized Tolerance.



The next observation, also quite simple, is that Adaptation leads to complexity. This is basic Darwinian natural selection stuff. Incremental adaptations create higher order forms, which are almost by definition more complex than their predecessors.

As I said, most of these relationships were of interest to me, but this paper passed this off in the intro as if it were common knowledge in the field.

For them, it's only this bottom link here that really gets interesting. And it's obvious why. Robustness and Evolvability aren't different stages in a continuous cycle, they're opposites.

The Mystery of Evolvability

How can robust, highly evolved species continue to evolve?

Robustness ⟷ **Evolvability (variation)**

Low genetic variation
Stability
Sameness
Predictability

High genetic variation
Experimentation
Novelty
Unpredictability

Remember, for biologists, robustness means something about the consistency of genetic code. In other words a relatively stable set of traits. But evolvability is the opposite of that.... it's variation and novelty.

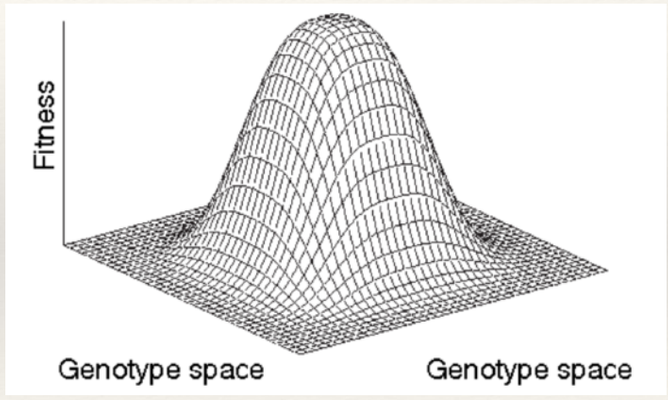
Evolvability is about variety, experimentation and novelty.

Once you've got a complex, highly evolved creature.. what makes it keep evolving?

Fitness Landscapes

Sewall Wright

American Geneticist, 1889-1988

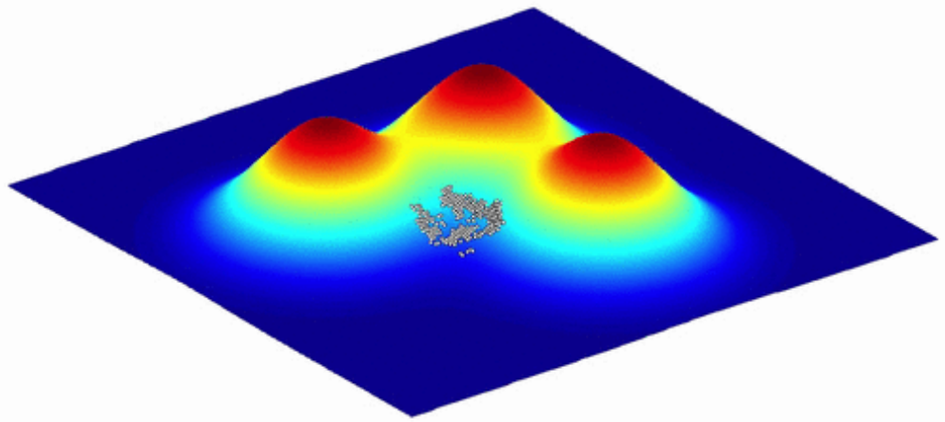


This question dates back to some of the earliest geneticists.

This is Sewall Wright. He was interested in trying to bring some mathematical rigor to genetics. He is credited with the idea of the Fitness Landscape

- X and Y: genetic recipe. Z: fitness
- Genetic variation at birth places organism randomly
- Natural selection favors those in the sweet spot
- Cause a species to converge on a beneficial trait

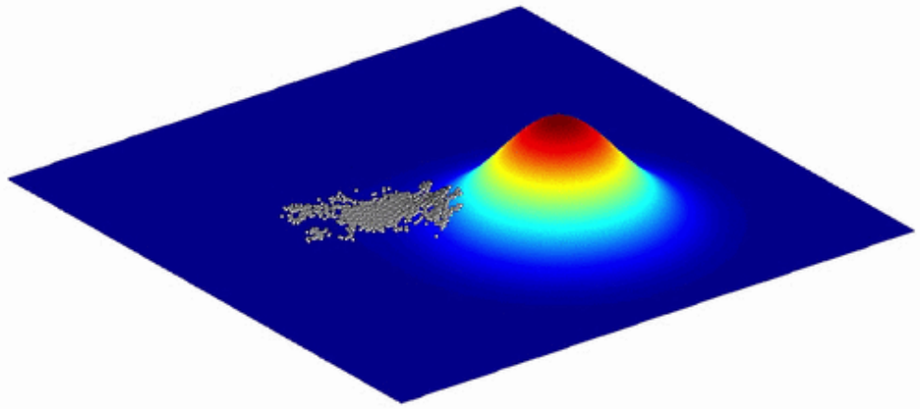
Static fitness landscape



Population size, N = 2,304
Mutation rate, μ = 0.05 per trait

Visually, it looks like this. This is a visualization created by two Michigan State University Students.

Dynamic fitness landscape



As the fitness landscape changes, the population evolves to track the peaks

Population size, N = 2,304
Mutation rate, μ = 0.5 per trait

As we're aware, though, the landscape isn't static. The very definition of fitness changes with the environment.

For those of who aren't biologists, let's consider a business example. At one time SEARS evolved atop the mail order catalog peak, which is now gone.

And Google is happily atop the Internet advertising peak. Perhaps Amazon is on the e-commerce peak. And neither of those is permanent.

It's hard enough that the landscape is dynamic... but there's another problem....

Fisher's Fundamental Theorem of Natural Selection

The **better adapted** you are,
the **less adaptable** you tend to be

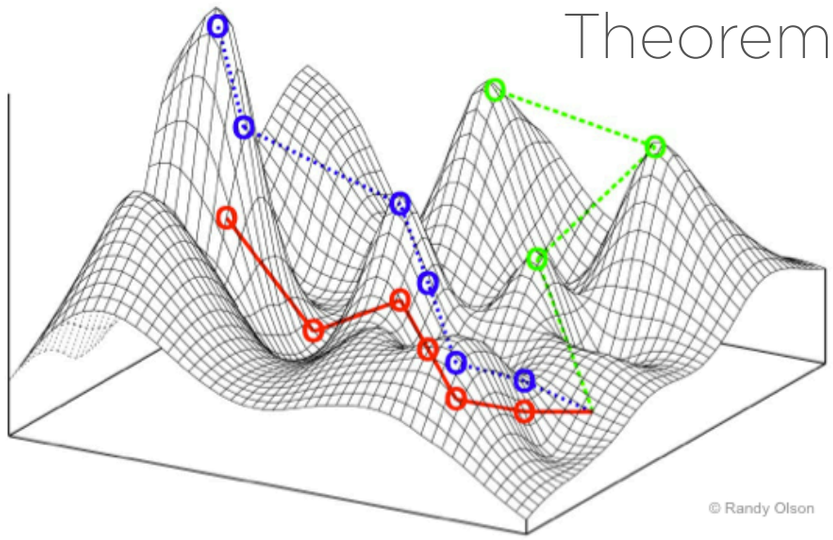
Ronald Fisher
British Geneticist, 1890-1962

This is Ronald Fisher, a contemporary of Sewall Wright's. One of his more well-known contribution is known as the Fundamental Theorem:

- "The better adapted you are, the less adaptable you tend to be".

This is true at many levels
- Businesses that ossify
- People who stop learning new skills

Fisher's Fundamental Theorem of Natural Selection



When you're at the bottom, everywhere is up.

Let's look at what this means.

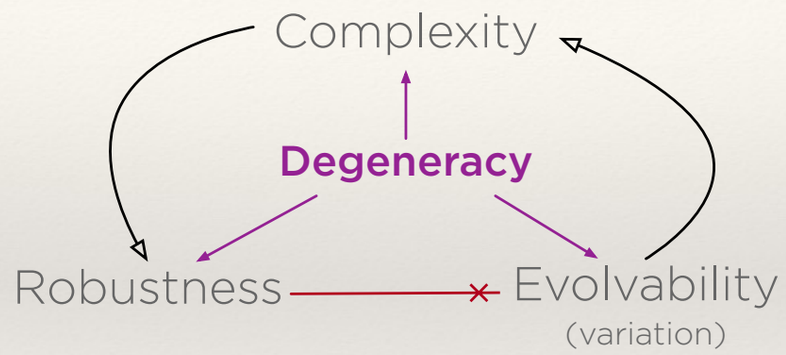
When your fitness is poor, your offspring benefit greatly from random variations.
But when you've reached a peak, the odds are against your offspring having success if you vary wildly. So you don't.

In other words:

Evolvability slows as fitness increases.

As I said. This is a problem as old as genetics.

But this paper points to a solution.



The authors point to recent research across a number of fields that show that a concept called degeneracy underlies all 3 properties, and provides a way to allow evolvability to continue despite robustness.

So what's degeneracy?

Redundancy

- Multiple **identical** structures
- Each producing the same outcome

Degeneracy

- Multiple **different** structures
- Each producing the same outcome

Pluripotentiality

- Single structure
- Producing **multiple outcomes**

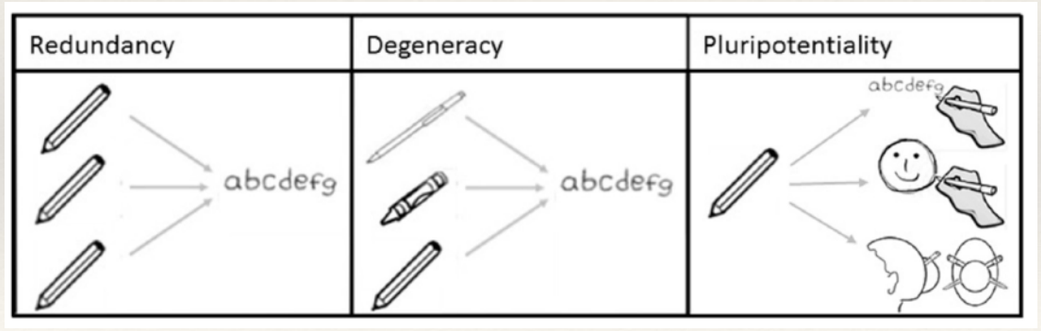


diagram credit: Mason, Dominguez, Winter, Grignolio
Hidden in Plain View: Degeneracy in Complex Systems, **BioSystems** 128 (2015)

Degeneracy is best understood in comparison to redundancy.

Where redundancy is structurally identical copy, degeneracy is a functionally identical copy with a distinct structure.

