Complexity didn't kill the cat

Keynote | BSides Bern - 2024

Dr. Stefan Frei

Security Officer @ sdx.com / lecturer @ ETH Zurich

frei@techzoom.net | BlueSky stefanfrei.bsky.social

TLDR

Complexity and connectivity are growing and we're left with systems beyond our full understanding. As the nature of incidents is changing with increasing system complexity, our models and understanding of the cause of incidents need to change.

Failures in complex systems are typically the result of multiple interacting factors. These factors can include technical issues, human errors, organizational problems, and environmental conditions. The interactions between these factors can be nonlinear and unpredictable, there is no longer a single root cause to the system failure.

Uncertainty is increasing which limits our ability to predict novel attack types. One hundred percent prevention is an illusion. We need to shift our mindset from “assumption of protection” to “assumption of breach.”

The only thing that ever yielded real security gains was controlling complexity. Complexity is not the enemy of security. Bad design is. The goal of effective defense is to ensure the critical functions and ultimately the services that the systems provide are maintained in the face of disruptions.

Our best defense is building resilient systems that can tolerate the inevitable attacks. We need to transition from focusing on individual threats and attack prevention to ensuring the resilience and continuity of critical services.

An Example

NAVY AIRCRAFT WERE FERRYING MISSILES ..

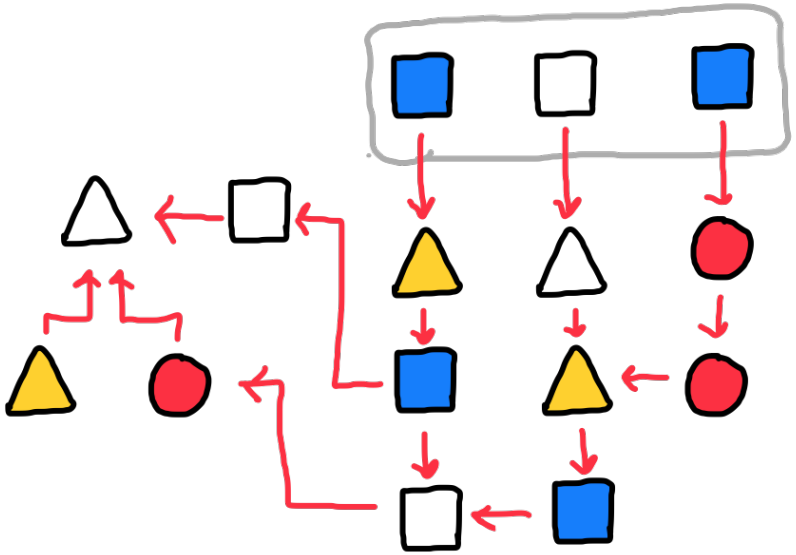
- A pilot was told to execute a planned test by aiming at the aircraft in front and firing a dummy missile
- Nobody knew that the fire control software was designed to use a different missile if the one selected to be fired was not in a good position
- In this case, there was an antenna between the dummy missile and the target, so the software decided to fire a live missile located in a different (better) position instead

“Who or what do we blame for this incident?”

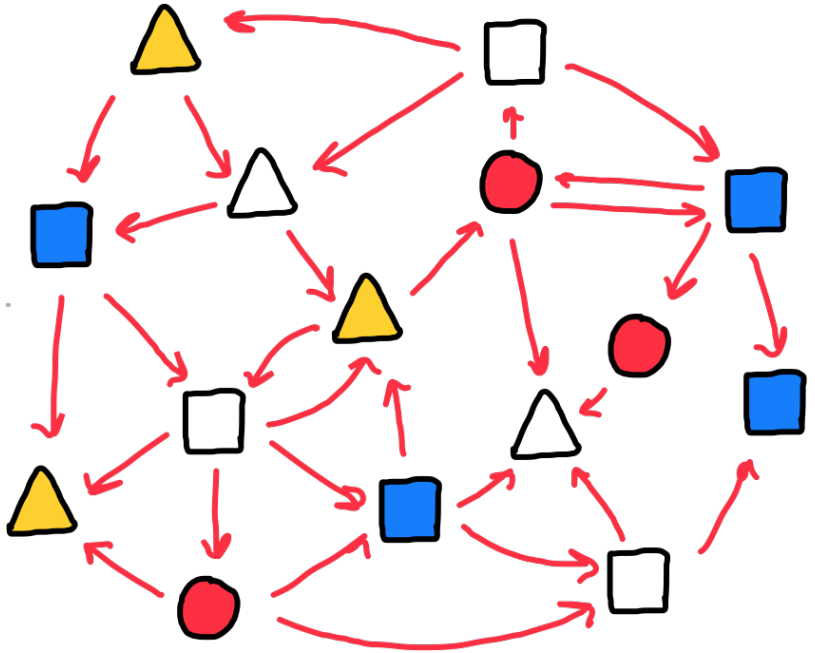
- What aircraft component(s) failed here?
- Can we use our usual excuse and blame the human operator (pilot)?

We build and use
complex systems that
we no longer fully understand

DIFFERENT TYPES OF SYSTEMS



SIMPLE



COMPLEX

tightly coupled
software intensive
interconnected

COMPLEXITY ISSUES

UNPREDICTABILITY

Complexity means that a large number of interacting and diverse parts give rise to outcomes **that are really hard, if not impossible, to foresee**

EMERGING BEHAVIOR

Properties of complex systems as a whole are **very different, and often unexpected, from properties of their individual parts** and cannot be deduced from the parts
e.g. wetness, bird flocks or fish schools, market dynamics

