



An Introduction to Network Complexity

Russ White/rule11.us



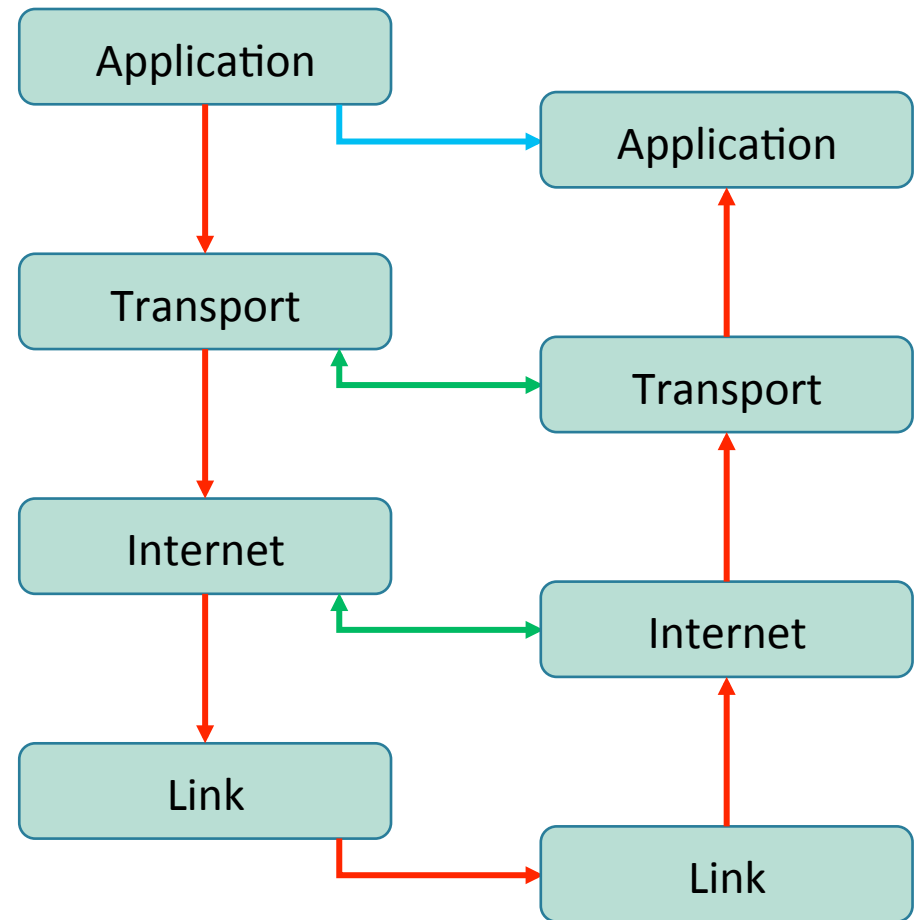
Networks are complex...

- And getting more complex all the time
- But what is complexity, really?
- Can we put a network on a scale that measures complexity?



Defining Complexity

- Move information between applications
- Carry information down the stack, across the wire, back up the stack
 - Split responsibilities up
 - Control amount of information managed by any given layer/application
- Carry information laterally between stack elements



Complexity is...

- Anything I don't understand
- It's true that *hard to understand* is often a stand-in for complexity, but...
- There are many things I don't understand that aren't complex
- Often understanding is a matter of context rather than complexity
- *So this is only part of the answer*

Complexity is...

- Anything with a lot of parts
- It's true that most complex things have a lot of parts, but...
 - Mandelbrot sets have a lot of parts
 - But they aren't necessarily complex
- *So this is only part of the answer*



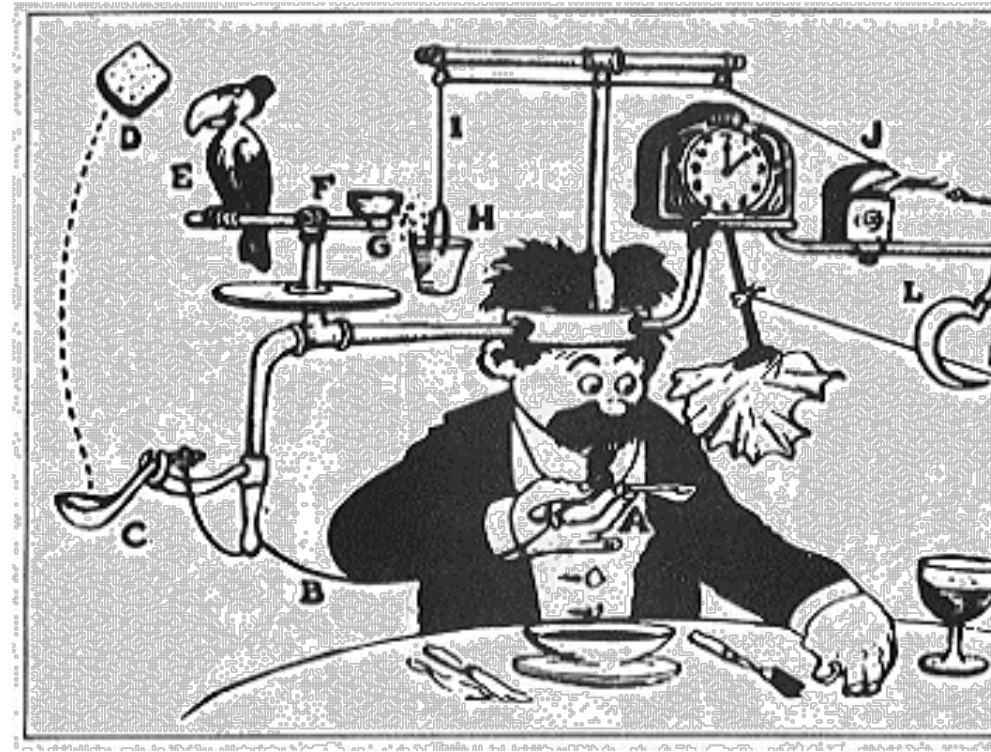
Complexity is...

- Anything that produces unintended consequences
- It is true that complexity prevents us from seeing every possible result of an action, but...
 - Sometimes we just don't look hard enough
 - Sometimes we just can't foresee every possible situation
- *So this is only part of the answer*



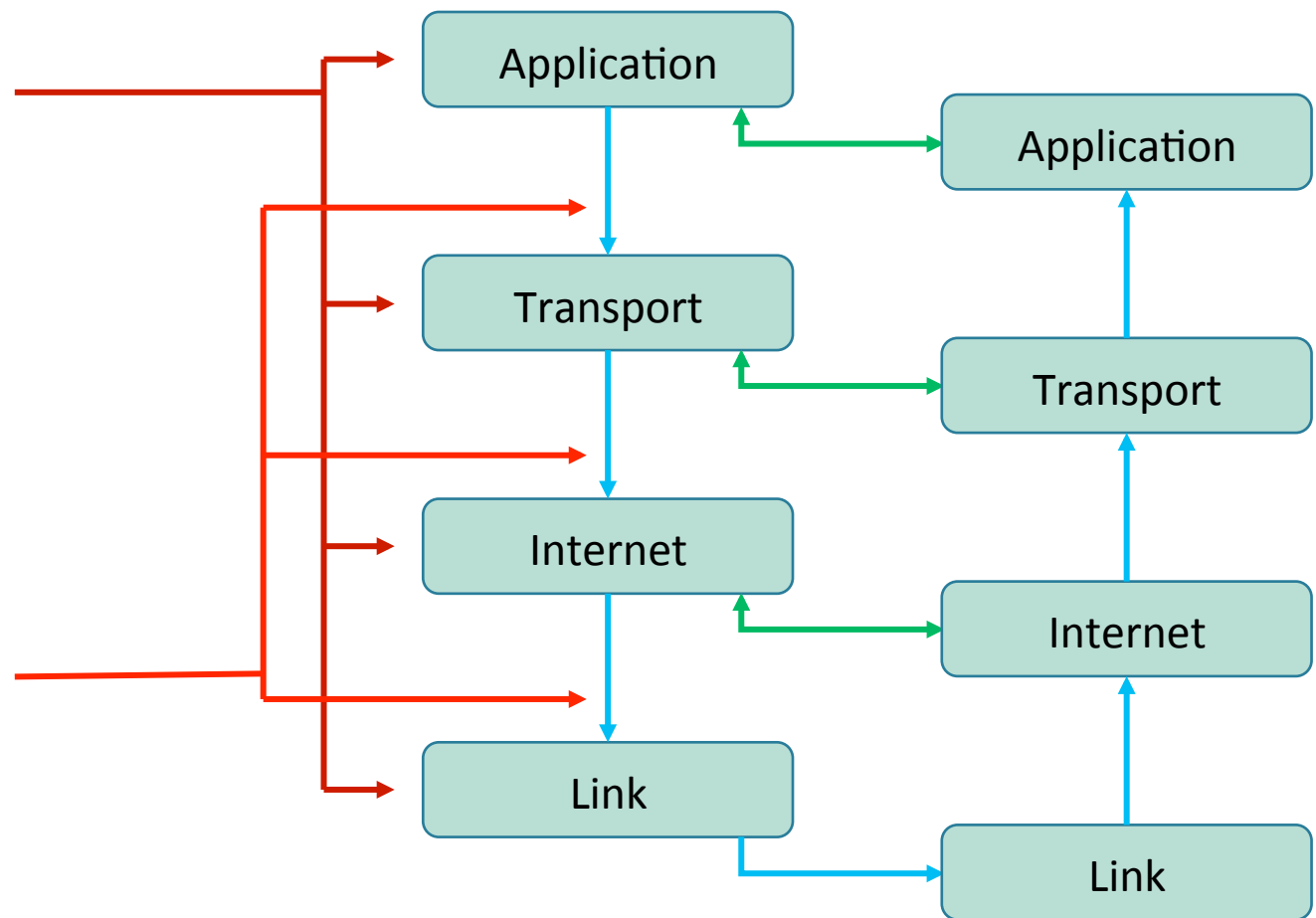
Complexity is...