A degenerate system, then is something which has many different subsystems which can do the same thing.

CONTEXT: SAT/Goverment form.

For completeness, I've also included the diagram for Pluripotentialilty. Scientists have found that some of the most powerful adaptive systems exhibit a trait they call "bow tie", which is when they have components which are both degenerate and pluripotent.

What Biologists Got Wrong

1. It's robustness of **functions, not traits** that matters.
2. Degeneracy leads to distributed robustness of functions.
3. Evolvability is enhanced by the unpredictability of degeneracy.

Here's what I think happened. Early geneticists got distracted by their racism.

They focused on visually defined **traits**. But for evolution, it's not what it **is** that matters, it's what it **does.**

In the last 10-15 years, researchers have begun to understand that the robustness in most complex adaptive systems is actually distributed robustness.

This has the same effect as redundancy, which we engineers understand well. It also introduces some unpredictability in the solution... and this leads to something I don't think we understand well... innovation.

Degenerate **Innovation**

"The propensity to innovate is enhanced in systems that are robust in many of their core functions yet flexible in how those functions are carried out."

Kirschner and Gerhart

Evolvability, Proceedings of the National Academy of Sciences 95:8420–8427

"The propensity to innovate is enhanced in systems that are robust in many of their core functions yet flexible in how those functions are carried out."

Think about how well that describes your company.... and how well you think you are innovating.

Fisher and Wright understood that there was value to randomness. It's how you avoid getting stuck on a local optimum. The beauty of degeneracy is that it allows randomness, but not too much.

Now, at this point, you might be thinking that this degeneracy idea is an interesting quirk from some remote corners of academia.... but that's a mistake. There is some evidence that degeneracy is **the defining mechanism** of evolvability.

1. Genetic code (many different nucleotide sequences encode a polypeptide)
2. Protein fold (different polypeptides can fold to be structurally and functionally equivalent)
3. Units of transcription (degenerate initiation, termination, and splicing sites give rise to functionally equivalent mRNA molecules)
4. Genes (functionally equivalent alleles, duplications, paralogs, etc., all exist)
5. Gene regulatory sequences (there are degenerate gene elements in promoters, enhancers, silencers, etc.)
6. Gene control elements (degenerate sets of transcription factors can generate similar patterns of gene expression)
7. Posttranscriptional processing (degenerate mechanisms occur in mRNA processing, translocation, translation, and degradation)
8. Protein functions (overlapping binding functions and similar catalytic specificities are seen, and "moonlighting" occurs)
9. Metabolism (multiple, parallel biosynthetic and catabolic pathways exist)
10. Food sources and end products (an enormous variety of diets are nutritionally equivalent)
11. Subcellular localization (degenerate mechanisms transport cell constituents and anchor them to appropriate compartments)
12. Subcellular organelles (there is a heterogeneous population of mitochondria, ribosomes, and other organelles in every cell)
13. Cells within tissues (no individual differentiated cell is uniquely indispensable)
14. Intra- and intercellular signaling (parallel and converging pathways of various hormones, growth factors, second messengers, etc., transmit degenerate signals)
15. Pathways of organismal development (development often can occur normally in the absence of usual cells, substrates, or signaling molecules)
16. Immune responses (populations of antibodies and other antigen-recognition molecules are degenerate)
17. Connectivity in neural networks (there is enormous degeneracy in local circuitry, long-range connections, and neural dynamics)
18. Mechanisms of synaptic plasticity (changes in anatomy, presynaptic, or postsynaptic properties, etc., are all degenerate)
19. Sensory modalities (information obtained by any one modality often overlaps that obtained by others)
20. Body movements (many different patterns of muscle contraction yield equivalent outcomes)
21. Behavioral repertoires (many steps in stereotypic feeding, mating, or other social behaviors are either dispensable or substitutable)
22. Interanimal communication (there are large and sometimes nearly infinite numbers of ways to transmit the same message, a situation most obvious in language)

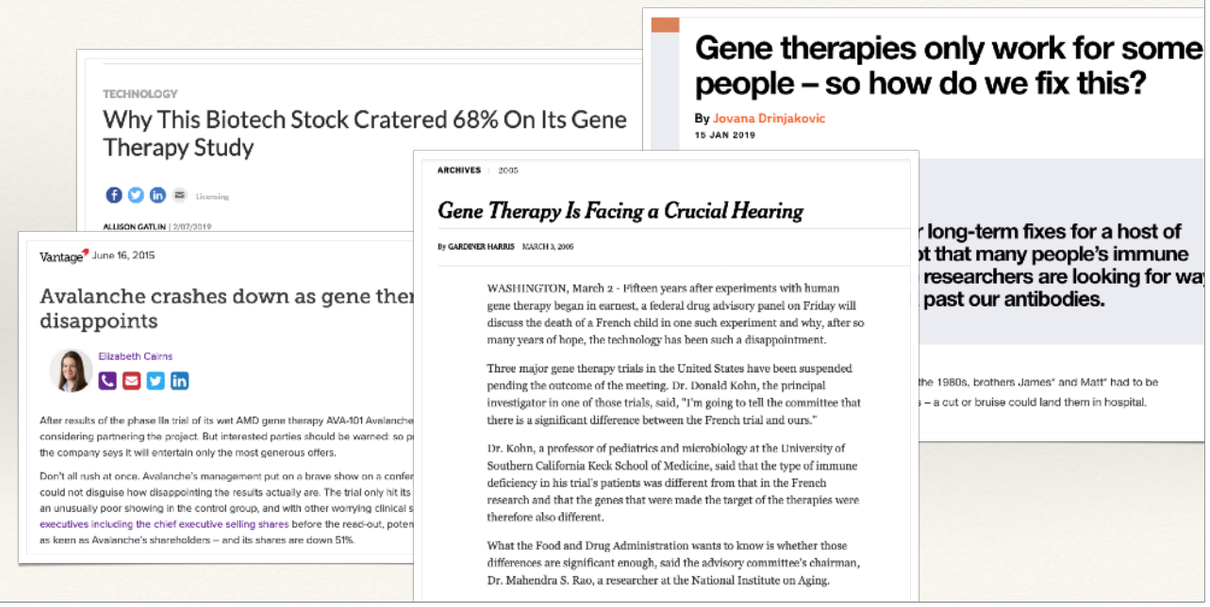
excerpt *Degeneracy and complexity in biological systems*.
Edelman and Gally, 2001

For one thing, it's literally **everywhere** in nature.

This list is hard to read, but that's the point.
This paper outlines 22 distinct levels in which degeneracy plays a role in animal biology.

Let's look at a few:

Degeneracy: Gene Therapy



Once upon a time, the biggest money in the stock market was on biotech.
On the premise that successfully mapping the genome would lead to all kinds of miracle drugs.
For the most part, it hasn't panned out. yet.

The reason? Degeneracy. Very few traits are controlled by a single gene, most are multiple genes acting in concert, **depending on context.**

This one is my favorite.. it's probably the easiest to understand.

Food Sources. Our bodies depend on a variety of nutritional components. Take healthy fats:

- Cali or Mexico: Avocados

But Avocados don't grow well in the climate here in Oregon.

Degeneracy: Nutrition

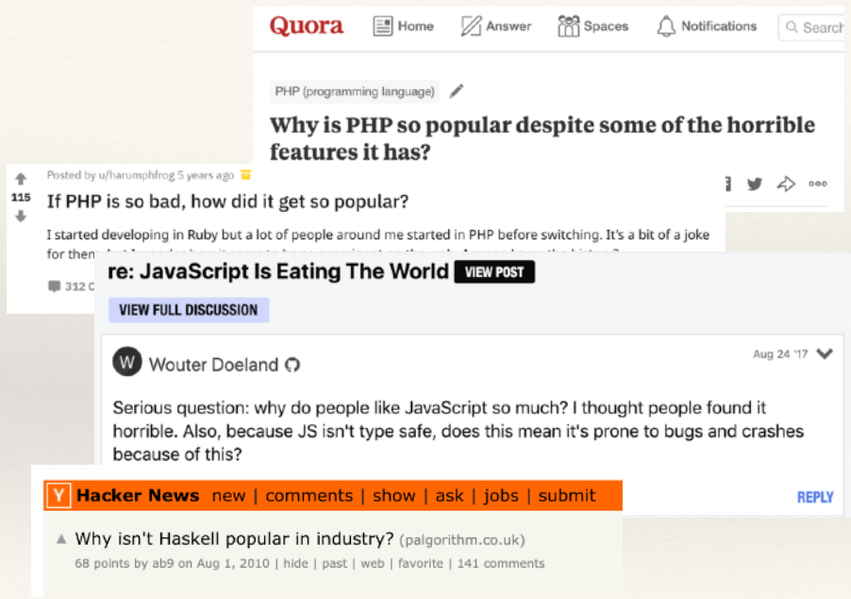
- So in Portland, we turn to Voodoo Donuts for our healthy fats.
- Again, I'm a developer not a nutritionist or scientist.. consult your
- doctor for the nutrition that's right for you.

Degeneracy: Immune System

The degeneracy of the immunoglobulins made by an animal ensures that the animal possesses the ability to make antibodies that protect against essentially any foreign, infectious agent.

– Edelman and Gally
Degeneracy and complexity in biological systems
PNAS 2001

This is not variation and natural selection.
This is not a trip "back to the drawing board" to respond to a new virus.



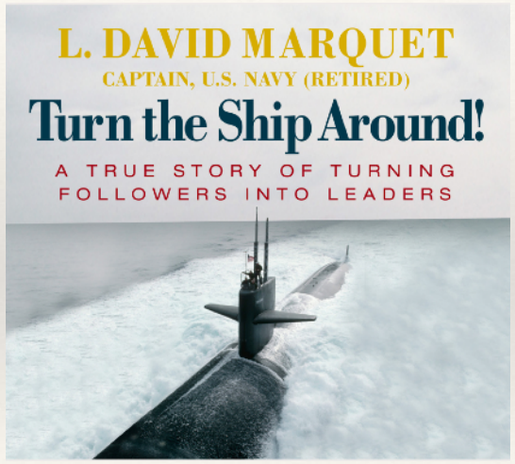
But around the same time, there was an incredible flourishing of new languages and databases in computer science, but they didn't unseat Javascript and PHP. In fact, each seemed to grow more popular... and I realized... huh, maybe degeneracy might explain this.