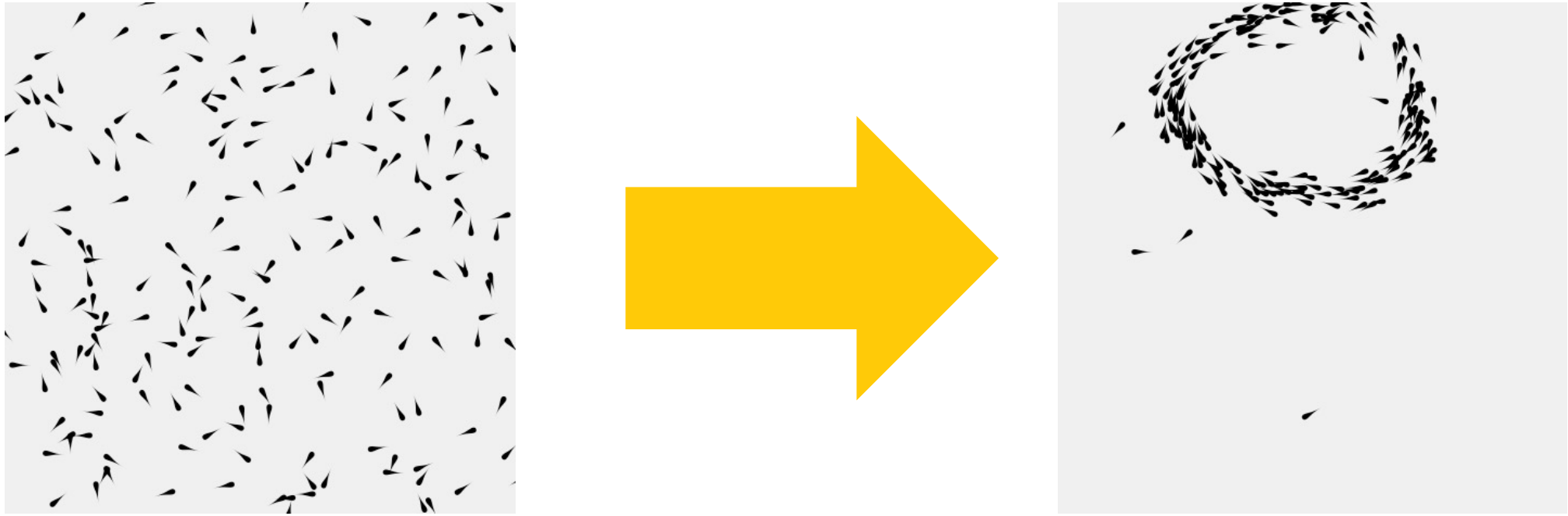
SENSITIVITY NON LINEARITY

Complex systems are non-linear, where **small changes can cause significant shifts in behavior** like **tipping points**. Some systems are chaotic, highly sensitive to small perturbations (butterfly effect)

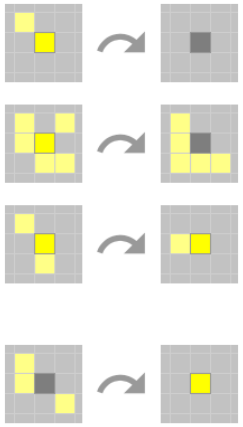
EMERGENCE: SWARMING AND FLOCKING IN ANIMALS

Individuals move at constant speed and noisy directional changes and ..

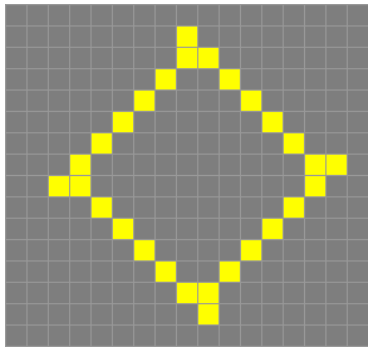
- avoid collisions
- align with others in their neighborhood
- try to move towards the center of mass of all individuals

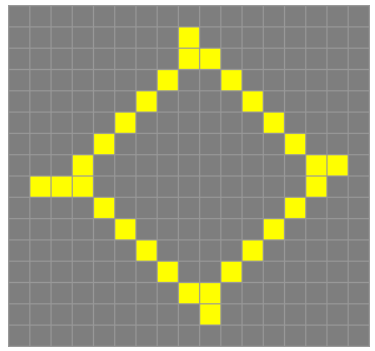


EXAMPLE: GAME OF LIFE



- Each cell has 8 neighbors and is either populated or empty
- A cell with less than 2 neighbors dies (by solitude)
- A cell with more than 3 neighbors dies (by overpopulation)
- A cell with 2 or 3 neighbors survives
- An empty cell with 3 neighbors becomes populated





Demo: https://playgameoflife.com

COMPLEX SYSTEMS ARE HARDER TO UNDERSTAND

A SIMPLE SYSTEM CAN BE SIMPLIFIED INTO SIMPLER SUBSYSTEMS

One can solve each simplified subsystem to solve the whole

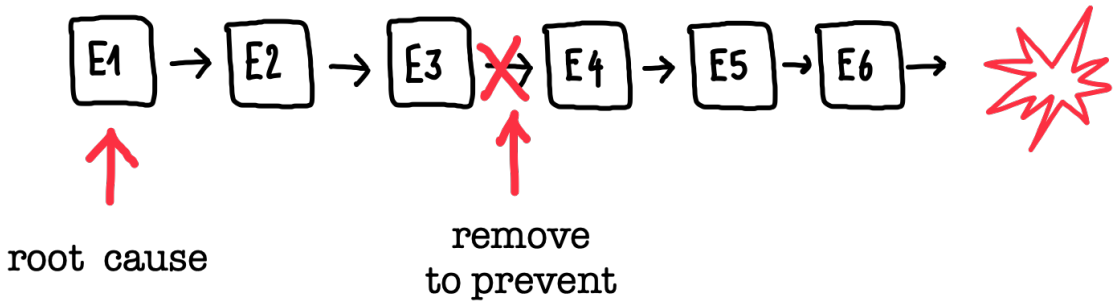
A COMPLEX SYSTEM CANNOT BE SIMPLIFIED

A complex system requires different methodologies for its investigation

Perspective Matters

TRADITIONAL VIEW – FOCUS ON *COMPONENTS*

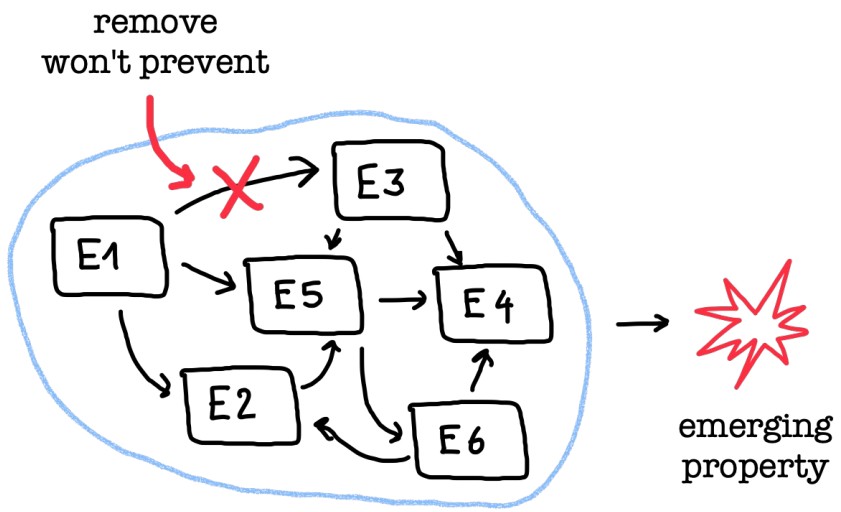
- System failures require component failures or human errors, which then cascade through the system
- The initial error (root cause) triggers subsequent failures until the final loss occurs