- Anything that has more parts than required to solve a specific problem
- Define “too many...”
 - For this you need to decide what “the right number” is
 - And you’re not likely to get an single answer on this one...
- *So this is only part of the answer*



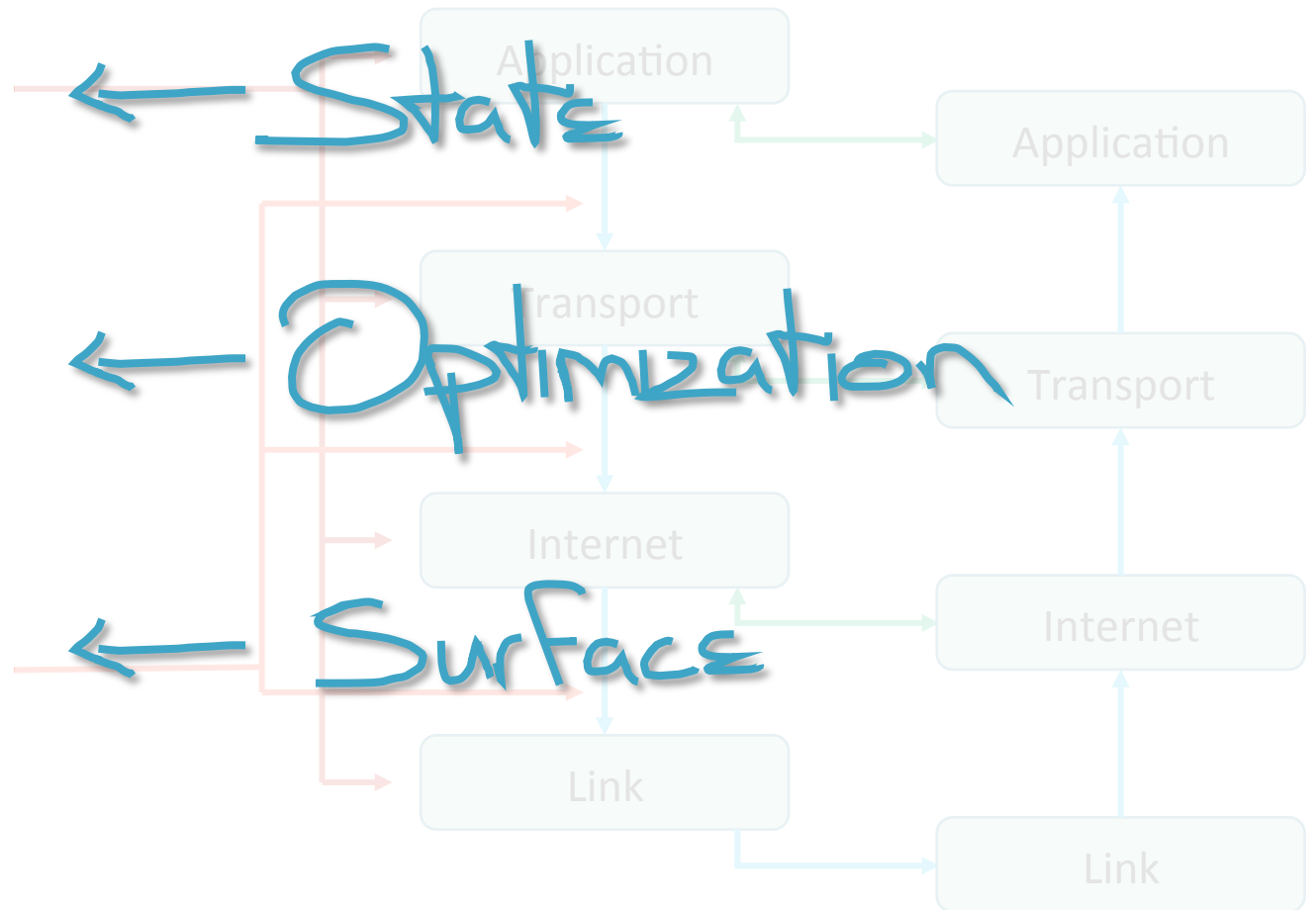
Defining Complexity

- What state is contained here?
- How much?
- How fast does it change?
- What am I optimizing by splitting this state up?
- What is the goal of breaking this up into multiple subsystems?
- What state is shared here?
- How much does the internal state mix between these two components?



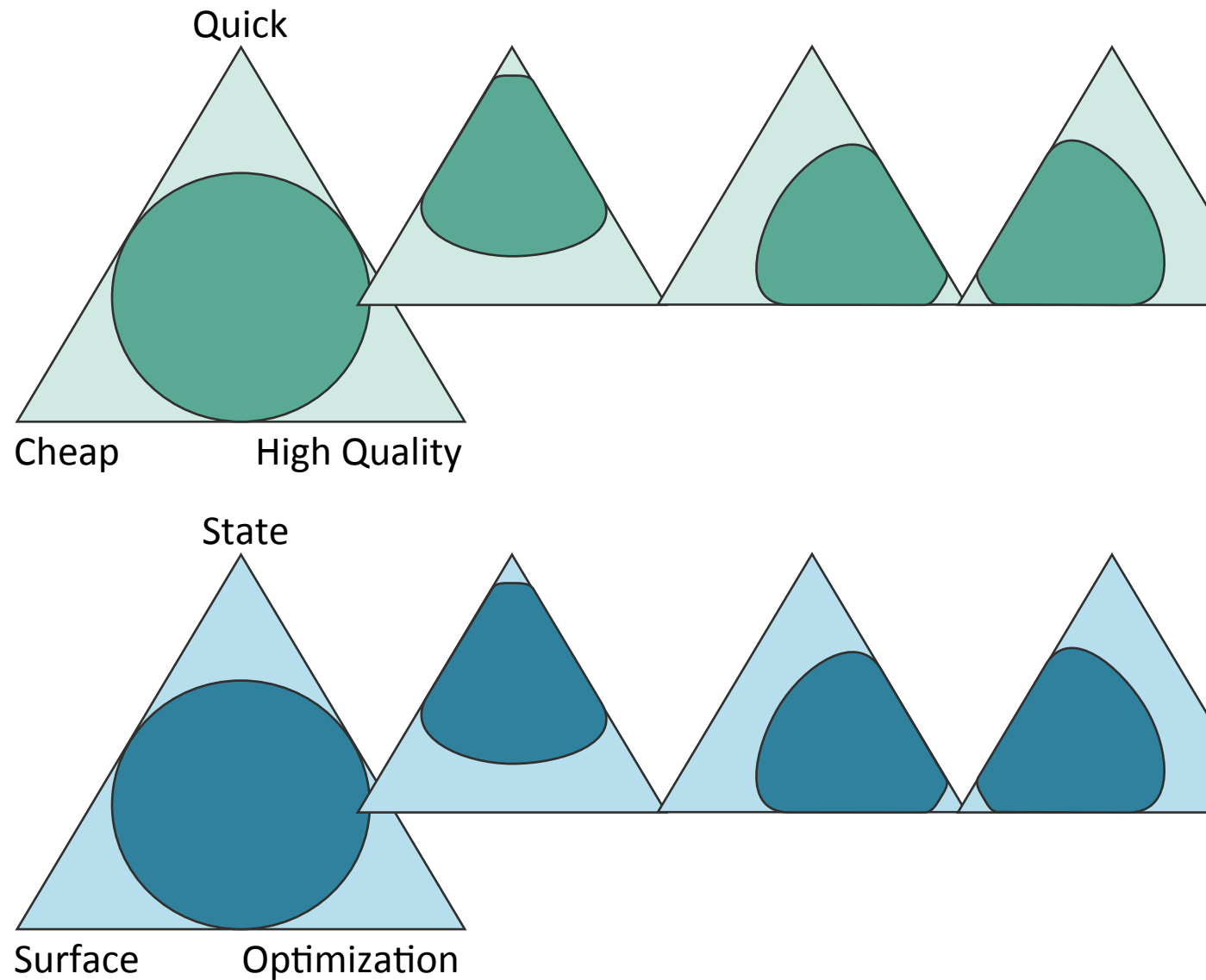
Defining Complexity

- What state is contained here?
- How much?
- How fast does it change?
- What am I optimizing by splitting this state up?
- What is the goal of breaking this up into multiple subsystems?
- What state is shared here?
- How much does the internal state mix between these two components?



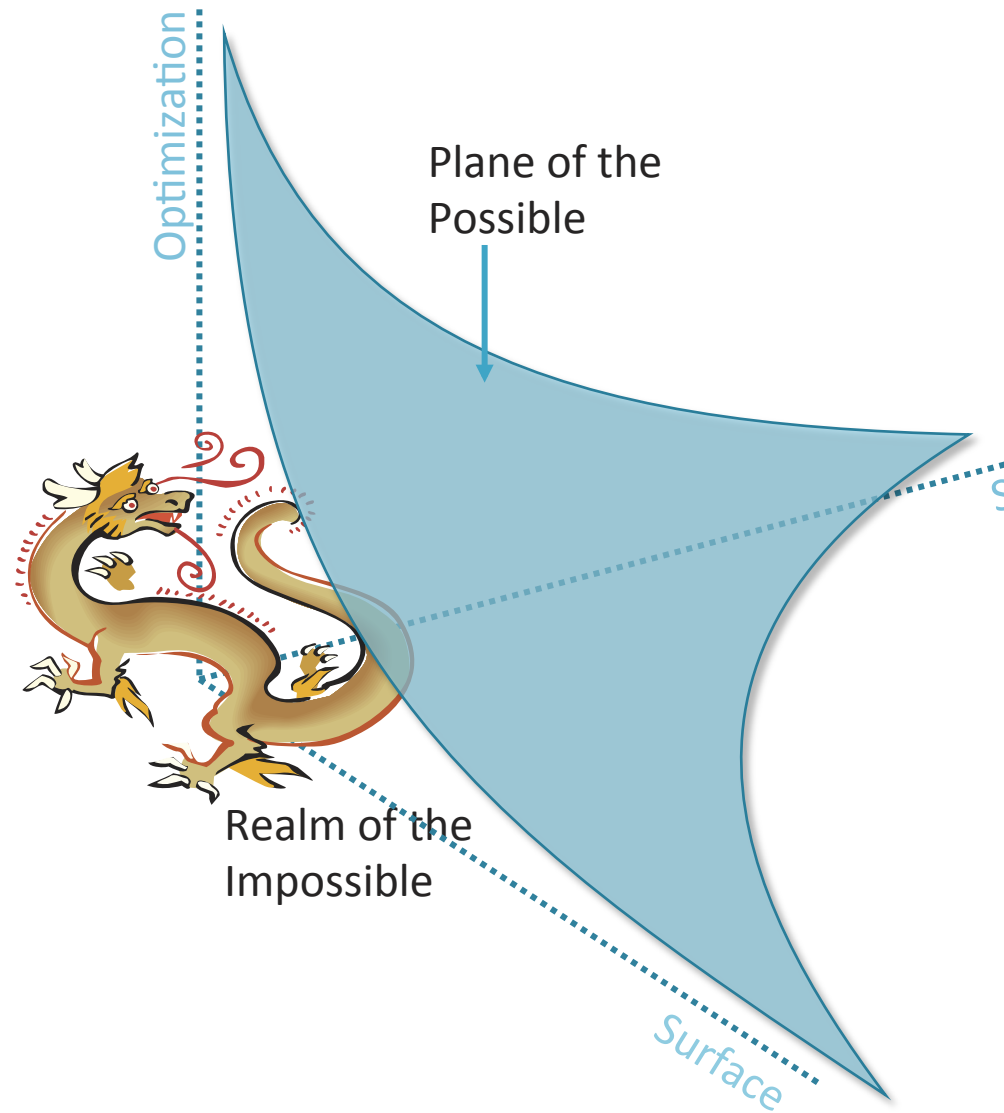
Defining Complexity

- Three way choices are common in the real world
- Network complexity is another form of this sort of three way choice



