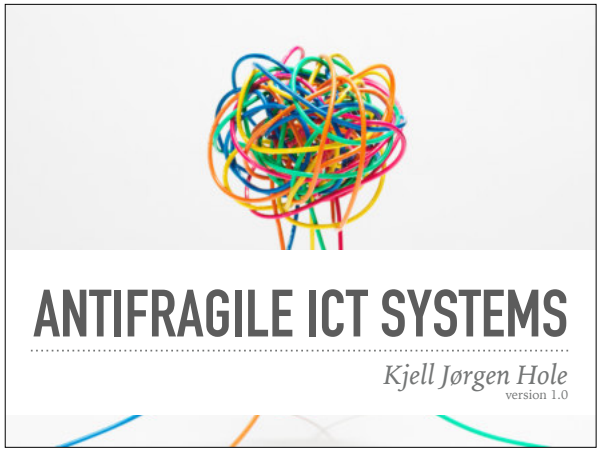
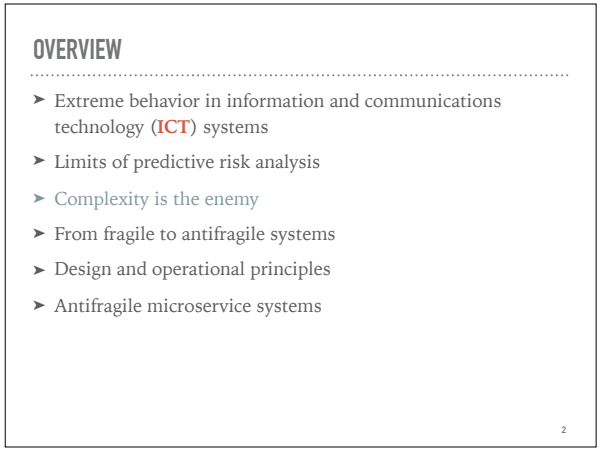


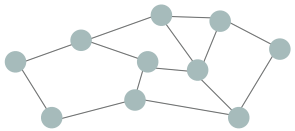
This talk models ICT systems as complex adaptive systems, discusses extreme behavior in these systems, argues why risk analysis will not reliably predict rare extreme behavior, and explains why we need to develop and operate antifragile systems.



We first model ICT systems as complex adaptive systems to understand why stakeholders are surprised by extreme behavior with intolerable consequences.

COMPLEX ADAPTIVE SYSTEM

- Man-made or natural system
- Consists of many entities that interact in involved ways
- Entities adapt to each other and the environment
- Adaption allows system to withstand perturbations



4

There is no single formalism that captures all properties of a complex adaptive system. Complex systems are often represented by graphs. While graphs can display the communication paths or the dependencies between different parts of a system, they cannot fully represent the emergent behavior caused by the interacting processes in the system.

EXAMPLES OF COMPLEX ADAPTIVE SYSTEMS

- The world-wide economic system
- National political systems
- Transportation systems
- Immune systems
- The Internet
- Beehives
- Anthills
- Brains
- ICT systems

5

We'll concentrate on ICT systems in this lecture series.

COMPLEX ADAPTIVE ICT SYSTEMS

- A complex adaptive ICT system consists of
  - stakeholders
  - technologies
  - threats agents
  - policies
- The complexity is mostly due to:
  - interactions between stakeholders and the networked computer system
  - communication between computers in the network

6

There is no single accepted measure of a system's complexity. A complex adaptive ICT system must be modeled in different ways depending on the properties being studied. Here, we develop a simple model to better understand why there is extreme behavior in complex adaptive ICT systems.

EXAMPLES OF COMPLEX ICT SYSTEMS

- Cloud computing infrastructures
- Telecom infrastructures
- Online social networks
- Banking systems
- Power grids

7

EXAMPLES OF STAKEHOLDERS

- Examples of stakeholders with interest in an ICT system are
  - Software architects and developers
  - System owners, operators, and users
  - Governmental supervisory entities

8

Stakeholders contribute to the complexity of a system by introducing, perhaps, conflicting requirements and regulations. Some of these stakeholders may even encourage risky behavior to reach certain goals. There is also a tendency among stakeholders to withhold information about design flaws and bad management of systems. This risk hiding leads to overconfidence and cause a slow “drift into failure” with intolerable consequences.