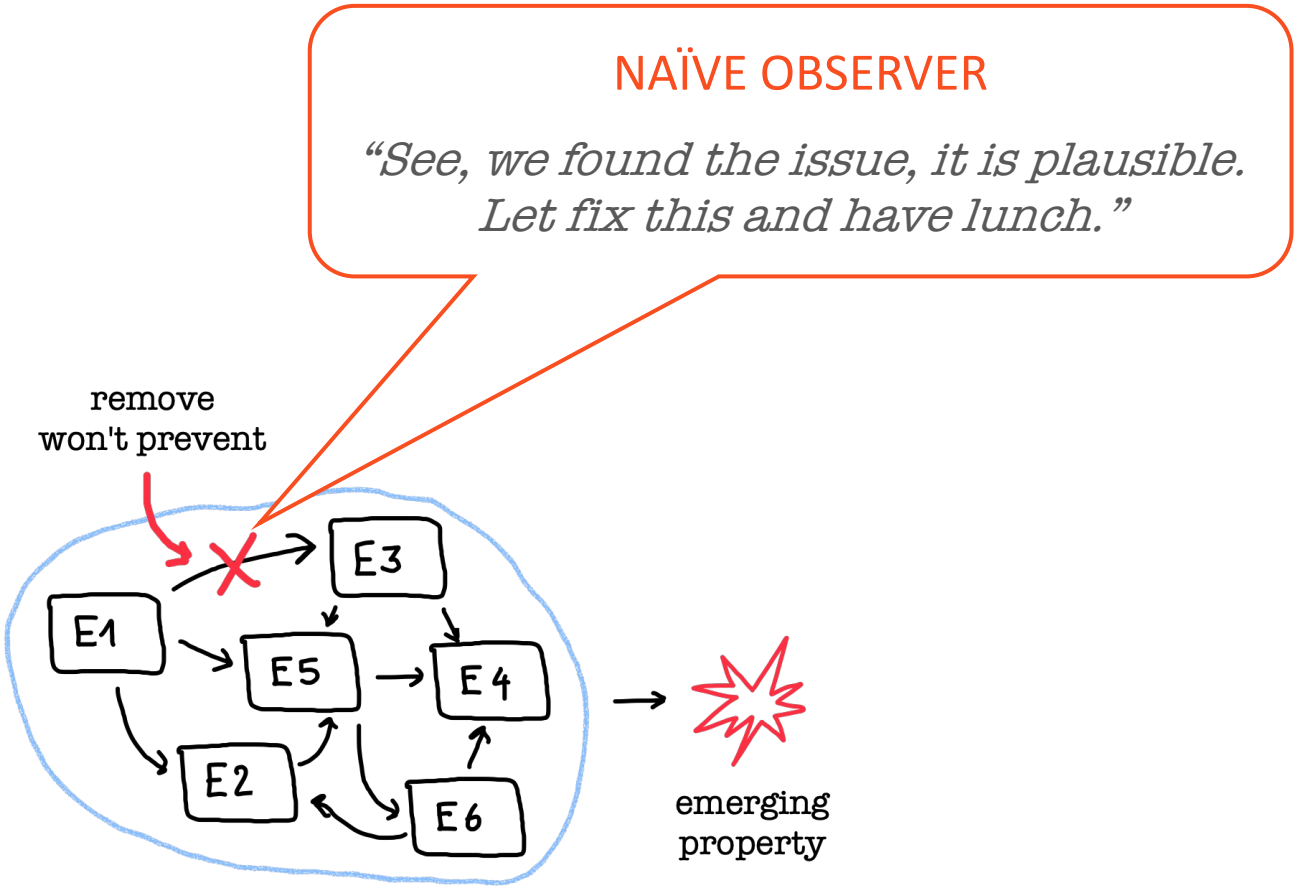
These assumptions are no longer true in our tightly coupled,
software intensive, highly automated, and interconnected systems today

SYSTEMS VIEW - FOCUS ON *INTERACTIONS*

- Complexity has given rise to a new type of accident called system accident
- In a system accident no component needs to be broken!
- Failures result from interactions and are typically the result of multiple factors, not a single root cause



SYSTEMS VIEW - FOCUS ON *INTERACTIONS*



Safety and security are emergent properties of a system which we cannot understand at the component level

SECURITY IS AN EMERGENT PHENOMENA

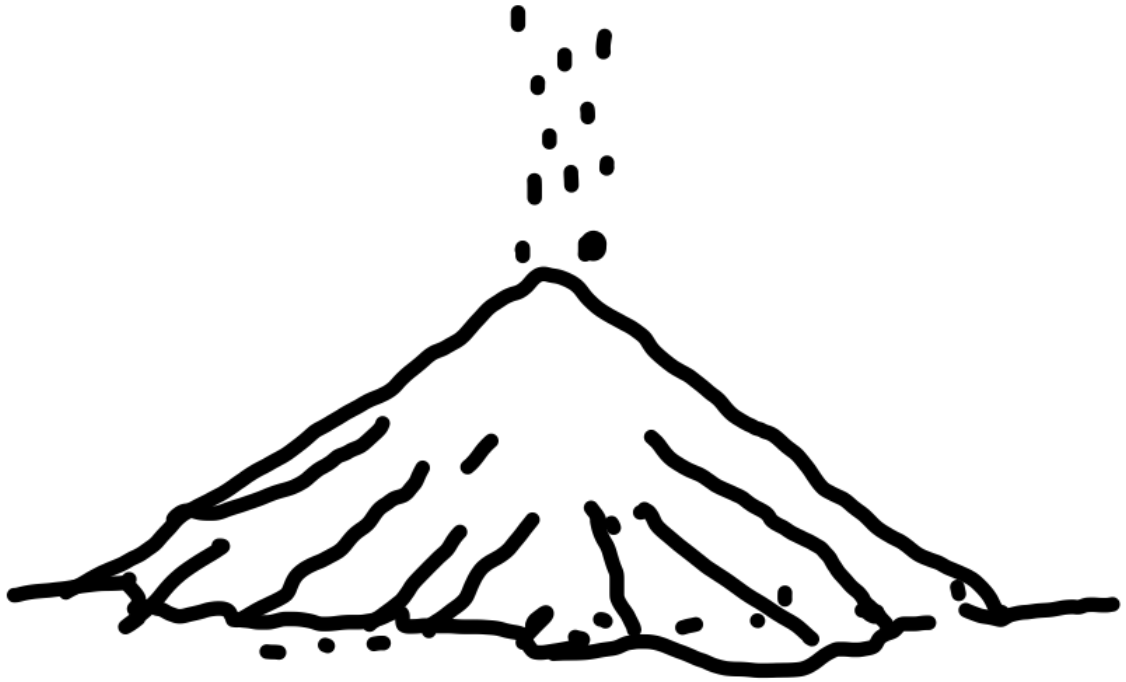
It is not possible to take a single system component (software, device, or human action) in isolation and assess its security

A component that is perfectly safe in one system or in one environment may not be when used in another