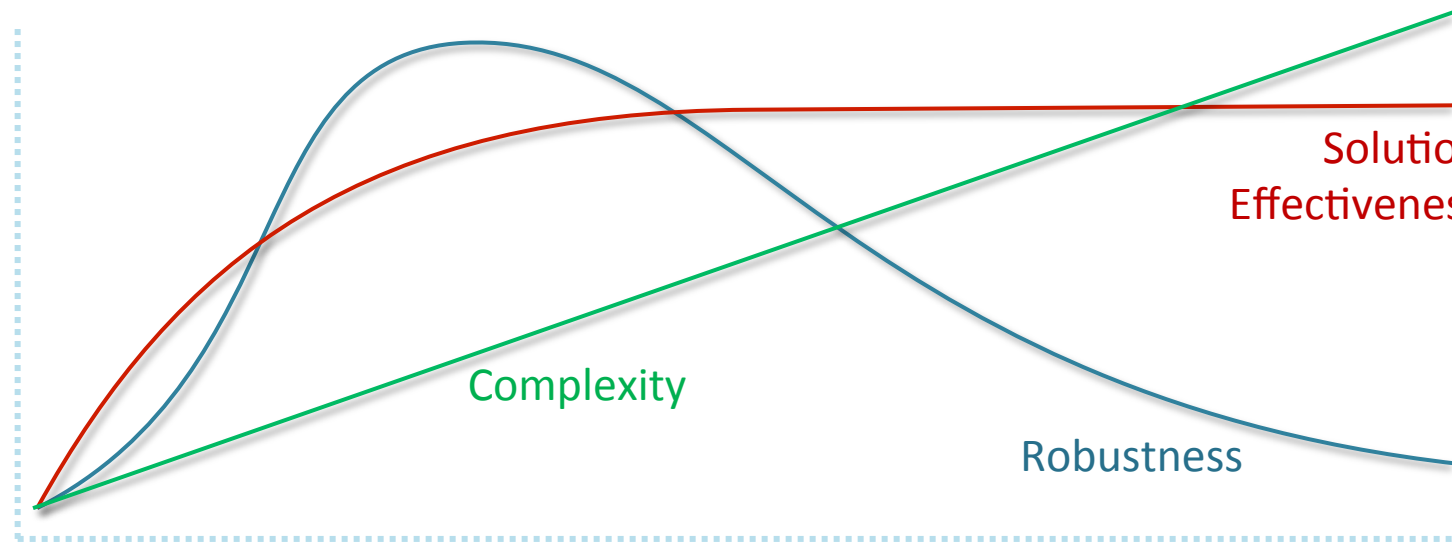
You can't "solve" complexity

- Why not just make things simple?
- The "plane of the possible" is just the way reality is built



You can't "solve" complexity

- Why not just make things simple?
- Because complexity is required to solve real world problems

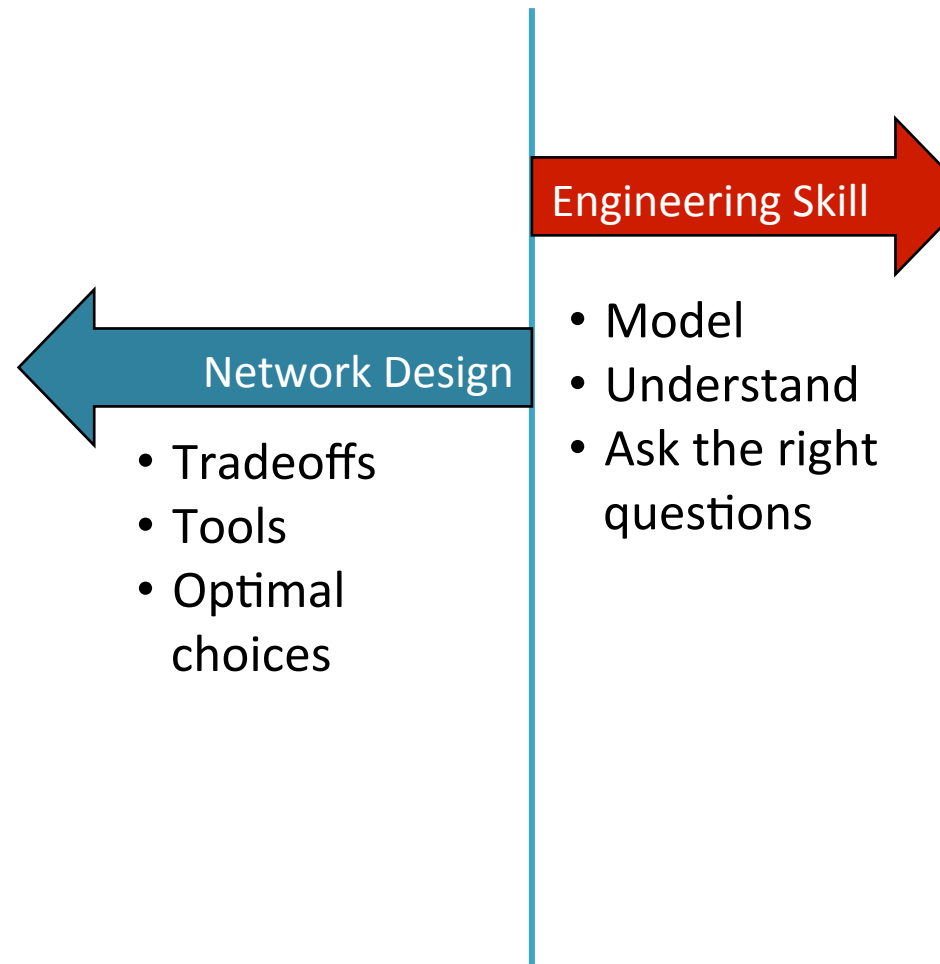


*In our view, however, complexity is most succinctly discussed in terms of functionality and its robustness. Specifically, we argue that **complexity in highly organized systems arises primarily from design strategies intended to create robustness to uncertainty** in their environment and component parts.*

Alderson, D. and J. Doyle, "Contrasting Views of Complexity and Their Implications For Network-Centric Infrastructures", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 40, NO. 4, JULY 2010

But you can manage it...

- There are two directions in which we can manage complexity
- Towards the design
- In our thinking as engineers

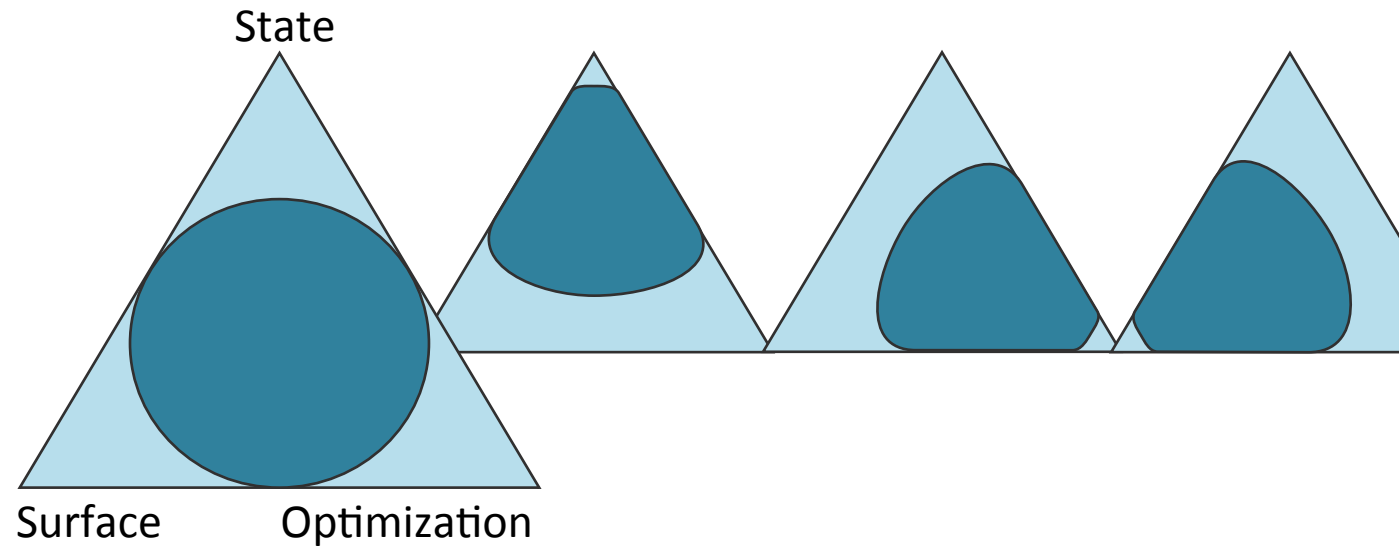




Complexity in Design

Complexity in design

- Complexity is a tradeoff
- Decreasing complexity in one part of the system will (almost) always increase complexity elsewhere



It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it.