Degeneracy: Languages

Efficient, Consistent, Robust. Better?	**Degenerate, Pragmatic, Multi-Paradigm. Worse?**
Latin	English
Lua, Erlang	Javascript, PHP
Haskell, Rust	Go

Let's take a few examples here.
Left: adjectives like: pure, academic, consistent.
Right: pragmatic, sloppy, POPULAR

Degeneracy: Management

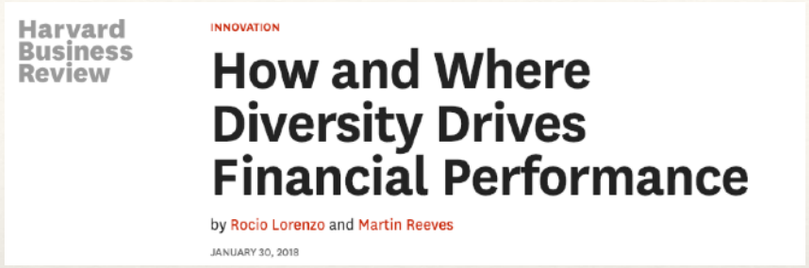


When you start looking for it, you'll start realizing that degeneracy can be found in lots of social situations.

Traditional military thinking says you should centralize decision making at the top of efficiently-designed pyramids.
David Marquet did the opposite. Upon inheriting the submarine with the worst safety record in the Navy,
he pushed authority downwards so that a decision could be made by any number of people.

This is inefficient in the traditional sense, but it is functionally robust, you have many, partially overlapping redundancies.
Spoiler alert for those of you who haven't read the book, but they not only became the safest sub, but the top performing.
It's clear that the improved agility and innovation was a key part of that.

Amazon's practice of small "pizza teams', and in fact the entire practice of Microservice-based teams likely succeeds for similar reasons.


Harvard Business Review

INNOVATION

How and Where Diversity Drives Financial Performance

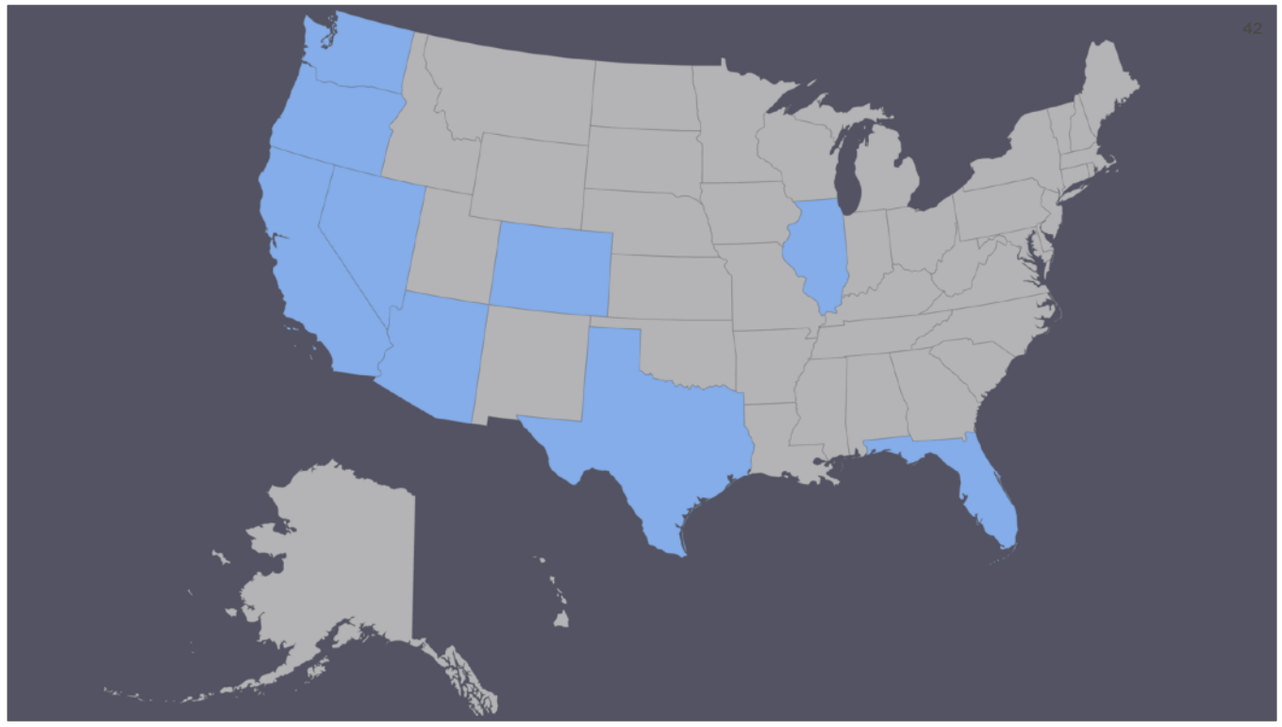
by Rocio Lorenzo and Martin Reeves

JANUARY 30, 2018

We found that indeed there was a statistically significant relationship between diversity and innovation outcomes in all countries examined. Furthermore, the more dimensions of diversity were represented, the stronger the relationship was, although the precise patterns of diversity and performance were different across cultures. We also found that diversity had gained momentum as a topic in more than 70% of the enterprises surveyed, especially in developing economies.

Here's an article from Harvard Business Review, showing research that bears this out. Focusing on diversity is having a significant effect on innovation outcomes.

People from different backgrounds, different cultures, people with different approaches, who are capable of the partially overlapping outcomes. Diversity and degeneracy are the same thing.
The irony here is amazing. When you consider that genetics and social darwinism were tools of racist policy



So what are the implications for our startup back in Denver? What did I do wrong?

We assumed a single, efficient solution would be best, and we started in Colorado, so we based our assumptions on that example. As we expanded, we incorporated new information into that unified model. Just like Bob Martin advised us, we avoided duplication. And just like John Ousterhout said we should, we avoided making our architecture as complex as state laws were.

The answer, conveniently, was an ego-stroke... we were **too good for our own good.**

Frankly, though, I was having a hard time seeing how this would help.
This was just too convenient.... I was worried it wouldn't generalize. So I wondered.. is avoiding perfection really **a thing** in evolution?

I had always assumed that evolution was "iterative", you know... like agile, that variation and natural selection just make things get better and better.

I think a lot of people have this impression, and I think they get it from pictures like this.

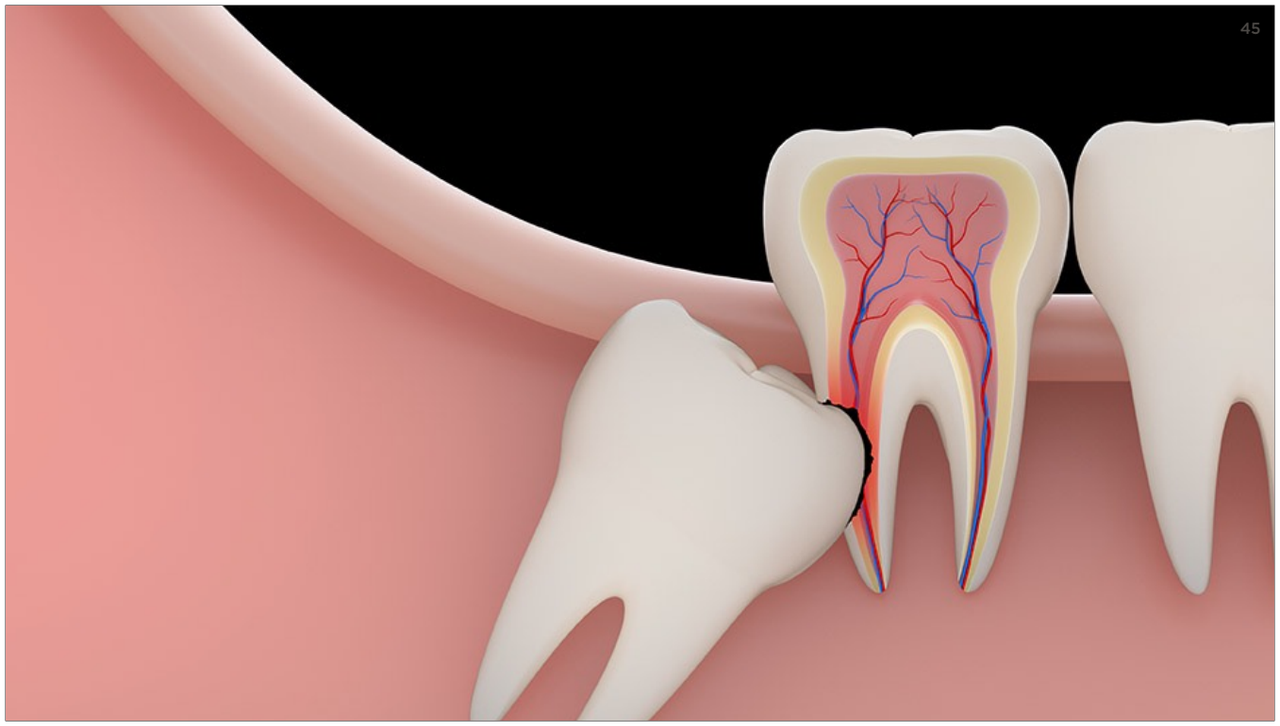
Katie-did. Like a color copy of its habitat. Super camouflage.
Looks like all the Katie-dids got together, and one said... you know.. I think we'd be nearly invisible if we made out wings look just like those leaves over there. I bet birds wouldn't eat us so often if we did that.

And like this...