The security of individual components does not imply the security of the system

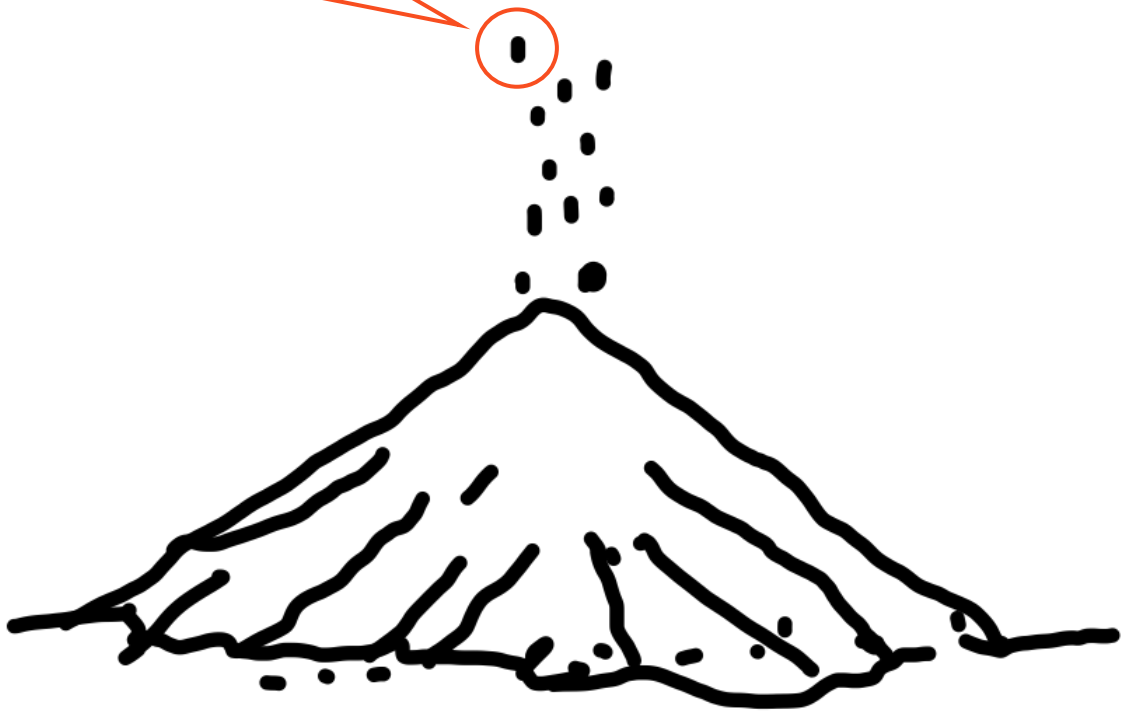
Sandpiles
and
Cyber Security

Consider a sandpile that will eventually collapse when adding sand on top



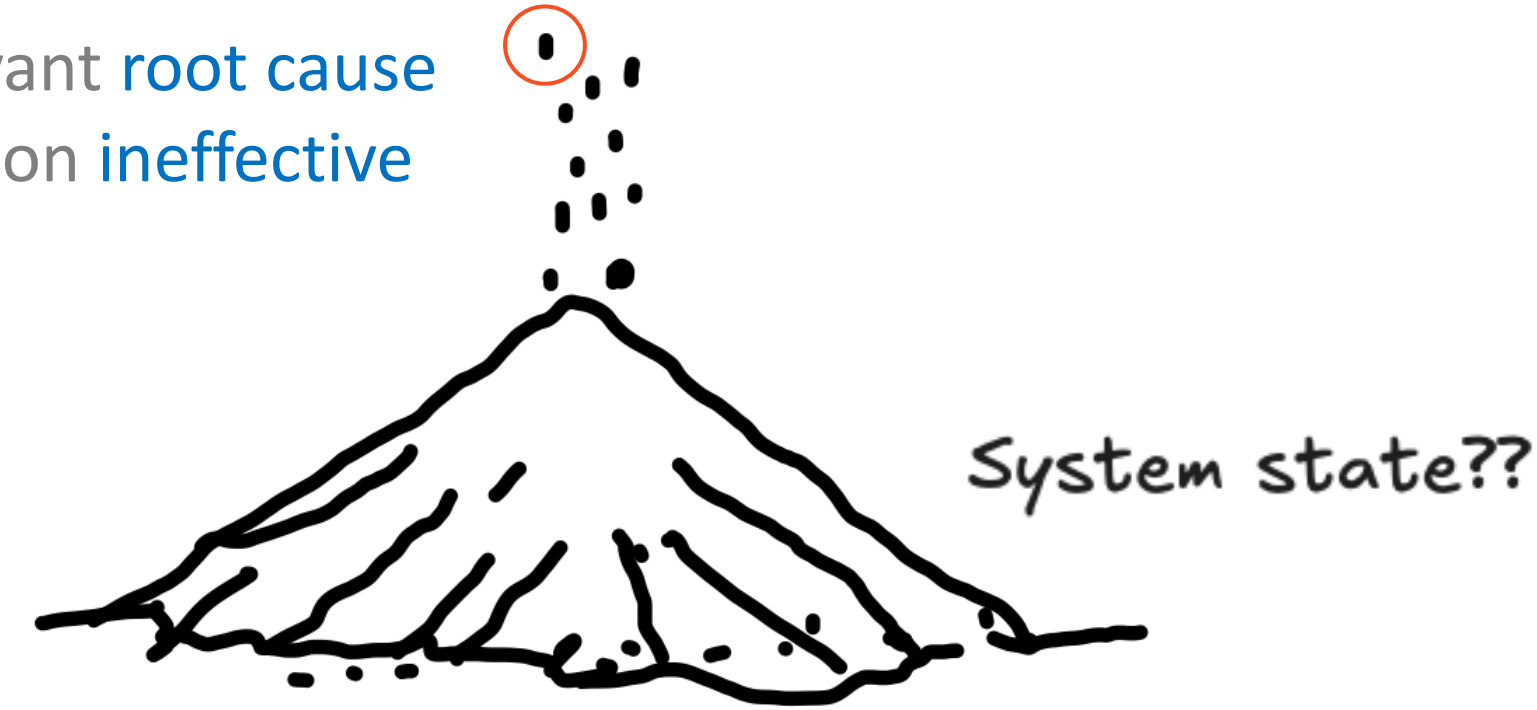
NAÏVE OBSERVER

“This caused the collapse!
Let’s go and fix (punish) it.



By focusing on the last grain of sand we miss the big picture ..

we identify an irrelevant root cause and waste resources on ineffective countermeasures ..



.. and the system happily continues its walk towards criticality

"The issue worth discussing is not which grain of sand caused the cascade …

.. but the configuration of the system at a specific time."