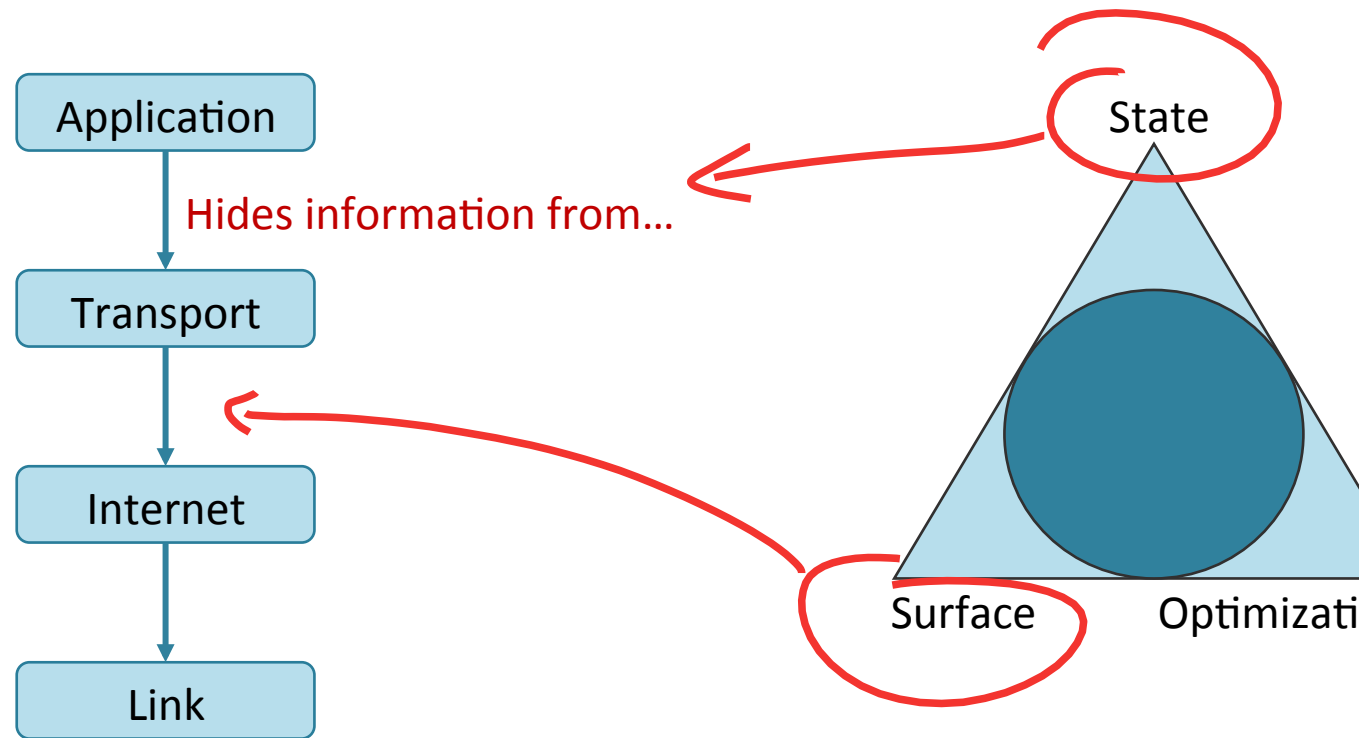
...

It is always possible to add another level of indirection.

RFC1925

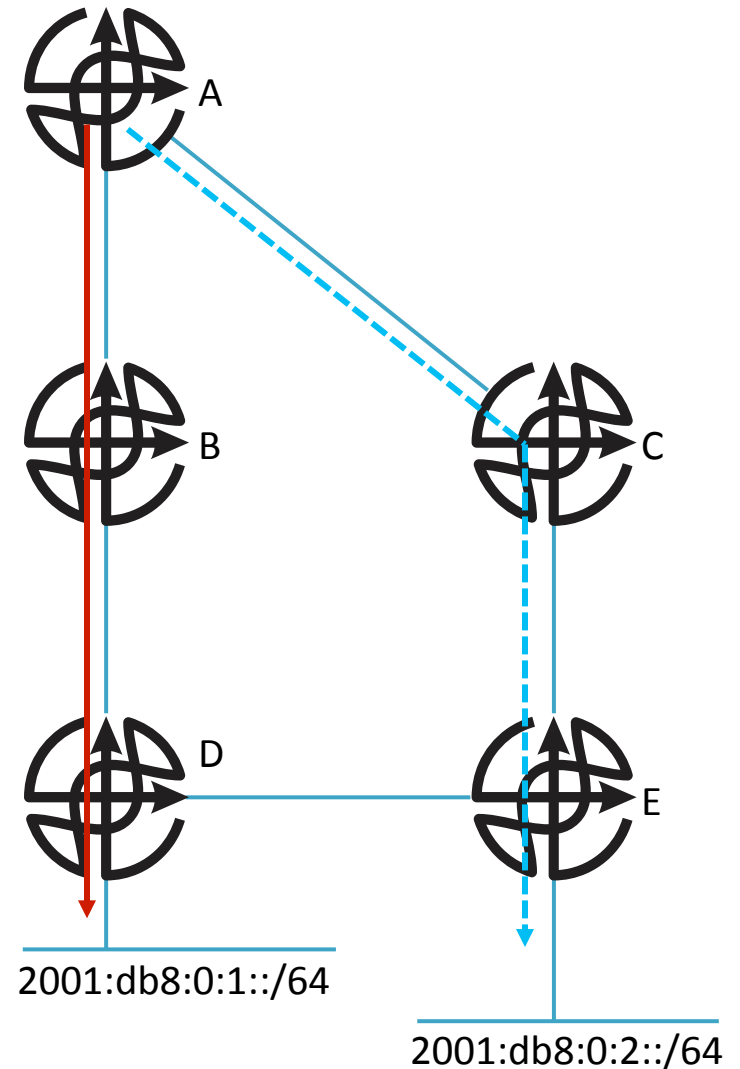
Hiding Information

- We layer protocols
- Manages the amount and speed of state at each layer
- Adds interaction surfaces between the layers, across the network, etc.



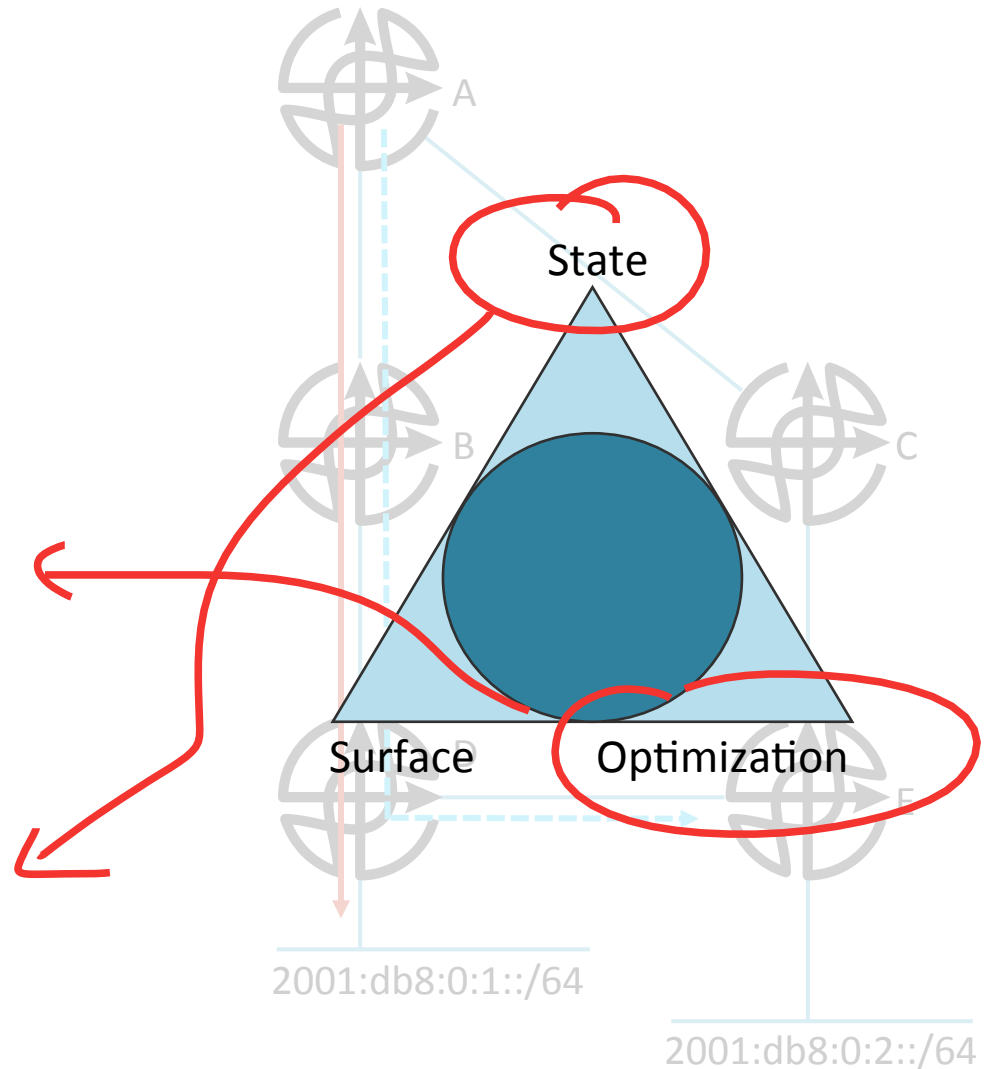
Hiding Information

- If B and C do not aggregate
 - A will have the optimal route to reach both $2001:db8:0:1::/64$ and $2001:db8:0:2::/64$
- But...
 - A will have more routes in its local routing table
 - A will receive topology state changes for all the links and nodes behind B and C
 - So more routes, more state change visibility, more complexity



Hiding Information

- Assume A aggregates to 2001:db8::/61
- Increases stretch
 - A will choose a suboptimal route to either 2001:db8:0:1::/64 or 2001:db8:0:2::/64
- Reduces state
 - A has fewer routes in its local table
 - A deals with less state change over time