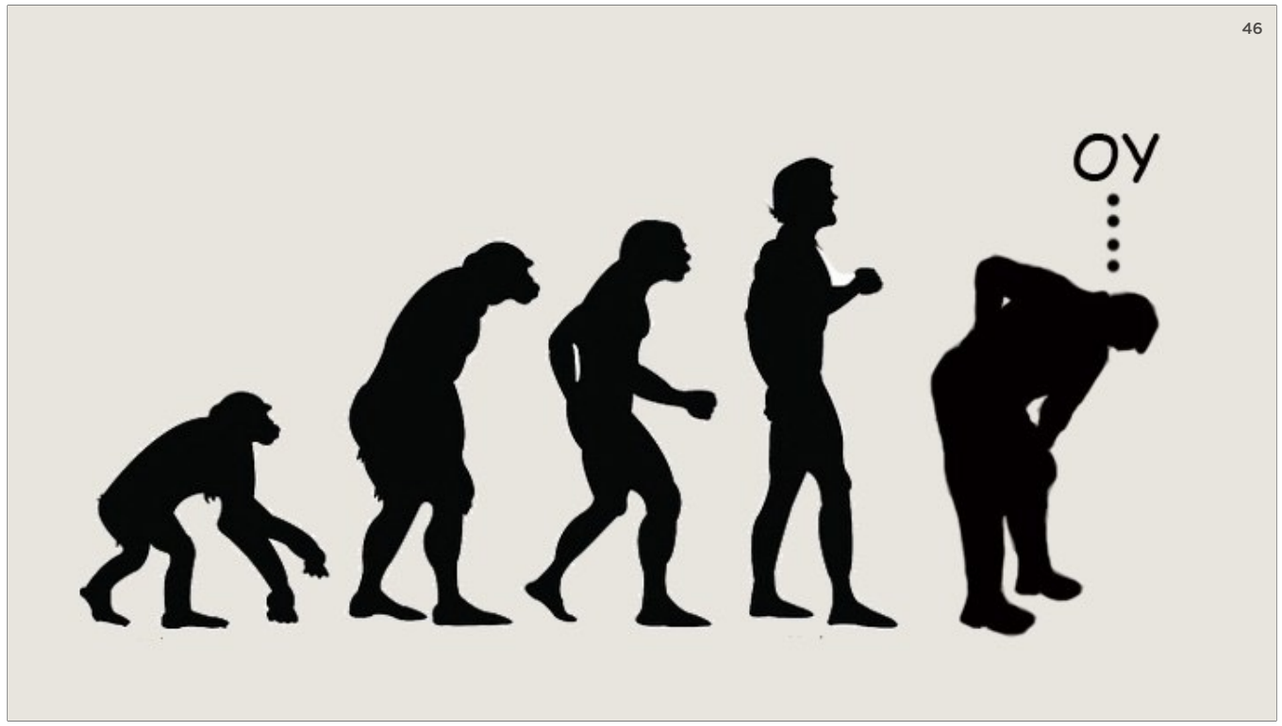
This fishing lure is so smart it’d be illegal to fish with it in many states.

Looks engineered. But the reason they are in textbooks and featured in movies is because they're exceptional. They are rare, interesting exceptions to the rule, which is varying degrees of mediocrity.

To an evolutionary biologist, poorly-engineered solutions are everywhere...

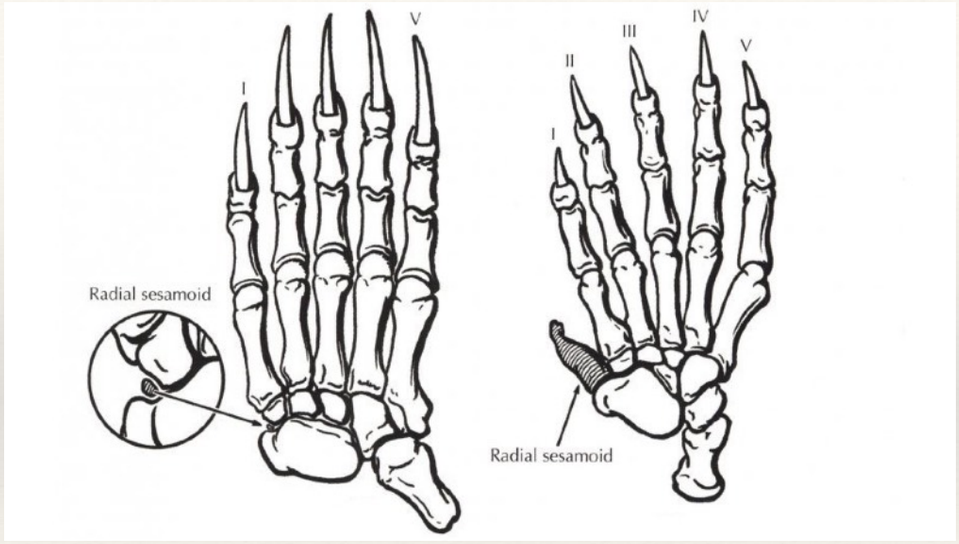


Take wisdom teeth.
Our species decided we need bigger and bigger brains.
And our brain cases have gotten so big that all our teeth no longer
fit in our jaws.
But.. get this. Instead of growing bigger jaws, or growing less teeth...
we decided we could just train some of us to pull out the extra teeth with pliers.
We invented orthodontia.



Any of you suffer from back pain? Or foot pain?
We switched our "tech stack" from quadruped to biped and didn't have time to re-architect our skeletons.

Pandas have evolved to live in an environment where their diet consists almost entirely of bamboo...
But they don't have thumbs. They have basically the same paws as your cat and dog... with one jury-rigged addition.



Conventional Bear Panda Bear

Left: Conventional bear. Right: Panda

- Thumb isn’t digit
- Bone spur off its wrist.
- Believed that the growth interrupted tendon
- First digit lost tendon. Thumb ineffective.
- Big toe has the same thing
- This is an inelegant mistake that just barely works.

If it had a desk, this would not be it.

Evolution:
- doesn’t start with a plan
- Doesn’t advance toward perfection.

This might be. Evolution:

- Hoards spare parts
- Loves using glue, duct tape and chicken wire
- Why?

- Because: Being ready for anything means nothing is irrelevant.

“Evolution is a tinkerer.”

–François Jacob, *Science*, 1977

Evolution isn't a finely tuned machine, and it isn't a highly optimized search algorithm for perfection.
Evolution is a tinkerer.

The Katie-did was in my textbook, and the anglerfish gets featured in Finding Nemo
because they are UNUSUAL. Rare is interesting.

Evolutionary biologists know that the real world is absolutely full of sub-optimal, partial solutions.
From Ronald Fisher's perspective, species get stuck on local optima all the time.

Engineering	vs.	Evolution
Plan and optimize		Tinker and experiment
Efficiency is the highest goal		Survivability is the highest goal
Redundancy when risks are well known		Degeneracy pretty much all the time
Predict and avoid failure		Fail constantly, but keep it survivable
Seek known and predictable paths		Embrace randomness

So, what have we learned...

In terms of management, in engineering we depend on a plan and we go to conferences to figure out how to get them done faster. Meanwhile Evolution is in the garage goofing off.

In terms of goals, engineers are ambitious being better, faster and stronger is what matters perfection is the goal and mistakes are avoided. Evolution sounds more like Don Quixote, "there is a remedy for everything except death."

In fact, Don Quixote would be a pretty good spokesman for evolution. It's all adventure, dreaming the **impossible** dream.. even if it means tilting at windmills.

~~The Wisdom of Nature~~

The Hubris of the Engineer

I know some of you look at those two lists and think... yeah but... what's really wrong with the engineering mindset. My CEO would never hire Don Quixote as a CTO.

At best it'd be annoying. At worst, it'd be dangerous, or so it seems.

I might have piqued your curiosity about degeneracy and the value of diversity, but you're not convinced there's anything wrong with the engineering side of the ledger. The rest of this talk is for you... because I am you, and like you probably do, I think I'm good at my job. I know what I'm doing.

But I'm going to share some things I've been reading, that make we wonder what the limits of that are... and if we all need a big dose of humility.

King Frederick William I

The "Soldier King" of Prussia (1688-1740)

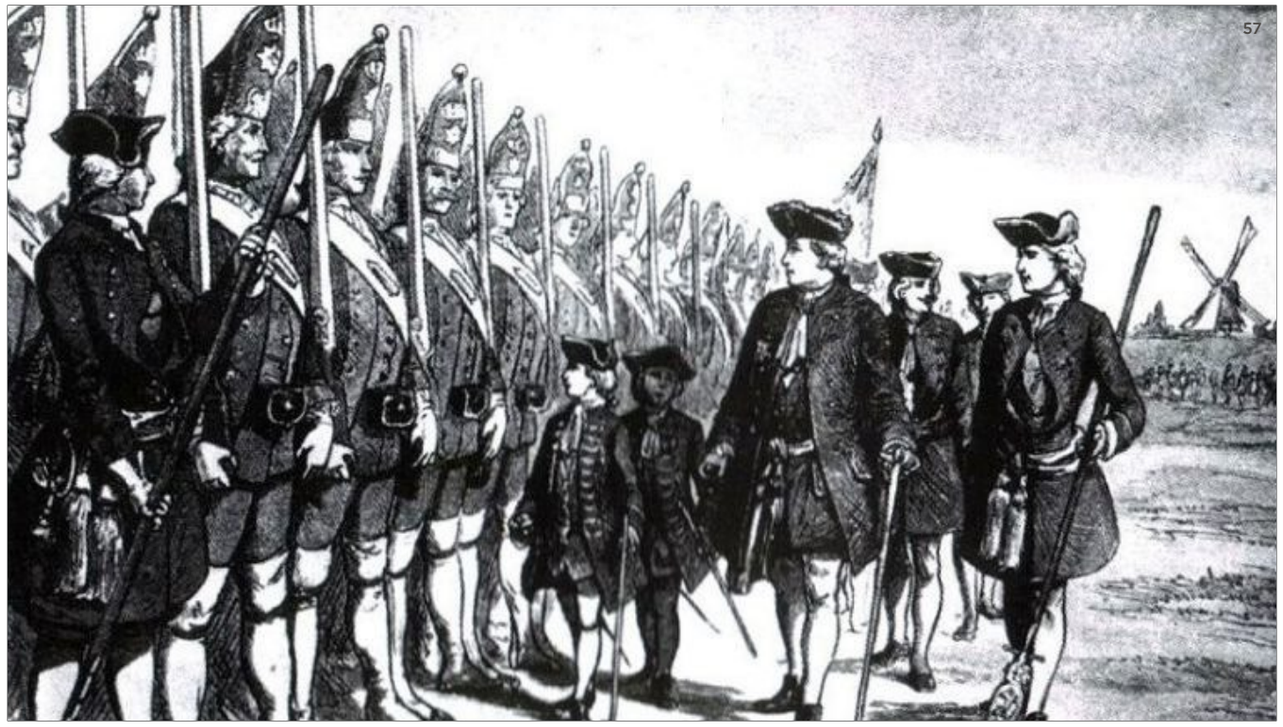
I think it can sometimes be hard to learn from lessons that are too familiar, so I want to take a little historical detour... this will be quick, I promise.

This is King Frederick William the 1st. Known as the "soldier king" of Prussia. You can think of him as the Mozart of military bureaucracy. As a 6 old he was given command of his own regiment of soldiers for drilling.

Obsessed with standardization, control and efficiency.
Technology enabled uniforms to be identical.

Practices still dominant in all militaries.
"Prussian Blue"

The goose step originated under his rule. It spread from there to Russia and from there around the world.



High performance. Thought big soldier = best.
Specially selected families with tall children.
"Bred" a regiment of exceptionally tall soldiers.
Known as the Potsdam giants.


King Frederick William I
(1688-1740)

Paternal Despotism

Despite his militaristic tendencies, he never started a war.
Wanted economic modernization.
Prussia was backwater, and he wanted to modernize.
Left no part of society unmanaged.
Well-intentioned. Paternal despot. This was the ENLIGHTENMENT. THE AGE OF REASON.
The idea of God-appointed rulers was waning, and you had to do good to lead.
To do good, you needed to know what worked.