Increasing Complexity
and
Cyber Security

TRADITIONAL SYSYEM

HARD LIMITS ON COMPLEXITY APPLY BY DESIGN

- Systems designed for a specific task already limit what can happen
- Available volume, weight, materials, etc. limit complexity

Physical constraints limit what the system can do

TRADITIONAL SYSYEM

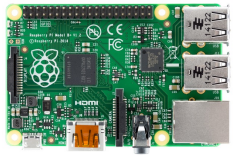
HARD LIMITS ON COMPLEXITY APPLY BY DESIGN

- Systems designed for a specific task already limit what can happen
- Available volume, weight, materials, etc. limit complexity

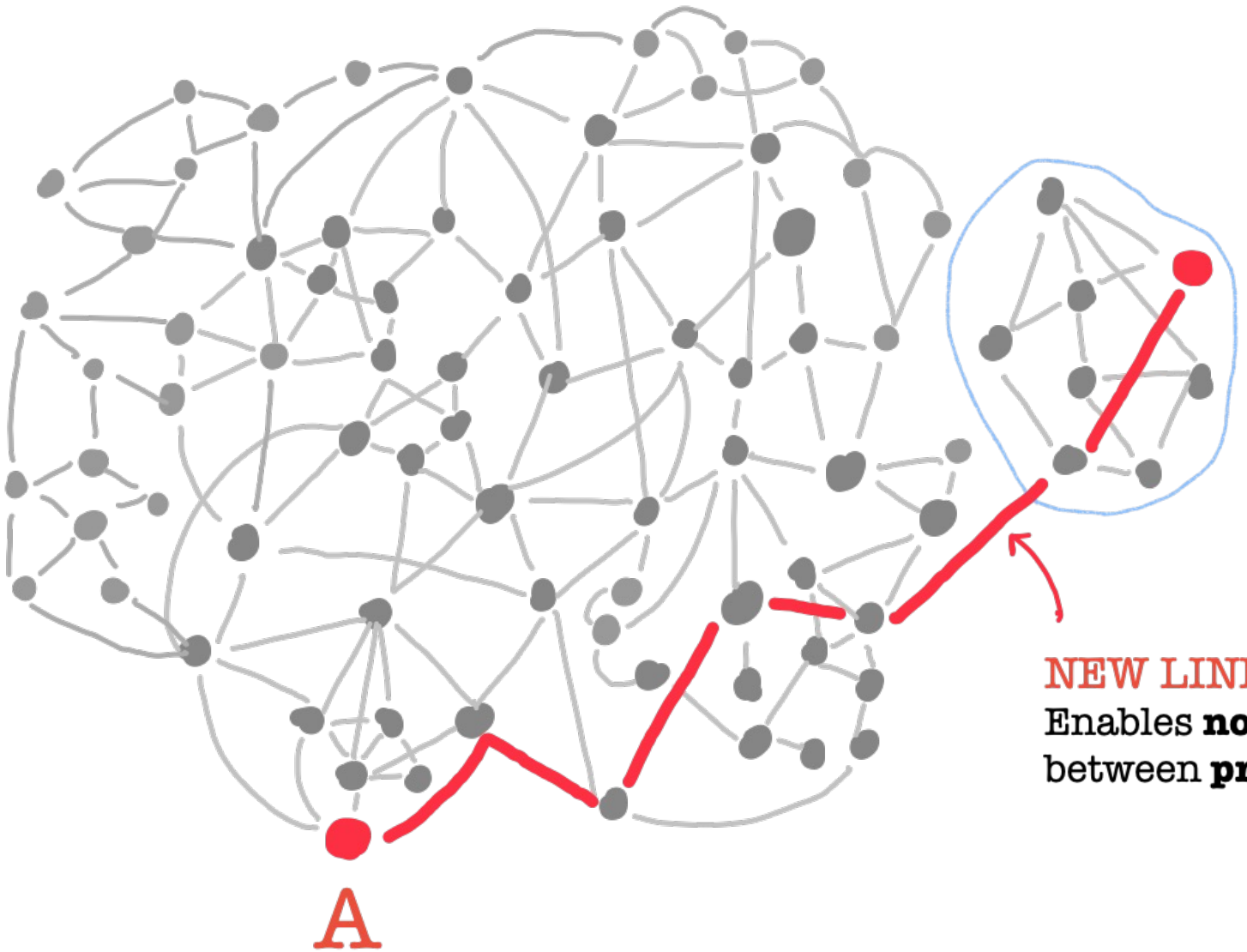
COMPUTER & SOFTWARE

WE HAVE TO BUILD “SECURITY” LIMITS INTO THE SOFTWARE

- Investment in time, effort, and funding
- When something goes wrong, the limits we imposed can go out of the window



Building “security” into a system involves limiting what can happen in the system



NEW LINK

Enables **novel types of interactions** between **previously isolated parts**

ASSUME COMPROMISE

Each new component or link exponentially *increases the number of interaction pathways*

Some new interactions *can be exploited* in novel ways giving rise to *new forms of attack*

Keeping pace in ***detecting novel attacks is an asymmetric arms-race*** favoring the attacker: *Predictability decreases*

What we can do

What we should do

What we shouldn’t do

PROBABILITY vs. CONSEQUENCES

Focus on the consequences rather than the probability

which you can know

which you can't know

x		$f(x)$
EVENT	$vs.$	CONSEQUENCE
Earthquake		Building collapsed
Hurricane		Power outage
Ransomware		Loss of data
0-day exploit		Systems hacked
Cloud failure		Application down

PROBABILITY vs. CONSEQUENCES

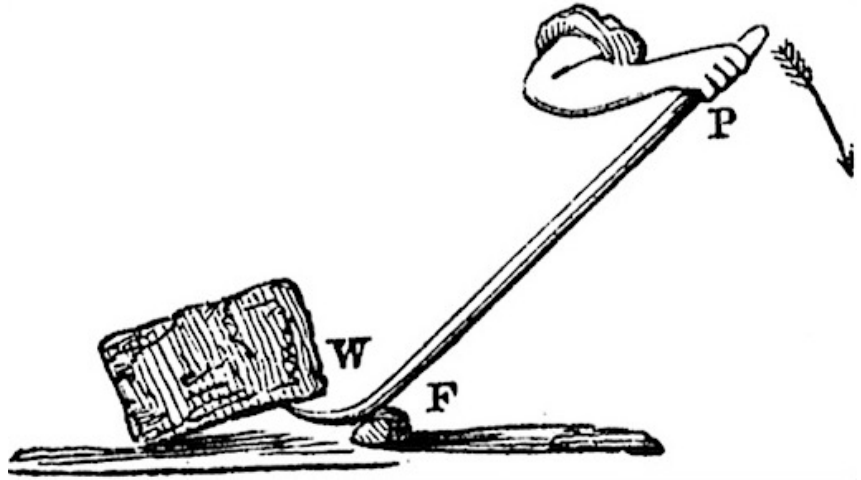
x		$f(x)$
EVENT	$vs.$	CONSEQUENCE
Earthquake		Building collapsed
Hurricane		Power outage
Ransomware		Loss of data
0-day exploit		Systems hacked
Cloud failure		Application down