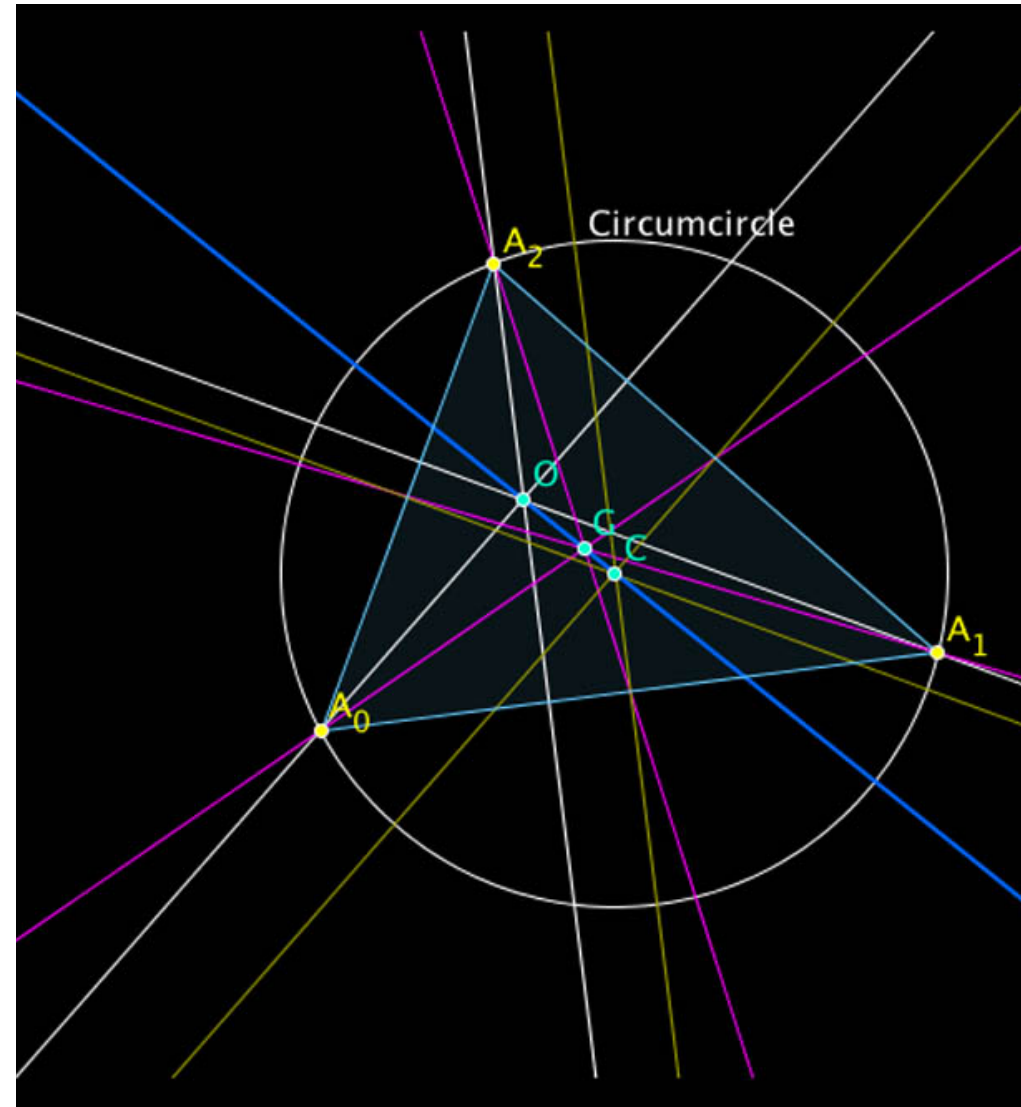
Automation

- More complexity can be managed with better tools
- Beware of the network effect
 - If your only tool is a hammer...
- Figure in the cost of the tool
 - Nail guns are harder to maintain than hammers
 - Sonic screwdrivers are notorious for breaking at just the wrong moment



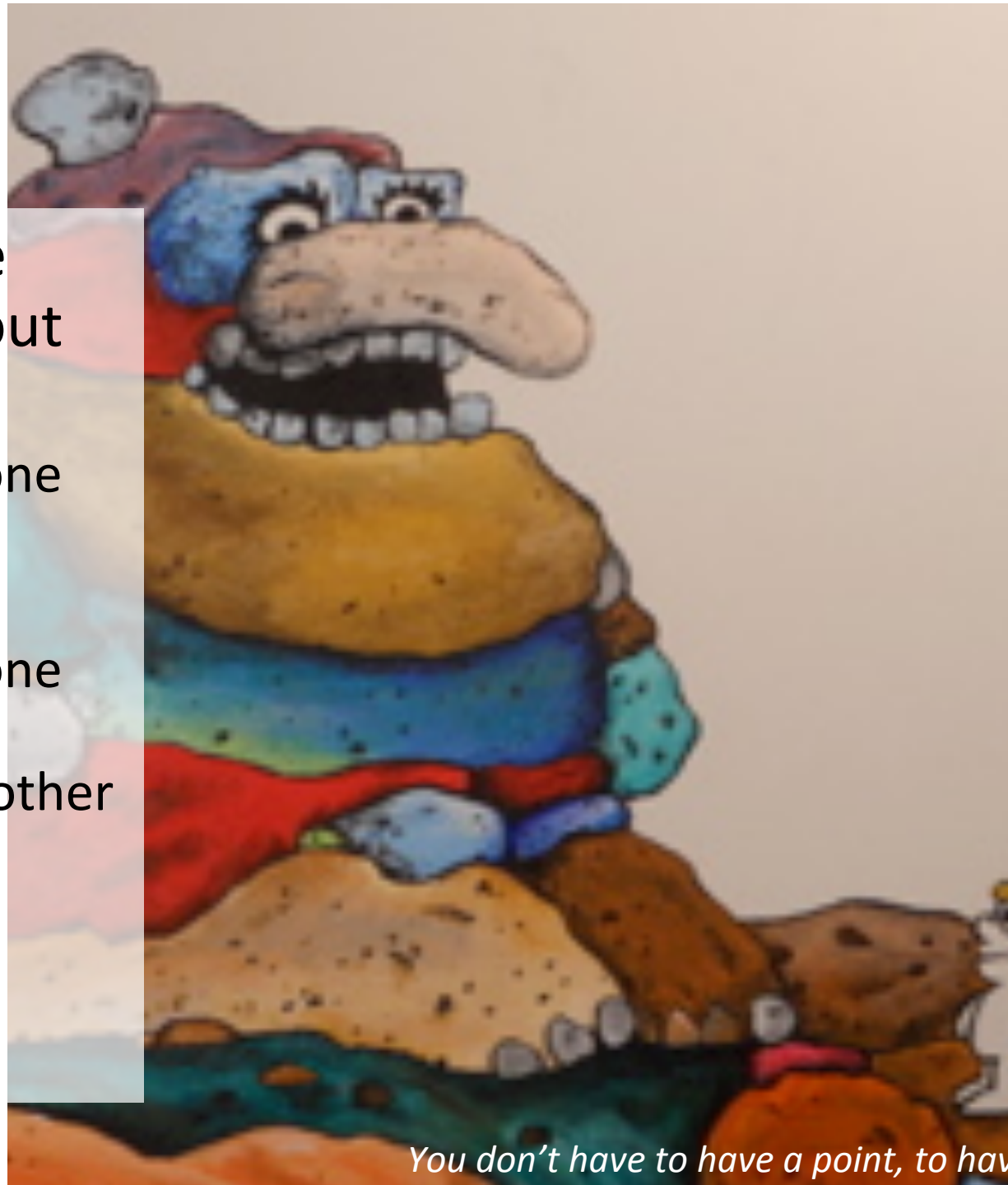
Complexity versus Skill Set

- Anything I don't understand?
- Then I need to understand more!
- Understand complexity
 - Look for the tradeoffs
 - Ask the right questions
- Be a collector of...
 - Models, protocols, theories, algorithms, ideas, etc.
- Ask the right questions...



The Point

- You can never reach some other desirable goal without increasing complexity
 - Decreasing complexity in one place will (nearly) always increase it in another
 - Decreasing complexity in one place will often lead to suboptimal behavior in another
 - Increasing service levels or solving hard problems will almost always increase complexity



You don't have to have a point, to have

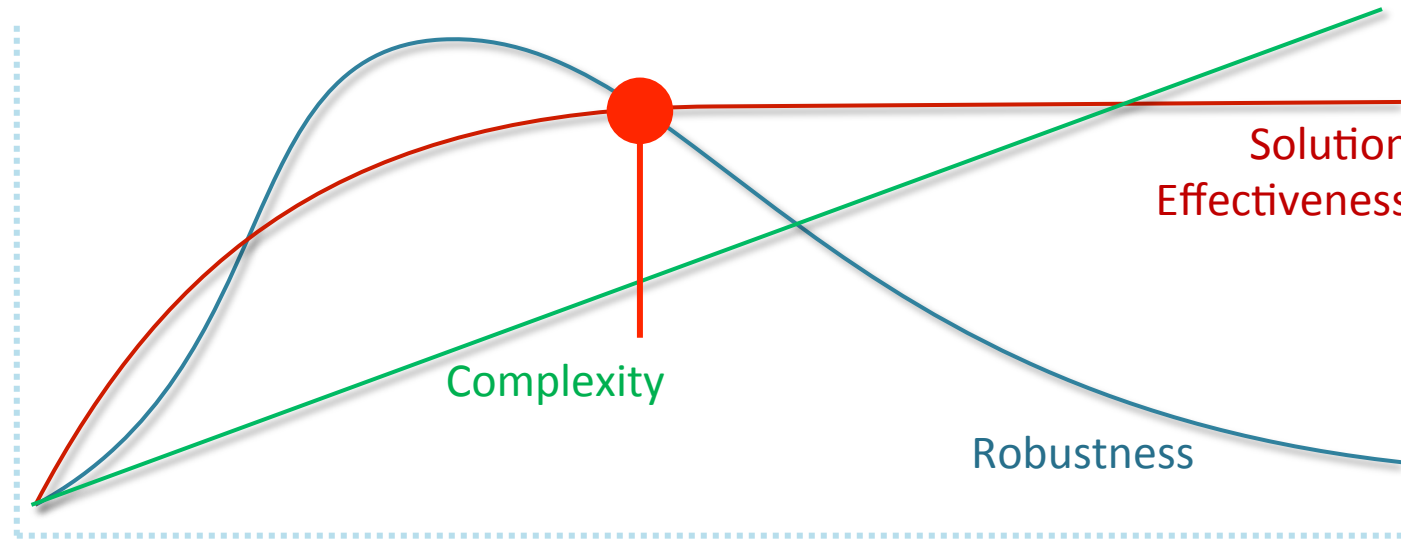
Ask the Right Questions

- Bad questions
 - How complex is this?
 - Will this scale?
- Good questions
 - Where will adding this new thing increase complexity?
 - If I reduce complexity here, where will I increase it?
 - If I reduce complexity here, where will suboptimal behavior show up?
- Complexity at the system level is about *tradeoffs*, not *absolutes*