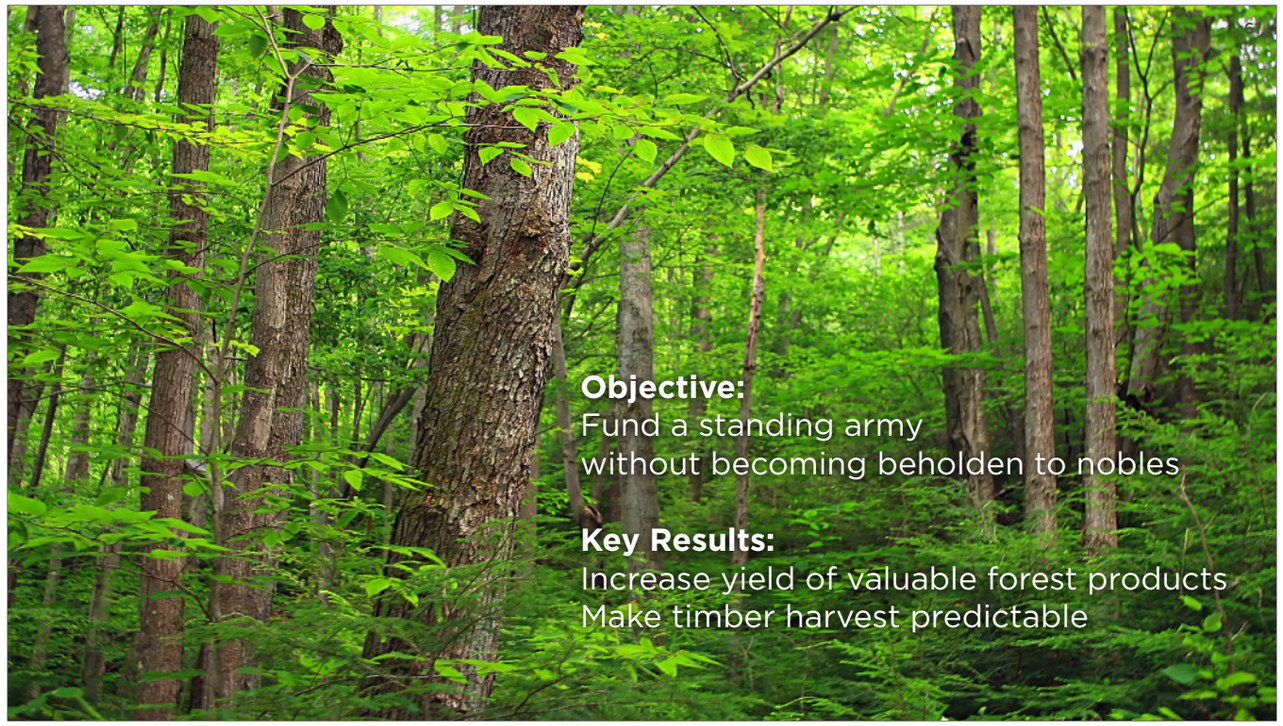
Apologize for history lesson... WHY AM I TELLING YOU ABOUT HIM?
Underrated influence of leadership culture.

Standardized, state-run schools

Non-religious. State funded. Compulsory schools.
Horace Mann took a trip.
Set up Massachusetts school system which became a national model.

We likely have business schools because of him.
If you've been to Wharton, you may have driven
through a town called King of Prussia, PA
It's either named after him or his soon.

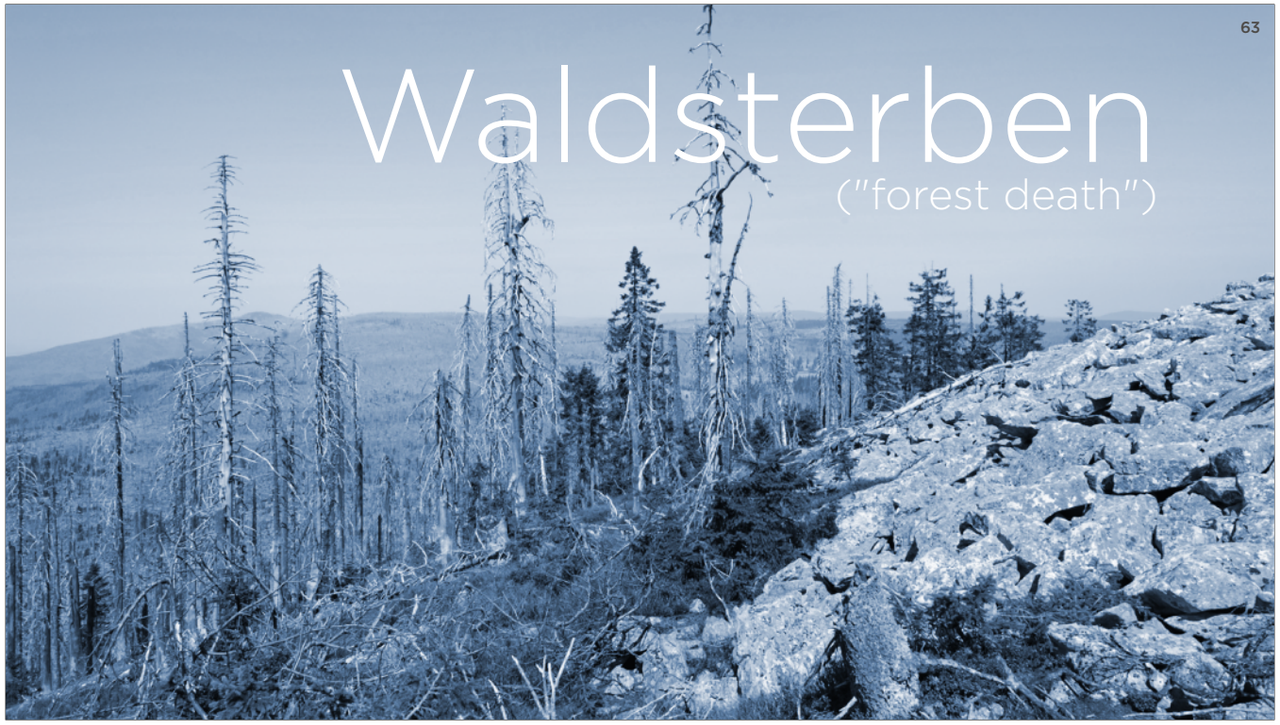
All of these things came from cameralism, the science of managing.



Important to manage: natural resources.
Wood is oil of this era. Depleted so fast.. danger of running out.
The practice: rotate forest plots.
By the time you got back to the beginning it'd hopefully grow back. <CLICK>
Problem: not predictable.
Didn't have OKRs, but maybe this is how they thought of it.

So they invented Scientific Forestry. The spreadsheeted all the things.

Here's what it looked like.
Deputy. Normalbaum: standard tree
Experiments. Consistent. Same Age. Aesthetic.
Like commander inspecting uniforms of cadets... inspect the trees.
Most important: Predictable. Know how many board feet.
Just like military, school, bureaucracy: spread like wildfire.
Gifford Pinchot. Roosevelt. USFS.
THIS WORKED WELL.... FOR 100 YEARS



And they needed a new word... Waldsterben.. "forest death".

It turned out their approach was myopic. The forest was complex, and they valued only one of its components. Everything else was a weed or a pest or a distraction, and deemed unnecessary. They didn't know it at the time, but all of those seemingly worthless components of the system were immensely valuable to keeping the soil, and by extension, the trees... healthy.

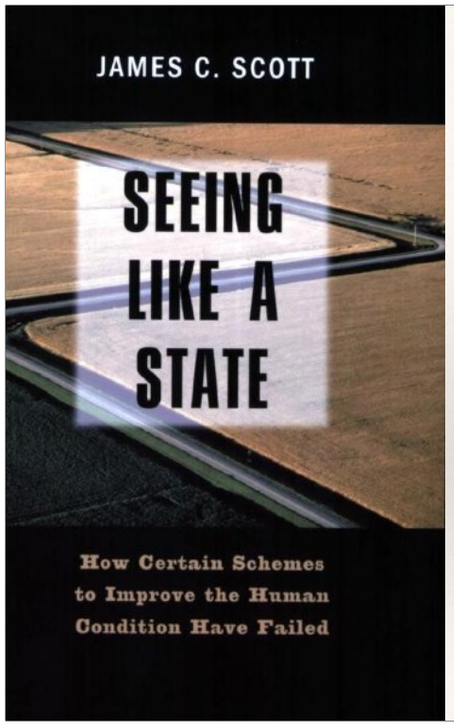
THEY HAD, **LITERALLY**, MISSED THE FOREST FOR THE TREES. LITERALLY.

“Every system has parts you don't understand.”

– **Aaron Longwell,** Just Now

I think about this story every time somebody complains about people wasting time at work.
Every system has parts you don't understand.

The Scientific Foresters thought they knew what they were doing for 100 years before they figured out they errors of their ways.



"...much of early modern European statecraft (was) devoted to rationalizing and standardizing what was a social hieroglyph into a legible and administratively more convenient format."

— James C. Scott

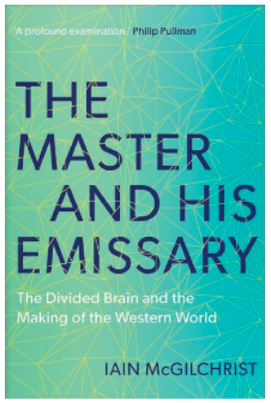
This story comes from this book.

Examples throughout history of well-intentioned planners who managers to not only fail to get their desired outcomes, but made the situation worse.

Examples from city planning, from government policy, and from business.

What they all have in common is a kind of engineering hubris... of a basic error in which the engineer at the controls had more confidence in their understanding that was warranted.

He emphasizes this idea of legibility. That because things are complex, there is a tendency to want to simplify, and that creates a myopic distortion that disconnects you from reality.