- Building standards
- Alerts to stop trains/pipelines
- Underground cabling
- Encrypt data at-rest
- Backup & restore
- Network & access segmentation
- Prepared incident response
- Critical functions hosted locally
- Distributed dependencies

CONTROL IMPACT, NOT ATTACK

Focus on aspects of the problem that we can control:

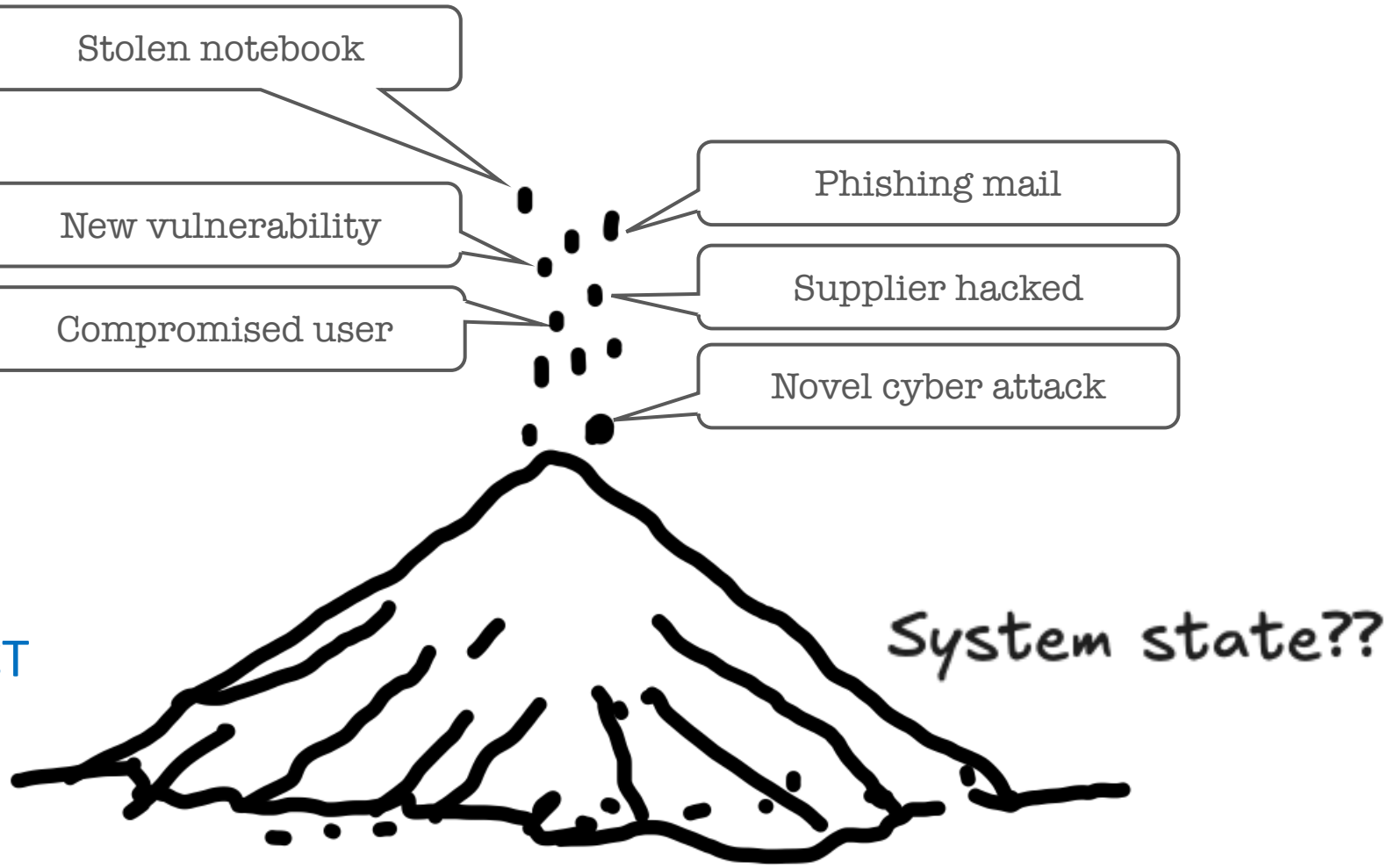
- Identify and protect critical assets rather than anticipating every possible attack
- This yields a much smaller and manageable set of high-level potential losses we need to address



Build systems that can absorb disruptions and recover fast.

DON’T CONTROL ATTACK

CONTROL DESIGN AND IMPACT

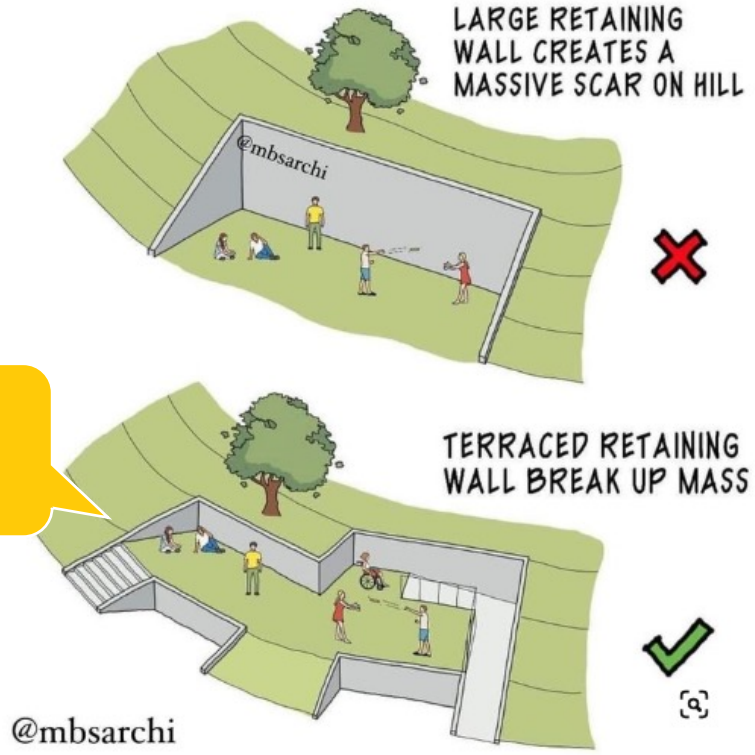


“Not seeing a tsunami, an economic event, or a cyber-attack coming is excusable. Building something fragile to them is not”

- Complexity is not the enemy of security
- Bad design is

RESILIENT SYSTEMS

Prevent, **absorb**, **recover** from, and **adapt** to an adverse occurrence



No need to predict rain

- Network Segmentation
- Zero Trust
- Vulnerability Management
- Conditional Access
- Backup & Recovery
- Desaster Recovery Plans
- Redundant Systems
- Chaos Engineering

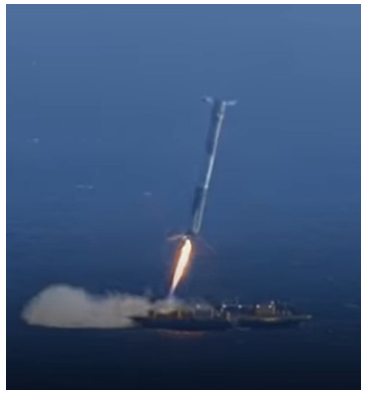
Dealing with Uncertainty & Change

THE ROLE OF TESTING ..