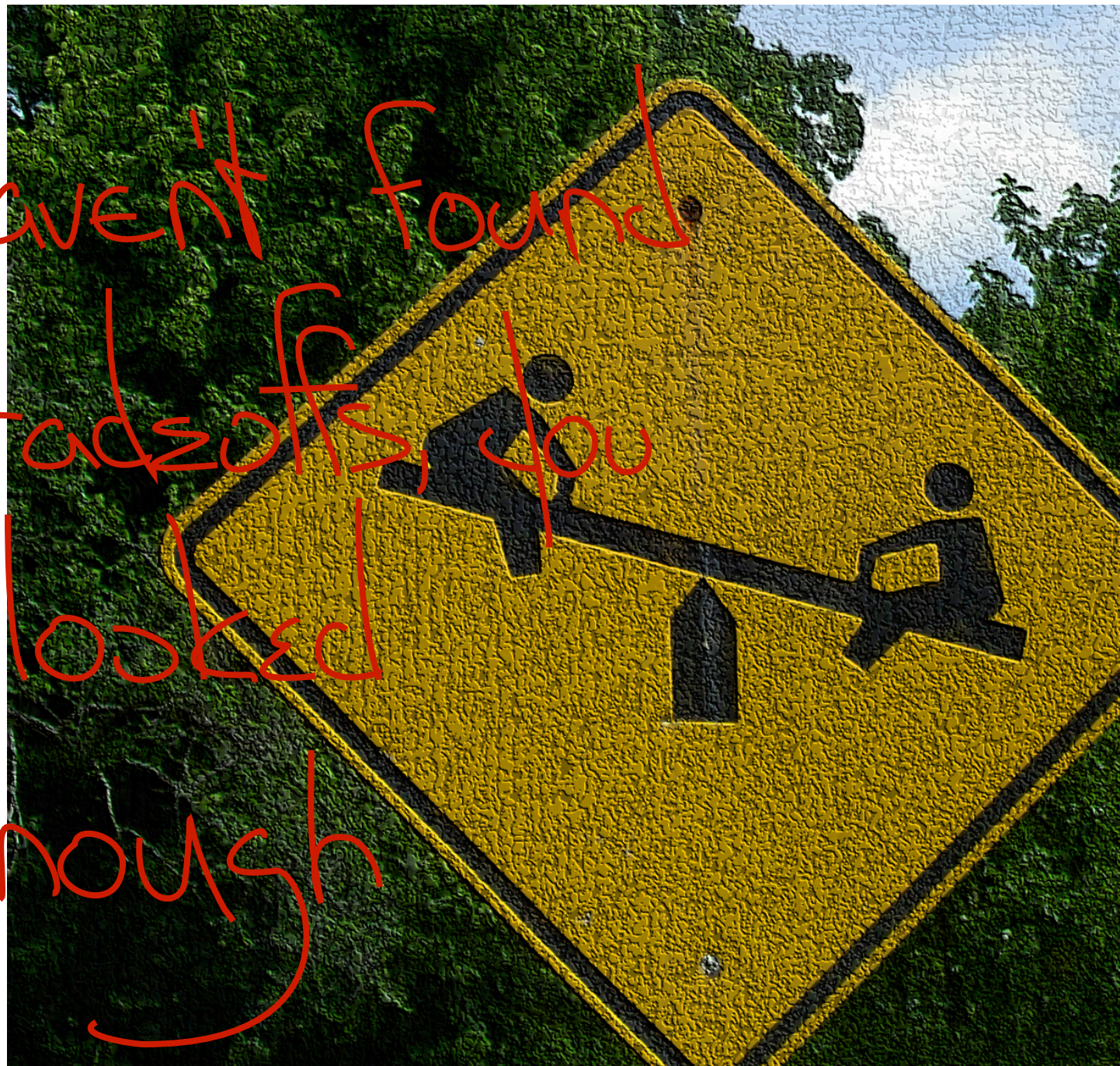


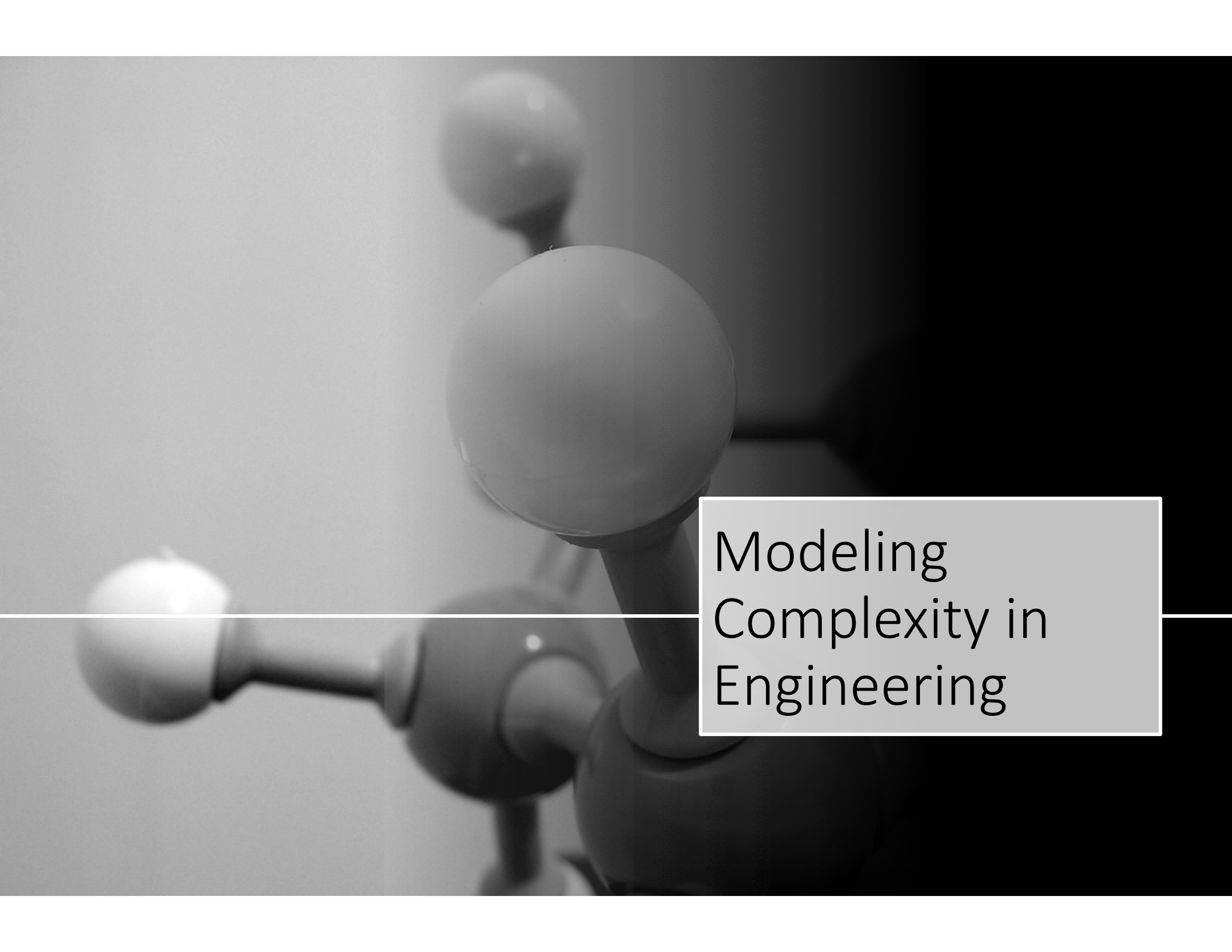
Complexity in design

- Always look for the tradeoff
- *If you don't see the tradeoff, you've not looked hard enough*
- Seek out the point where complexity is lowest, robustness is highest, and the solution isn't really improving any longer...