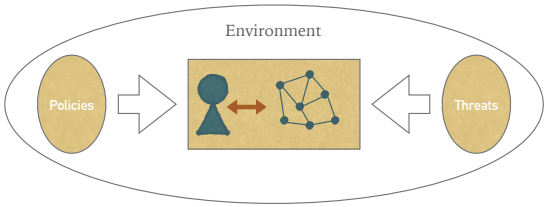
EXAMPLES OF THREATS AGENTS

- Benevolent users and operators making security related mistakes
- Insider attacks from malicious system operators
- Outsider attacks from hackers exploiting software bugs or design flaws
- Hardware failures

9

Threat agents trick benevolent stakeholders and/or exploit vulnerabilities in a computer-based system to misuse protected assets.

COMPLEX ICT SYSTEM



Observe that the stakeholders are part of the system

10

In this talk, a complex adaptive ICT system consists of stakeholders, technologies, threats, and policies.

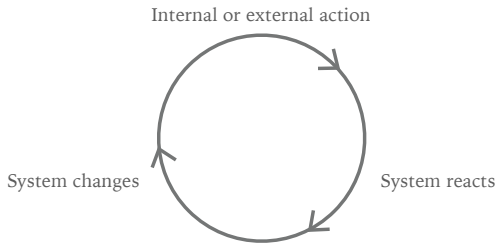
NEVER-ENDING CHANGE

- A complex adaptive ICT system's architecture, functionality, technology, environment, and regulatory context change over time
- Complex ICT systems never reach a final form
- They continue to adapt to satisfy the changing needs of stakeholders and to protect against changing threats
- A complex ICT system in "equilibrium" is a dead system

11

Constant change is an inherent property of complex adaptive ICT systems.

FEEDBACK LOOPS



12

Complex adaptive systems contain feedback loops. A feedback loop is a series of interacting processes that together result in a system adapting to the effect of its previous behavior. Feedback loops are what make complex systems adaptive. The loops create emergent global patterns or behaviors. While the concept of feedback loops is highly useful to explain adaption and extreme behavior, it is usually very hard to pinpoint all feedback loops in a real system.

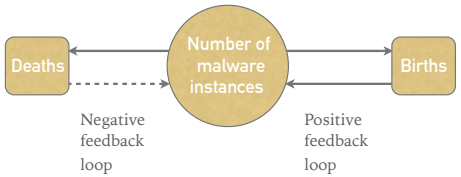
TYPES OF FEEDBACK LOOPS

- A feedback loop is a series of interacting processes, which cause a system to adapt its behavior based on previous behavior
- It is the feedback loops that make a complex system adaptive
- Positive feedback loops ***propagate local events*** into global behavior
- Negative feedback loops ***dampen local events***, preventing changes to global behavior

13

Positive feedback loops cause events to initiate more events (e.g., people buy a book because other people bought it). In particular, positive feedback loops amplify a complex system's parameter values and makes the system very sensitive to small changes in the initial conditions. In fact, changes in parameter values can make a system transit from one global pattern to another. However, a global pattern tend to be stable over a range of parameter values.

EXAMPLE: MALWARE EPIDEMIC

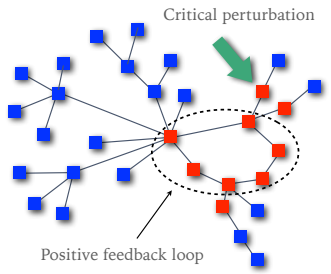


14

The figure depicts a simple model of an infectious malware epidemic that involves a positive feedback loop of increased births and a negative loop of increased deaths. Without deaths, the population size will increase exponentially. In other words, negative feedback is needed to keep the positive feedback under control.