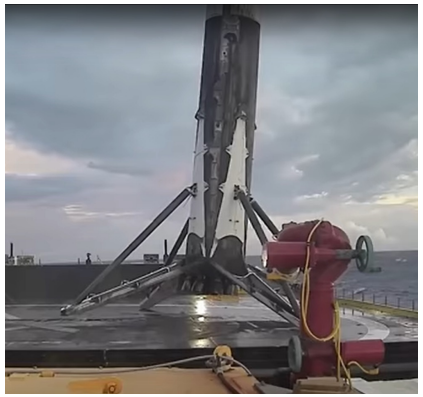
Failures are inevitable if you try something new ..

MISSED THIS ONE

CLOSE, BUT…



ALMOST



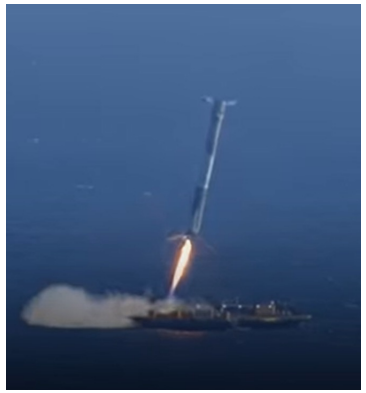
TESTING REDUCES THE UNKNOWNS

"The cause of the failure was not even on our risk list"

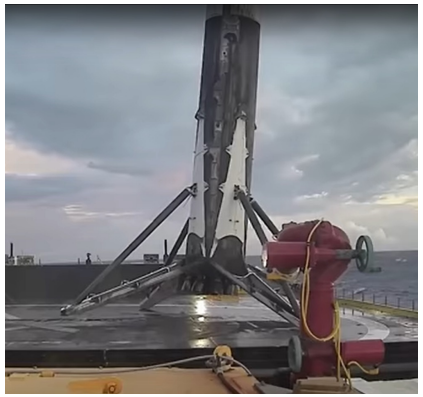
Space-X

MISSED THIS ONE

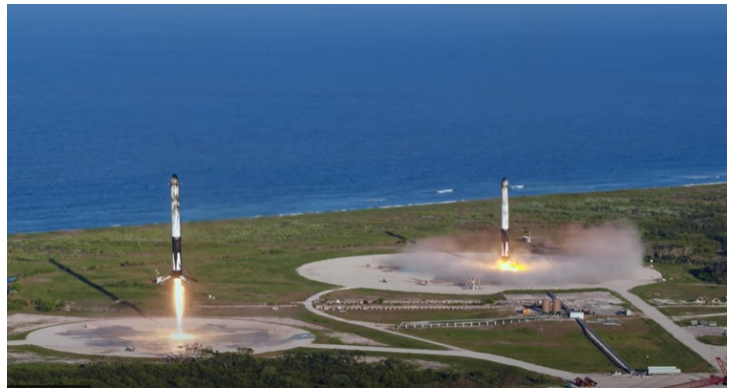
CLOSE, BUT…



ALMOST



BUSINESS AS USUAL



View failures as learning opportunities

Test early and often as the cause of failures might not even be on your risk list

CHAOS ENGINEERING

INTENTIONALLY INDUCE FAILURES IN *PRODUCTION ENVIRONMENT*

- Approach developed by Netflix in 2011 to improve resiliency
- Move from **assuming no breakdowns** to a model **with inevitable breakdowns**

ENHANCE ABILITY TO WITHSTAND DISRUPTIONS

- Understand how system will react to **unknowns**
- Create conditions needed to **uncover hidden bugs**
- Drive teams to consider **resilience an obligation** rather than an option

Take home message

- Complex systems should not be viewed with models created for simple systems
- Control impact, not probability or attack
- Complexity is not the enemy of security
  Bad design is

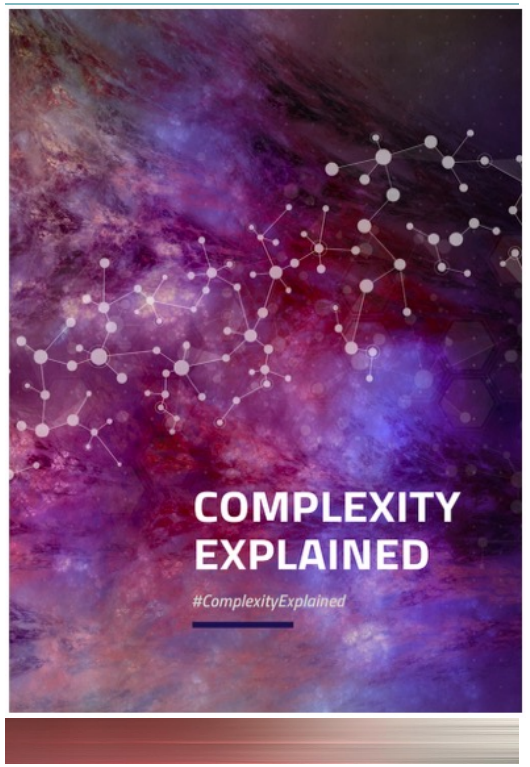
https://techzoom.net/book

Readers

Complexity Explained [Free Booklet] 2019

Complexity Explained

by M. De Domenico, D. Brockmann, C. Camargo, C. Gershenson, D. Goldsmith, S. Jeschonnek, L. Kay, S. Nichele, J.R. Nicolás, T. Schmickl, M. Stella, J. Brandoff, A.J. Martínez Salinas, H. Sayama.



Outline

The "Complexity Explained" booklet introduces the concept of complexity science, which studies how simple interactions at a small scale can lead to complex behaviors and patterns at a larger scale. It covers topics like the dynamics of complex systems, which can show unpredictable behavior over time, and the phenomenon of self-organization, where system components spontaneously form ordered patterns. Additionally, it discusses adaptation in complex systems and the interdisciplinary nature of complexity science, applying its principles across various domains to understand and manage complex systems effectively.