If you haven't found
the tradeoffs, you
haven't looked
hard enough





Modeling Complexity in Engineering

Metamodel
to understand
a complex system quickly

*Build an abstract model by
asking the right questions*

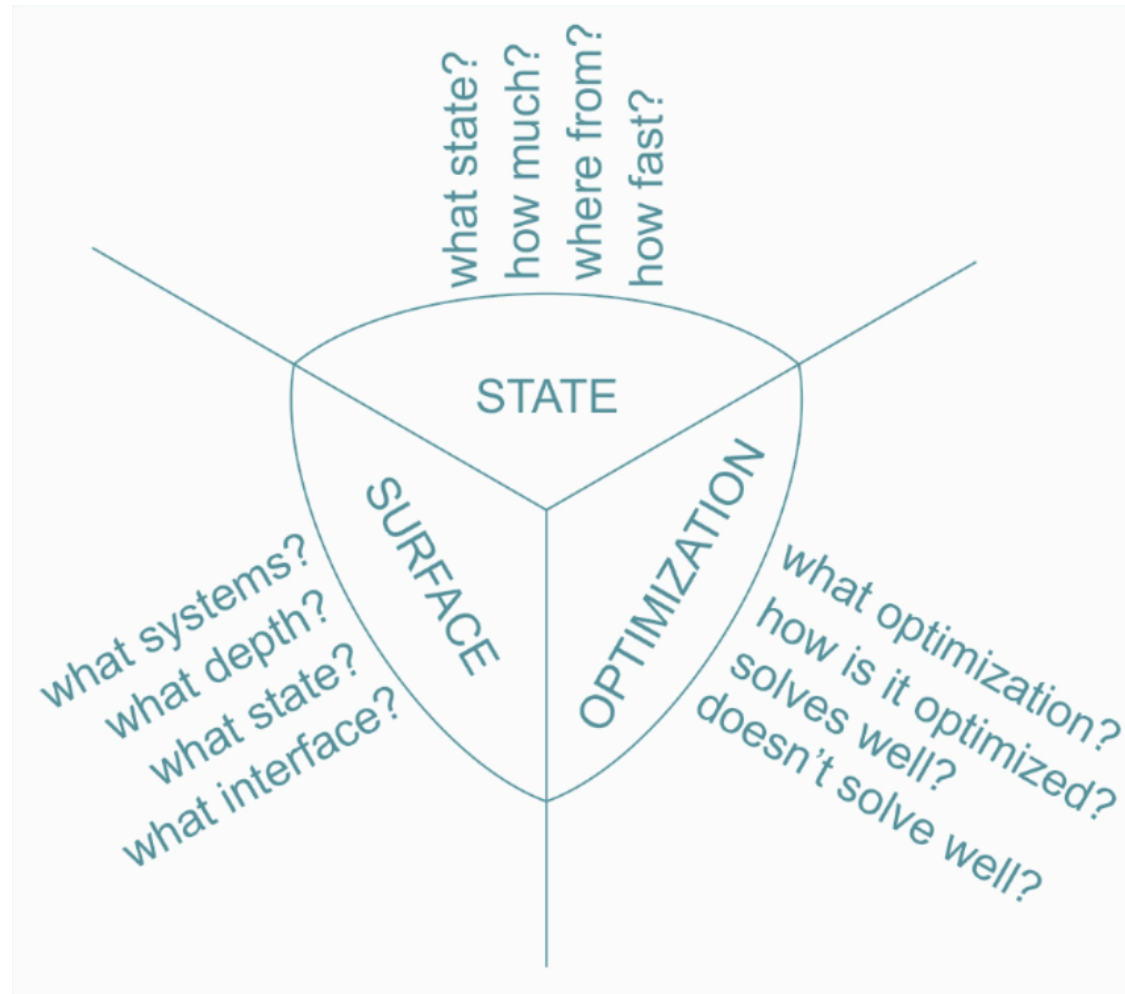


Why?

What problem am I solving?

Narrows my view –
don't boil the ocean

What & How?



This is like?

Has this problem been solved before?

How was it solved?

What are the problems with this solution?

The “Why” Question

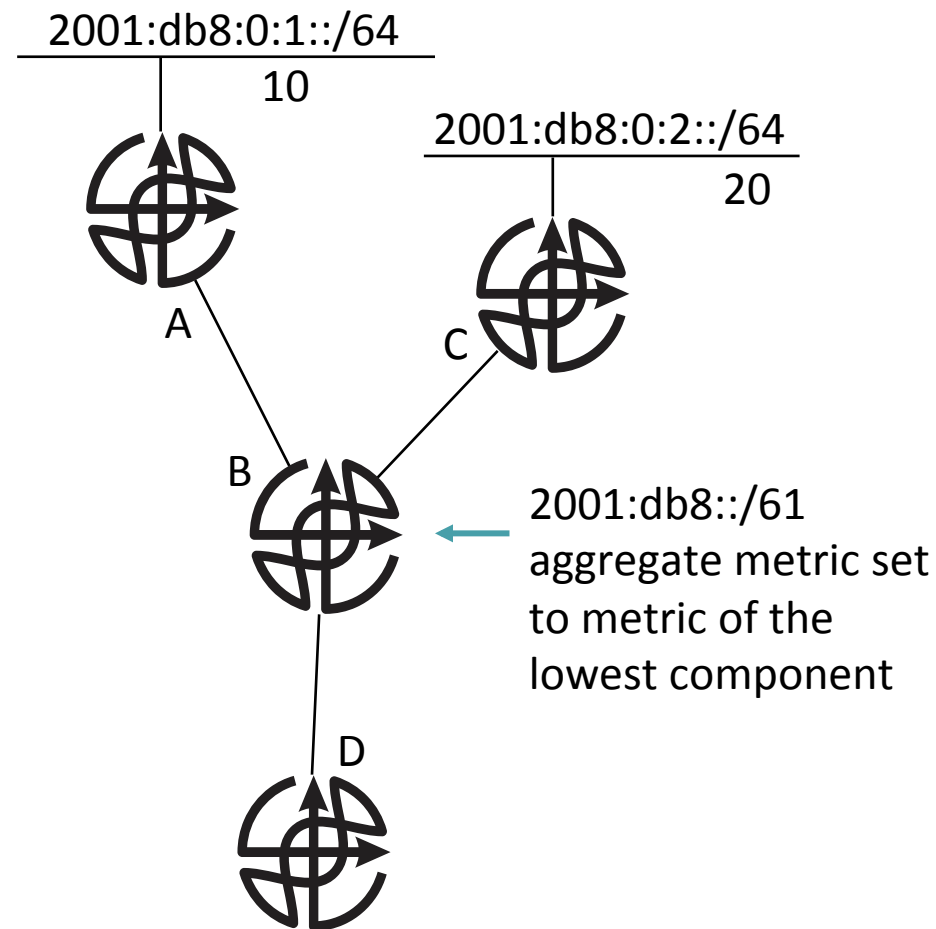
- Acts as an abstractor
 - Only look at the stuff you need to understand the system
- Closes off rabbit trails
 - “Why does IS-IS elect a DIS?”
 - “That’s a good question, but it’s not related to fixing these slow DNS queries...”
- Connects the technical to the business
 - Unless you can explain why this problem is important to solve...
 - It’s probably not!

What is this like?

Common Problem	Common Solution	Common Problems with the Solution
Find the shortest path	SPF	Microloops, state too fast
	Path Vector	Slow convergence, inconsistent state
	Bellman-Ford	Slow convergence, feedback loops
	DUAL	Drops during convergence, feedback loops
Distribute Data	Flood	Possibly too dense, CAP, transmission errors
	Multicast	Complex control plane, CAP, transmission errors
	Unicast	Complex control plane, CAP, transmission errors
Reliable Data Transmission	Forward Correction	Carry unneeded state in many cases
	Detect/Retransmit	Slows down transmission

Models and Abstractions

- Abstractions Leak
- Aggregation
 - If the A->B link fails, the cost of the aggregate changes
- TCP retransmissions
- Embedded IP addresses



Models and Abstractions

- Abstractions hide things
 - You might not understand the entire system as well as you might by studying it “in detail”
 - But comprehending a complex system without abstractions often just leads to confusion
- More than one model will help expose different aspects of the system

Why?

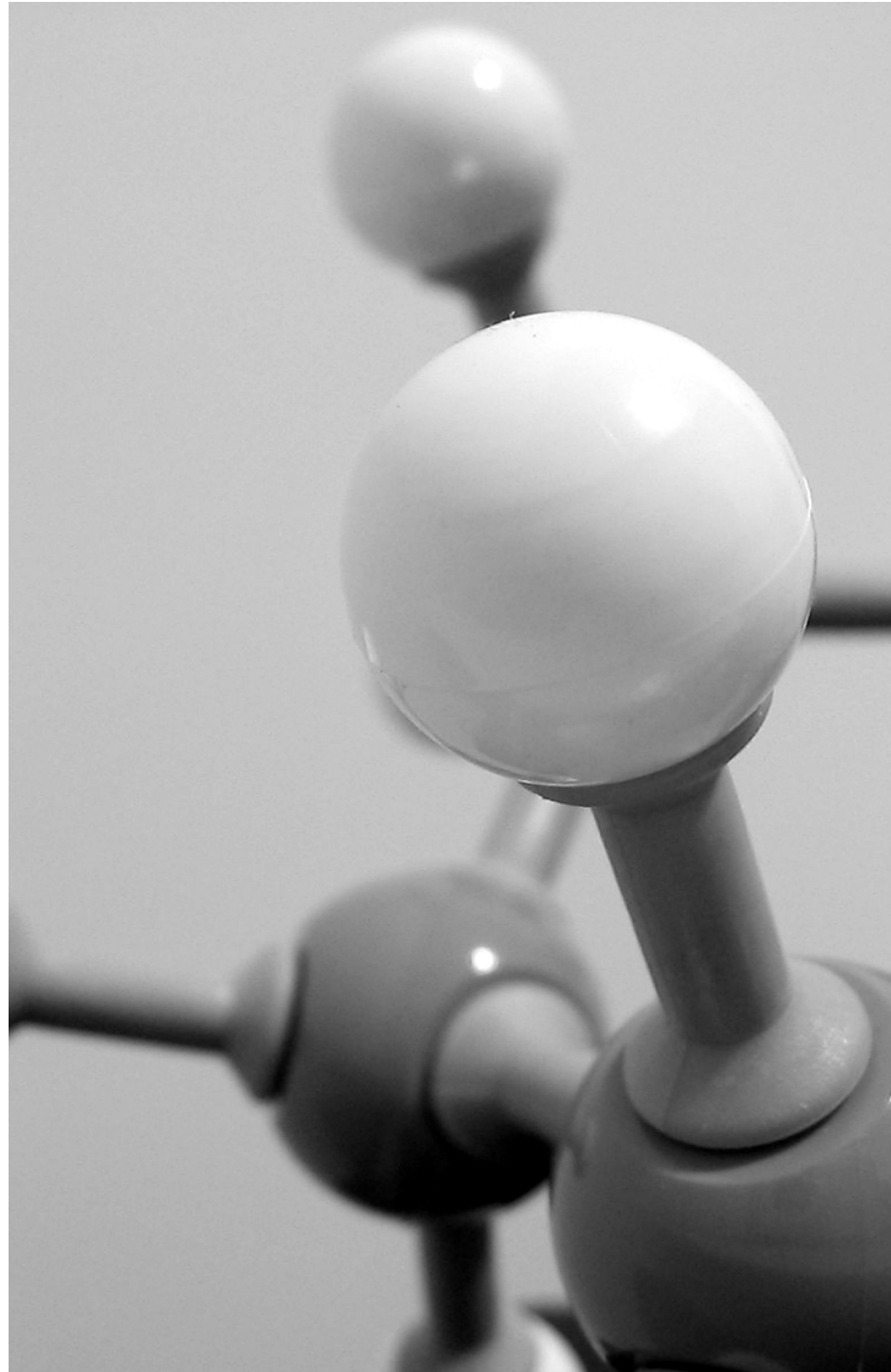
State/Surface/Optimization

What?

How?