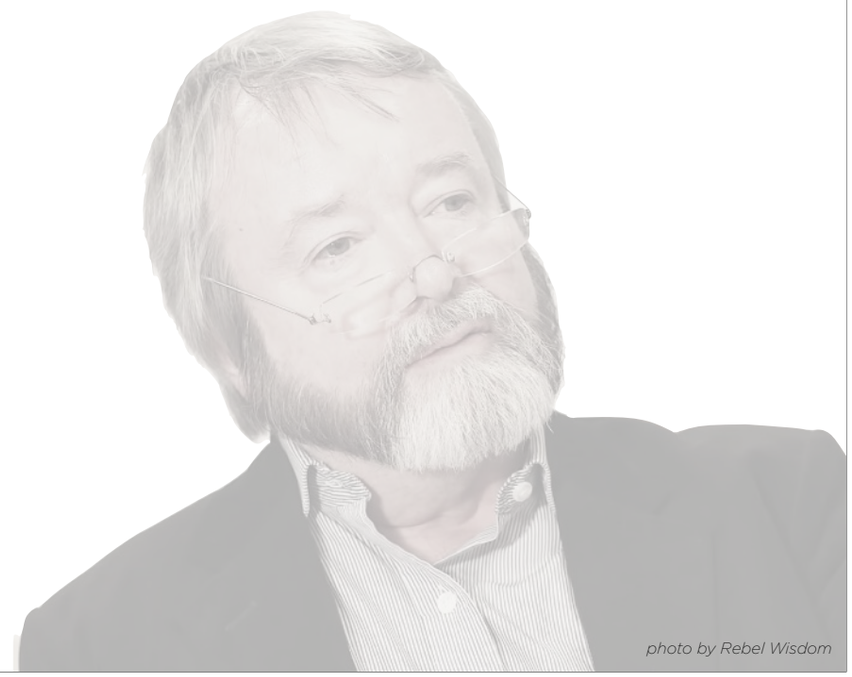


photo by Rebel Wisdom

Iain McGilchrist talks about a very similar idea, but from the perspective of psychiatry and brain science in The Master and His Emissary.

Same mistake is being made in every industry,
In society as a whole...

The first part of the book...
Forget everything you think you know about left and right-brain.
The details are much more subtle than popular reports

Your left and right brain don't **do different things, and it's not really accurate to think that artists are more right brained.**
Both sides share an enormous amount of the load.

But they are divided, and the division is important.
What's important is **how they** work**.**

Left Brain	Right Brain
Different "ways of attending"	
Focused attention	**Peripheral attention**
Division (Boundaries)	**Integration & Connections**
Narrow concerns (Parts)	**Holistic concerns (Gestalt)**
Symbols, Maps & Categories (Generalization)	Individual examples (facial recognition)
Abstraction	Context
Impersonal	Personal (empathy)

When I feel pain, that's everywhere... but when I experience your pain is in my right hemisphere.

Left Brain	Right Brain
"I think therefore I am" (mind disembodied)	The body is the self.
Only one correct answer (Certainty)	Many right answers. Even contradictory. (Ambiguity)
Rationality	Reason (Intuition, "Knowing")

Schizophrenics are detached from their bodies in interesting ways. Some are convinced they are actually copy machines

Or will consider cutting open their wrists to see if they contain engine oil. This is left brain focus gone amok.

The right brain's processes are tightly coupled with the senses. The body is the self... there is no abstract "me" separate from my body from a right brain perspective.

Left Brain Rationality (Logos)

"Although constitutive for science and much of philosophy, because of its being based on argumentation and the provision of proof,

.. it cannot ground itself according to its own principles of proof and argumentation.

The value of rationality, as well as whatever premises it may start from, has to be intuited: neither can be derived from rationality itself.

All rationality can do is to provide internal consistency once the system is up and running."

— *Iain McGilchrist*

In other words, rationality (Left) needs to be subservient to reason (Right).
This is turtles all the way down, St. Thomas Quinas, prove the existence of God shit here. Proof isn't enough... at some point, arguments ultimately rest on something like faith.

Why he calls the Right Brain the master and the left brain the emissary.

Now... isn't this weird... this is the same problem that functional programming has.

I shouldn't pick on functional programming too much, because the same left-brain bias infects **Object Oriented programming**.

Our brain sees the whole world as made up **things**, material, substances.
Maybe a DBA would say... of **entities**. An OO programmer: **objects.**
But let's use the common term: **things**.

These things are **separate** from their environments. Our left brain wants to focus on the part, and pays attention to boundaries.

And these **things** are being *acted upon* by happenings or processes.
Maybe a DBA: **transaction**. A programmer: an **operation**.
But let's use the more common term: **events**.

So we have **things** and **events**.

Broad definition. Can have attributes.
Can potentially have a place.

Difference: time dimension. Things are **persistent in time**.

Clearly, a rock is a **thing**. I can pick it up and take it to another place, and it can be in another time.

You can come ask me tomorrow, where is that rock, now?

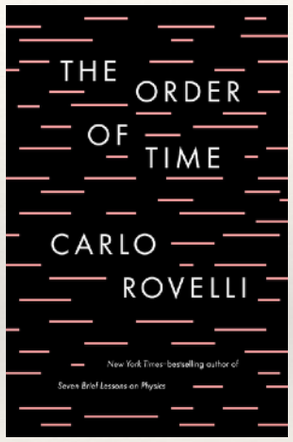
Clearly a **kiss** is an **event.**
Despite existing in a specific place, and despite the ability to last for a while, it is not persistent in time.

We know a kiss is an event because:

It would be **nonsense to ask where** a kiss will be tomorrow.

This is the way our left brain has us convinced the world is.

On closer inspection, though, this proves to be only an illusion.



"The hardest stone... is in reality a complex vibration of quantum fields, a momentary interaction of forces, a process that for a brief moment manages to keep its shape, to hold itself in equilibrium before disintegrating again into dust..."

— Carlo Rovelli

In the words of physicist Carlo Rovelli (read).

Hold this example of the rock and stone in your head for a moment, and think about your the User or Customer entity in your system. How persistent in time is that concept? I suspect if you're like me, then you've been continuously surprised at how often this concept, which seems like the simplest one of all... changes. Is your User really an entity? How persistent is that idea over time? Might it be better, at least conceptually, to realize that a User is a fluid thing, a temporary conglomeration of a Registration, a CohortIdentification, and a ProfileUpdate... maybe those concepts are closer to the reality.

The Struggle Is Real

photo by Rebel Wisdom

Iain McGilchrist says:
- Society has been drifting leftward for 500 years

Other times in history were more in balance, or were even right-brain dominant.

This leftward drift is cultural... it's happening despite active attempts at correction.

Agile... transformation?

Right-Brained Manifesto

Interactions between people, not tools

Actual software, not abstractions (docs) of it

Holistic collaboration with the customer

Left-Brained Trends

Trello · Jira Software · PivotalTracker · Airtable · asana

All new abstractions: Backlog, User Stories, Roadmap, Sprint

Agile training for the customer.
PMs, EMs, and a dozen other job titles.
Left brain ❤️ acronyms.


Medium
Equality
Become a member
Sign in
Get started
Why Don't We Just Call Agile What It Is: Feminist
Hanna Thomas
Follow
Sep 3 · 7 min read ★
The men who wrote the manifesto are unclear on whether they invited any women. Even so, the thing I notice from both the manifesto, the accompanying principles, and the fact that these 17 men call themselves 'organizational anarchists' is that what they came up with is inherently subversive, anti-authoritarian, and feminist. There is an emphasis on self-organising, collaboration, experimentation, welcoming change, and building high-trust and supportive relationships.

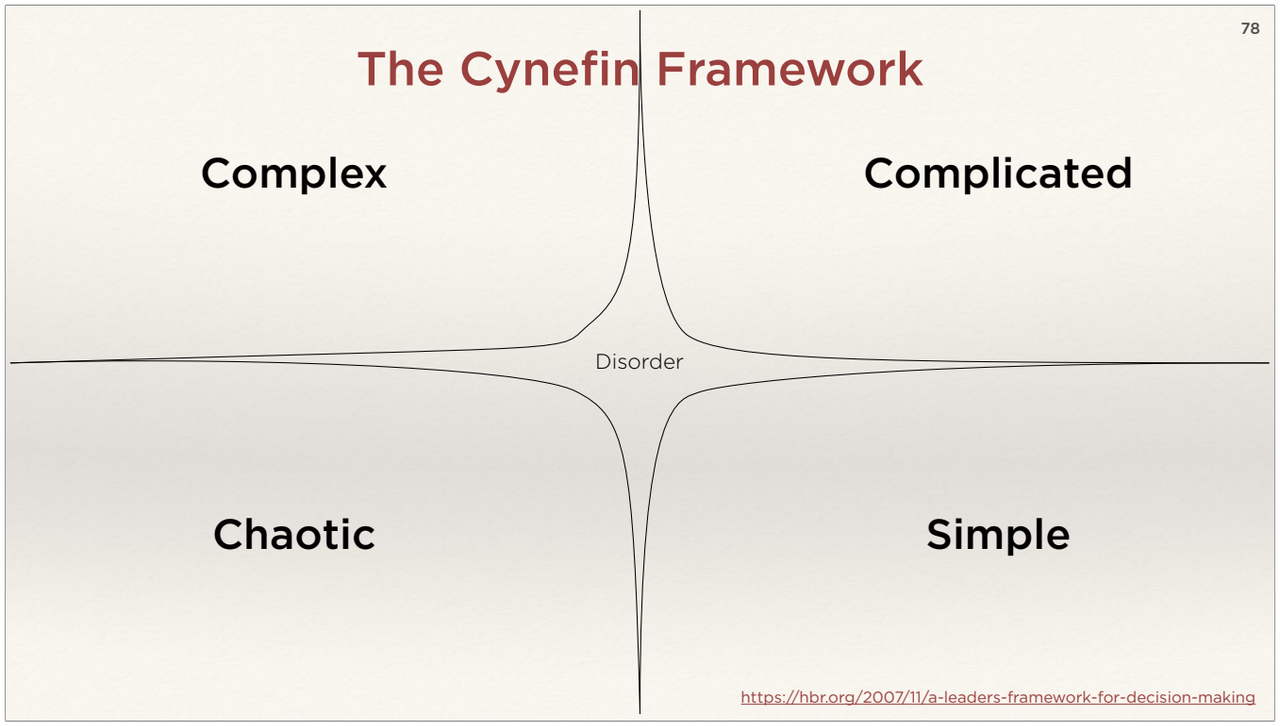
Devops... transformation?

Right Brain Origins	Left Brain Reality
Developers need more holistic awareness of environment	Dedicated platform engineers build devops pipelines (details abstracted away for ordinary devs)
Integrate ops and dev teams	Outsourced DevOps
Team needs shared ownership of total quality	Site Reliability Engineers: Narrow focus on specific metrics.