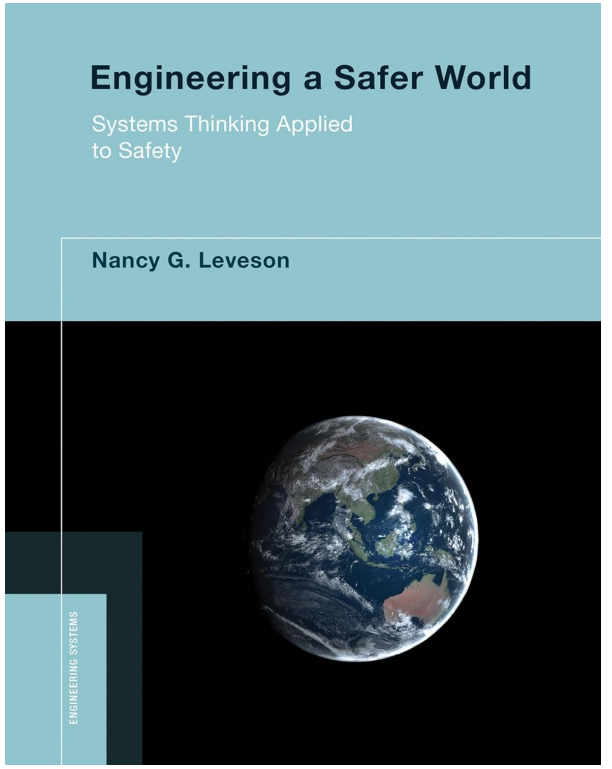
Source

- Complexity Explained (2019). DOI 10.17605/OSF.IO/TQGNW 10.17605/OSF.IO/TQGNW
- GitHub - https://complexityexplained.github.io

Engineering a Safer World 2016

Engineering a Safer World: Systems Thinking Applied to Safety

by Nancy G. Leveson



Outline

"Engineering a Safer World: Systems Thinking Applied to Safety" by Nancy G. Leveson proposes a shift from traditional safety techniques to a more holistic and integrated approach to safety and risk management. Leveson introduces systems thinking as a method to design safer systems in various industries, highlighting the interconnectedness and complexity of modern technological systems. The book critiques the inadequacy of current safety practices that fail to keep up with the increasing complexity of global technological systems. Through case studies and a new accident model and process called STAMP (Systems-Theoretic Accident Model and Processes), the author provides insights on how to anticipate and prevent systemic failures in safety-critical systems.

Source

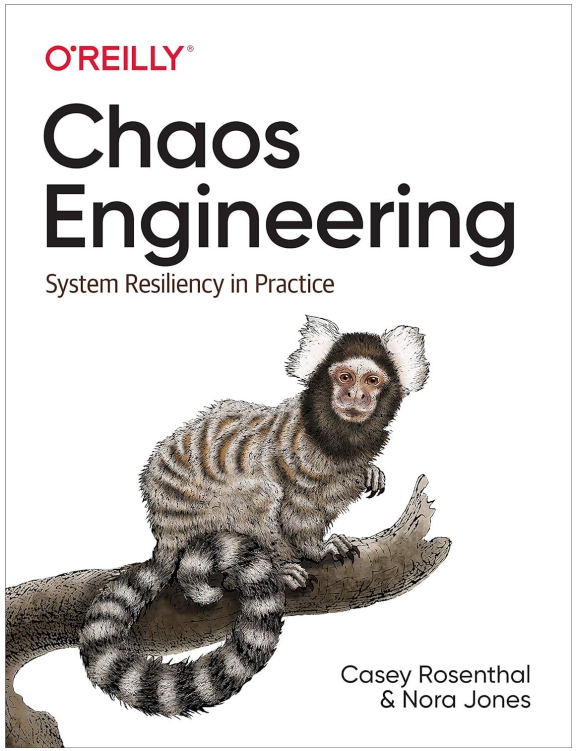
- MIT Press (free download) - https://direct.mit.edu/books/oa-monograph/2908/Engineering-a-Safer-WorldSystems-Thinking-Applied
- Amazon - https://www.amazon.com/Engineering-Safer-World-Systems-Thinking/dp/0262533693

Chaos Engineering

2020

Chaos Engineering: System Resiliency in Practice

by Casey Rosenthal , Nora Jones



Outline

"Chaos Engineering: System Resiliency in Practice" by Casey Rosenthal and Nora Jones is a practical guide that delves into the innovative field of chaos engineering, a discipline aimed at improving system resilience through proactive experimentation. The book outlines how intentionally introducing controlled disruptions into a system can help organizations discover hidden vulnerabilities, thereby allowing them to enhance the system's robustness against unforeseen failures. Through real-world examples and case studies, Rosenthal and Jones demonstrate how chaos engineering practices have been successfully implemented by leading technology companies to ensure their systems can withstand turbulent and unexpected conditions. This book is essential for software engineers, systems architects, and IT professionals seeking to apply chaos engineering principles to build more resilient, reliable, and fault-tolerant systems in an era where digital services are critical.

Source

- Wikipedia - https://en.wikipedia.org/wiki/Chaos_engineering
- Amazon - https://www.amazon.com/Chaos-Engineering-System-Resiliency-Practice/dp/1492043869

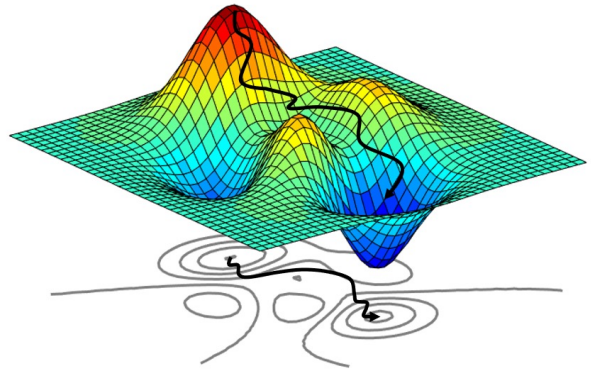
OPTIMIZATION MAKES YOU FRAGILE

THERE IS CONSTANT PRESSURE TO OPTIMIZE

- Optimized systems are designed to perform optimally within a narrow set of conditions
- This can lead to fragility over time, especially when subject to changing conditions

INCREASED SENSITIVITY

- Optimized systems are highly sensitive to slight deviations from the expected conditions



STRATEGIC DECISION

- Balance of optimization vs. long-term sustainability or resilience

Optimization requires a thoughtful consideration of the trade-offs between short-term efficiency and enduring robustness and adaptability

IDENTIFY SYSTEM WEAKNESSES

We learn about the behavior of a distributed system by observing it during a controlled experiment.

- Improper fallback settings when a service is unavailable
- Retry storms from improperly tuned timeouts
- Outages when a downstream dependency receives too much traffic
- Cascading failures when a single point of failure crashes
- Cyber attacks or sabotage

Rather than waiting for a system failure, at unfortunate times,
proactively simulate failures to uncover hidden issues and weaknesses