What is this like?

to find potential solutions and “know
where to look”

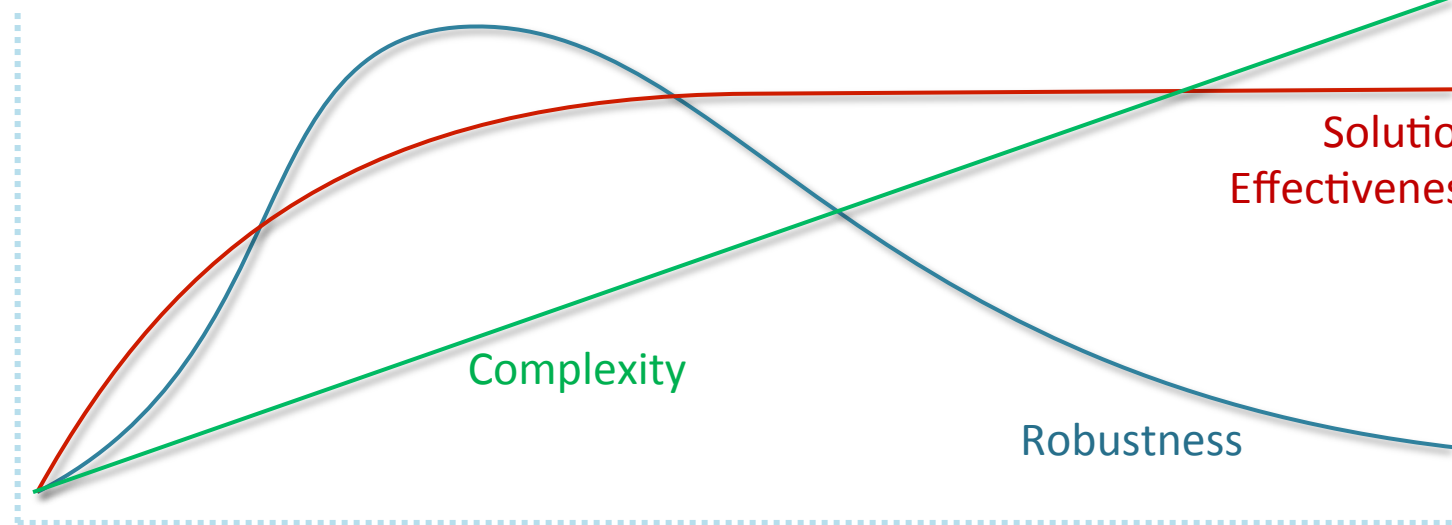


up, review,
summary,
abstract, r
ment.

Summary

Summary

- Complexity is necessary

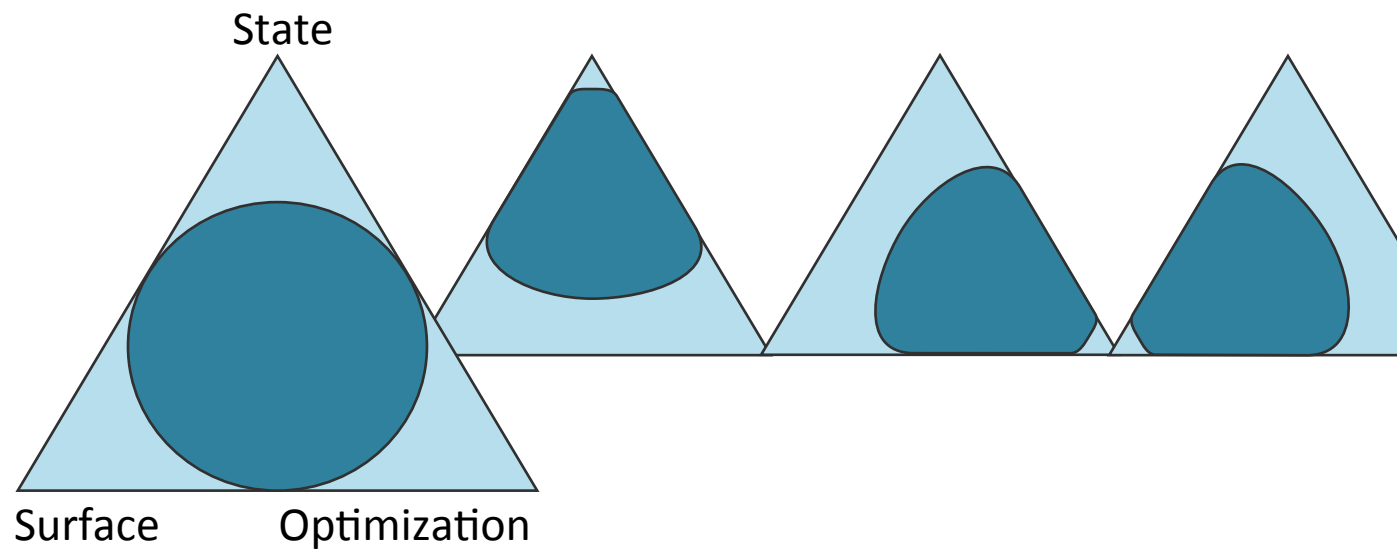


*In our view, however, complexity is most succinctly discussed in terms of functionality and its robustness. Specifically, we argue that **complexity in highly organized systems arises primarily from design strategies intended to create robustness to uncertainty** in their environment and component parts.*

Alderson, D. and J. Doyle, "Contrasting Views of Complexity and Their Implications For Network-Centric Infrastructures", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 40, NO. 4, JULY 2010

Summary

- Complexity is a tradeoff



It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it.

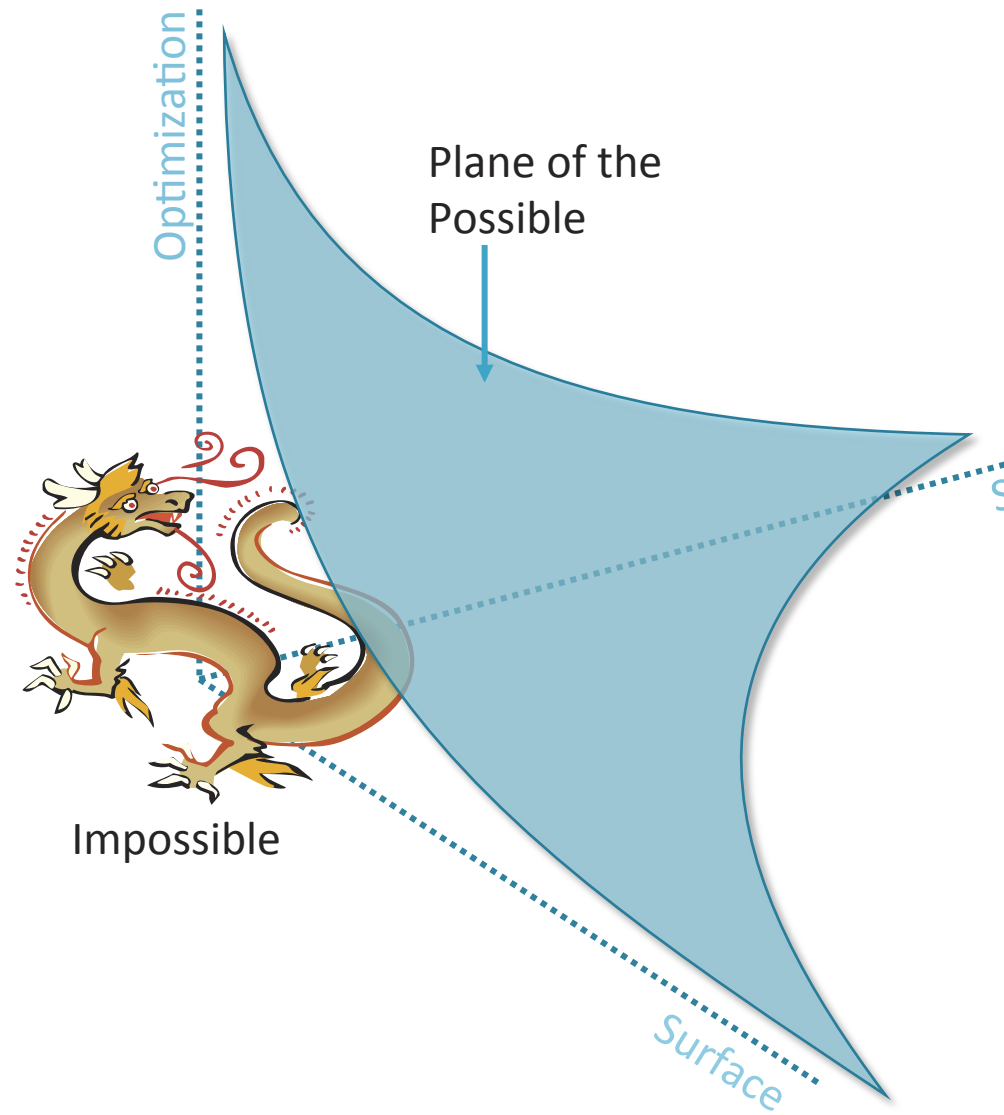
...

It is always possible to add another level of indirection.

RFC1925

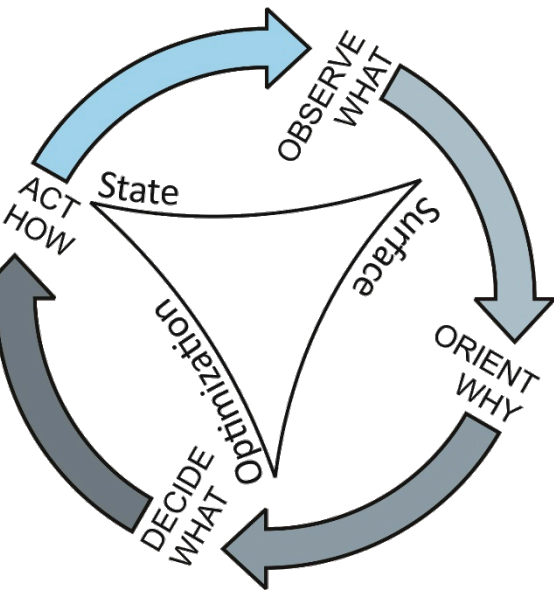
Summary

- You can't "solve" complexity



Summary

- But you *can* manage complexity



- Ask *why, what, how, and what is this like*
- Look for the state
 - How much and how fast
- Look for surfaces
- Ask what you're optimizing for
- Ask where you're losing optimization