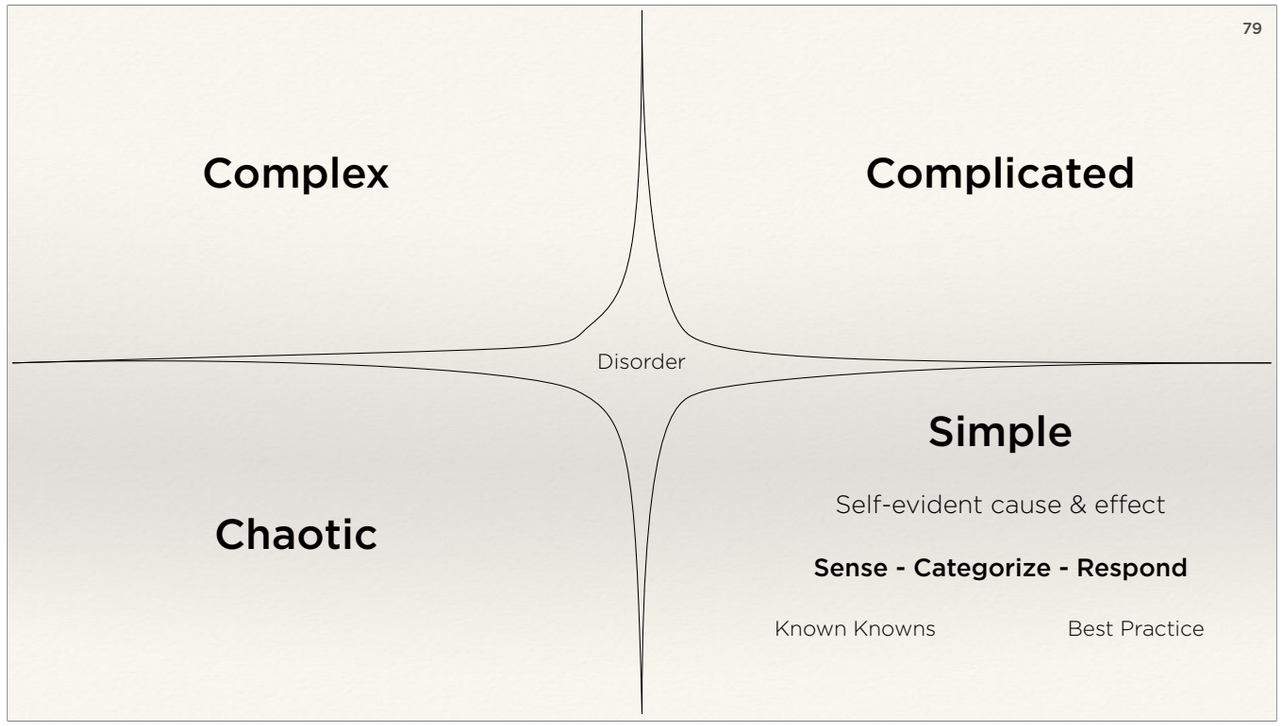
So What do We Do About It?

photo by Rebel Wisdom

The Cynefin Framework



https://hbr.org/2007/11/a-leaders-framework-for-decision-making

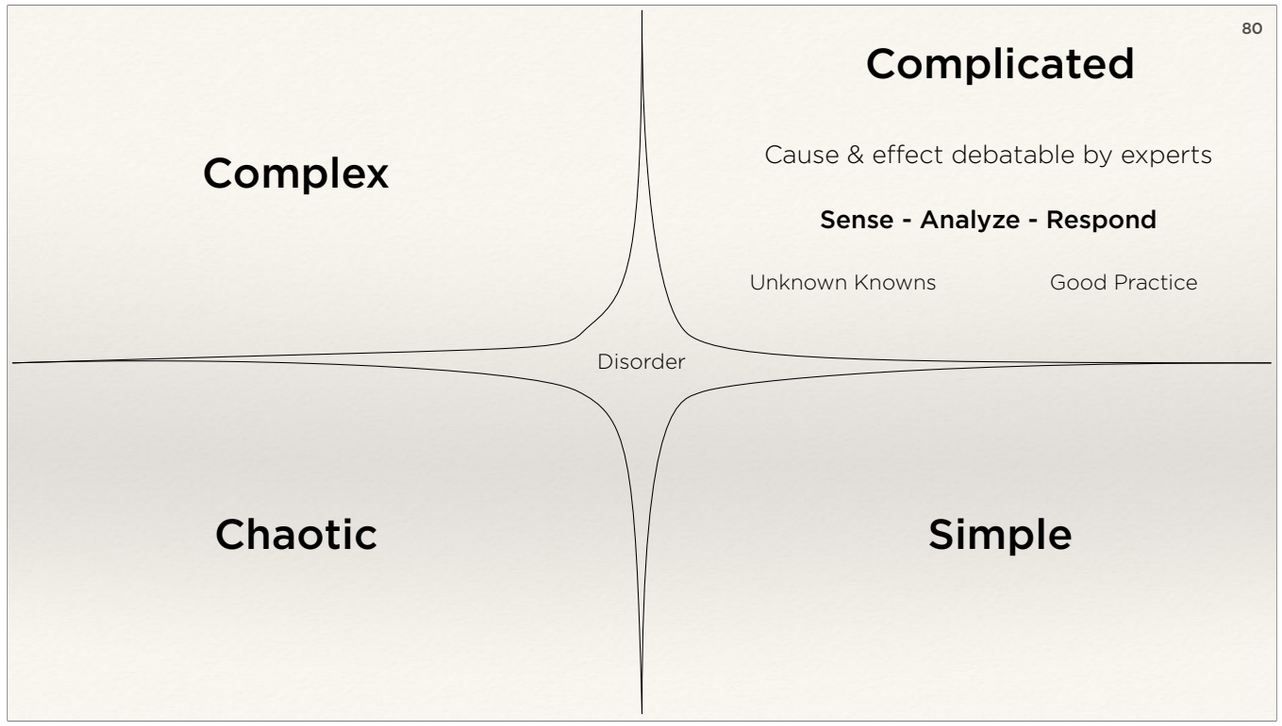


The simple domain is the first of two domains which Snowden says are "ordered", which are domains where there is a relationship between cause and effect.

In the Simple domain, the relationship between cause and effect is self-evident. He now calls this the "Obvious" domain because what do to is obvious.

In terms of how you behave, it's a simple matter of categorizing the scenario, and executing the obvious solution.

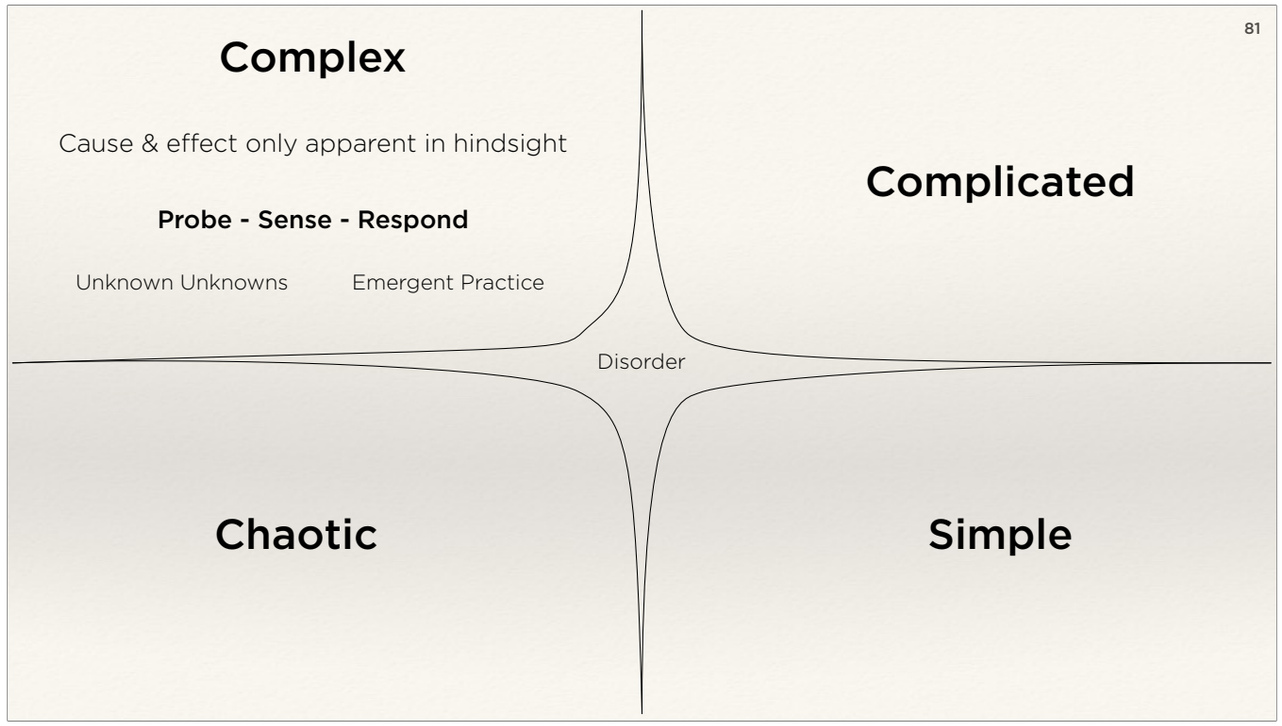
A good example of this domain is making coffee. There's a recipe, but you need to follow it closely. You can dramatically mis-measure the beans, and grind them poorly and you'll still get coffee. However, there is the idea of Best Practice in the Simple Domain. It's possible to execute perfectly in the Simple Domain.



The Complicated domain is the domain of experts... where we engineers thrive.

Analysis is the key here. There are more moving parts, so a diagnosis will not be possible at all for people with no expertise. Experts might disagree about the diagnosis or the solution. This is why Snowden points out that there are no "best" practices here. It won't be possible to fully understand all the interactions, so everyone will doing their best.

A good example here would be troubleshooting a mechanical noise on your car. To you, it's an unknown known... but you can hire an expert to solve it for you. There might be multiple valid solutions to the problem. There is no one "right" answer... but it will be possible to determine if their approach was reasonable, especially when reviewed by another expert.

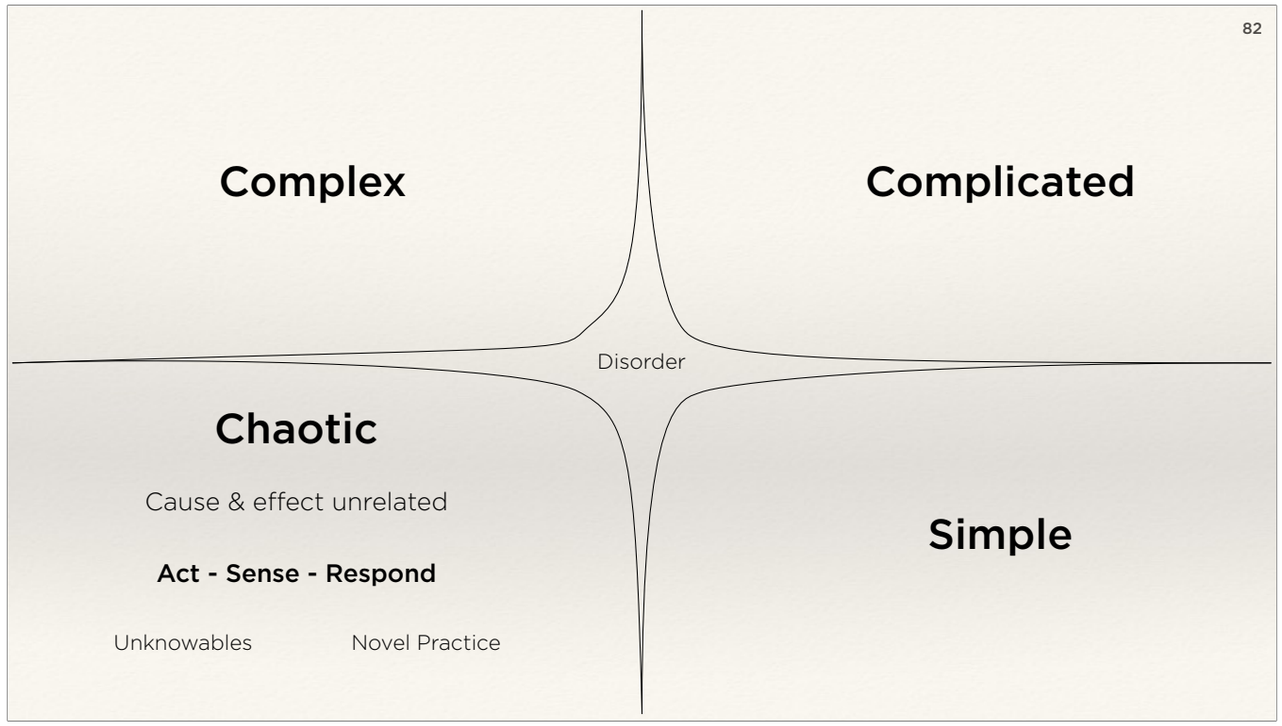


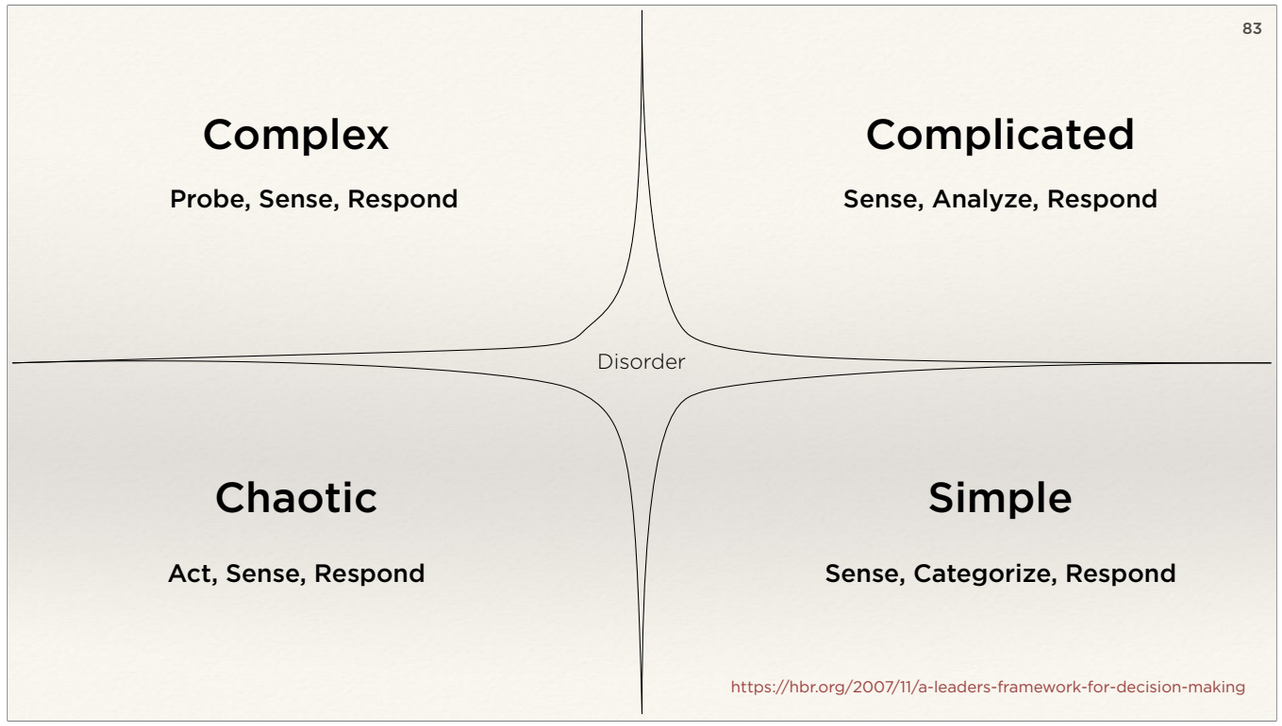
Complexity is the first of the two "unordered" domains. In a complex domain, even experts can't identify make accurate predictions, because there is no a priori relationship between causes and effects.

This is what Donald Rumsfeld was talking about when he was describing Unknown Unknowns.

When you're in this environment, your actions affect the system. So your plan of action is to take small actions and treat them as experiments. You're looking for some wisdom of practice to emerge from your experiments.

Because it's possible that you could trigger feedback loops in this environment, you should identify ahead of time how you will amplify or dampen the effects of your probes before you start them.


Complex
Complicated
Disorder
Chaotic
Cause & effect unrelated
Act - Sense - Respond
Unknowables
Novel Practice
Simple



Finally we have the center space... labeled Disorder. This isn't really a separate dog main... it's just the state of now knowing which domain you are in. This is the most common situation.

According to Snowden, we will all have a bias based on our preferred mode of action.

Bureacrats will be biased towards categorization. They'll assume everything is simple. Which form do I need?
People who like being in charge may have a bias towards action. They may assume or try to create a chaos where none exists. Not sure if you can think of an example of that.
Politicians may be biased towards the complex. They might like to bring lots of dissenting opinions into a room and then make a call.
It should be obvious, but I think you and I have an extreme bias towards assuming everything is complicated... this upper-right quadrant is home for me.

But our real estate startup was clearly in a Complex environmment.

What I Do Differently

1. **DRY** is not a virtue.

DRY is not a virtue. It might even be harmful.
Don't teach this idea to junior developers. If you insist on things being DRY to start with, you're on the way to stiff, lifeless, hard to change code.

If you have to teach them something. Fowler's 2 strikes and then you refactor. And even then, 3 is probably a better number.

Not because perfection is the enemy of the good.
Because if you achieve it, it's inevitably worse in the long run.

It's not how good it is that matters, it's how easy it is to change.

Realize that people are inevitably part of the software process. Hard-coding something is fine if it's easy to change.

3.Get **smaller** whiteboards.

Get smaller whiteboards. Think smaller. Stop doing projects, or at least make them shorter.
And build smaller things.
The larger your scope, the larger your error.
When you build software you are translating some
observation about the world and encoding it.
But our observations are flawed, and the encoding is lossy.
And the world keeps changing.

4. Design to **functions**, not **objects**

(a.k.a. data)

Functions in the biological sense here, not the mathematical one.

I try to think more about the interactions and behaviors of things, and less about their platonic identities.

Eventstorming.

5. Feel.
Be in the body.
Have emotions.

6. Don't hire copies of yourself.

7.It's wrong to be right

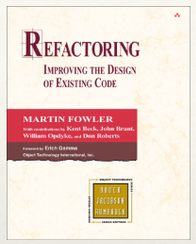
(and it's right to be wrong)

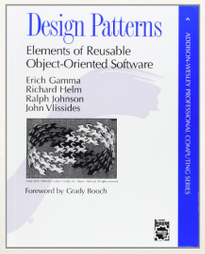
It's wrong to be right.
If you're right it's because you have knowledge and experience.
Knowledge and experience come from past, so they're safe.
Being wrong means you're taking risks.
If you're not failing to meet 50% of your forecasts, you're not taking enough risks.

"The intuitive mind is a sacred gift, and the rational mind is a faithful servant.

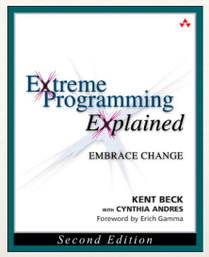
We have created a society that honors the servant, but has forgotten the gift."

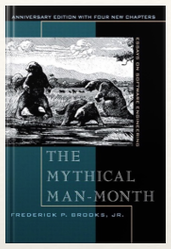
— attributed to Einstein (but he didn't say it)

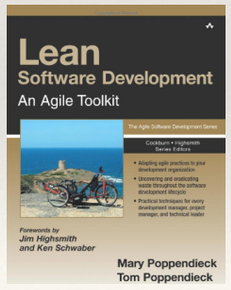

REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER

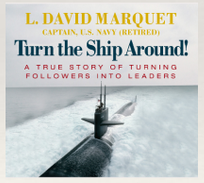

Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch

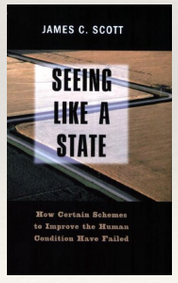

A PHILOSOPHY OF SOFTWARE DESIGN

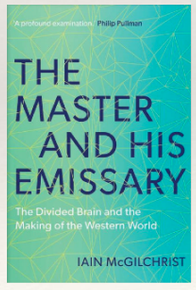

Extreme Programming Explained
EMBRACE CHANGE
KENT BECK
CYNTHIA ANDRES
Foreword by Erich Gamma
Second Edition

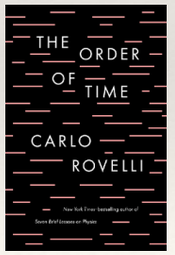

THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.


Lean
Software Development
An Agile Toolkit
Mary Poppendieck
Tom Poppendieck


L. DAVID MARQUET
CAPTAIN, U.S. NAVY (RETIRED)
Turn the Ship Around!
A TRUE STORY OF TURNING FOLLOWERS INTO LEADERS


JAMES C. SCOTT
SEEING LIKE A STATE
How Certain Schemes to Improve the Human Condition Have Failed


THE MASTER AND HIS EMISSARY
The Divided Brain and the Making of the Western World
IAIN McGILCHRIST


THE ORDER OF TIME
CARLO ROVELLI