EXAMPLE: FEEDBACK IN POWER GRID

- Feedback loop escalates the negative effect of local failure
- Local failure causes systemic failure



15

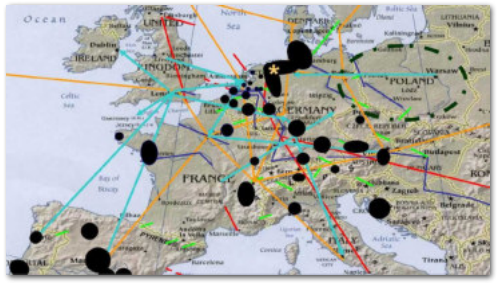
Examples of cascading events caused by positive feedback are blackouts in power grids, communication failure in mobile phone systems due to excessive signaling traffic, and "traffic jams" in computer networks.

POWER GRID IN EUROPE

- To allow for the transfer of a ship, one power line had to be temporarily disconnected in Northern Germany in November 2006
- The event triggered an overload-related cascading effect and many power lines went out of operation
- As a consequence, there were blackouts all over Europe (see black areas in picture)

16

D. Helbing, “Systemic Risks in Society and Economics,” SFI Working Paper 2009-12-044, 2009;
www.santafe.edu/media/workingpapers/09-12-044.pdf.



Blackout

17

Even to experts, the pattern of outages is surprising and very hard to foresee.

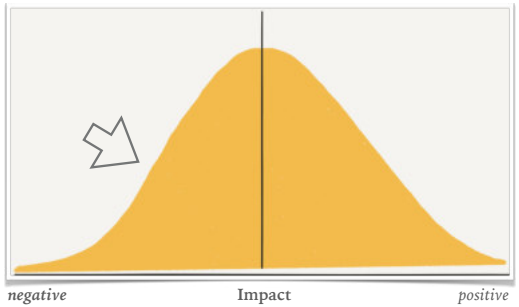
STOCHASTIC BEHAVIOR

- The **behavior** of a complex ICT system is modeled as a sequence of events that affect a group of stakeholders both positively and negatively
- We consider the financial **impact** of all possible events during a particular time period of five to ten years
- The high complexity makes it necessary to represent the impact by a stochastic variable that changes with time

18

For simplicity, we assume that the impact of events can be represented by financial gains or losses.

PROBABILITY DISTRIBUTION OF IMPACTS



19

Since we are interested in studying events with negative impact, we'll concentrate on the left half of the probability distribution.

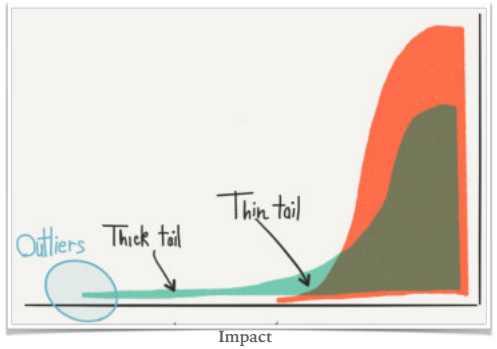
PROPERTIES OF IMPACT DISTRIBUTION

- Most of us are familiar with thin-tailed probability distributions with fixed expectation and well-defined variance
- The impact distribution for real-world ICT systems are likely to have
  - time-varying expectation
  - thick (fat) left tail
  - infinite variance

20

When the behavior of a complex adaptive system changes over time, the probability distribution of events also changes. If the system has a fat tail distribution, it is not possible to estimate the mean from samples since you need a huge number of samples. (The mean is determined by outliers that are very rare and unlikely to be in your samples.)

THICK LEFT TAIL



21

Power-law probability distributions with fat tails can be used to model (some aspects of) a complex adaptive system's behavior. The fat tails make it impossible to make statements about a system's extreme behavior from small samples.

PROPERTIES OF OUTLIERS

- Outliers are often caused by
  - positive feedback loops that propagate local failures into systemic failures
  - attackers exploiting software bugs and design flaws
  - single point of failures that take down whole systems
- ***Observation*** Since outliers are unlikely to be in a system's history, the past will not help us foresee outliers or calculate their probabilities

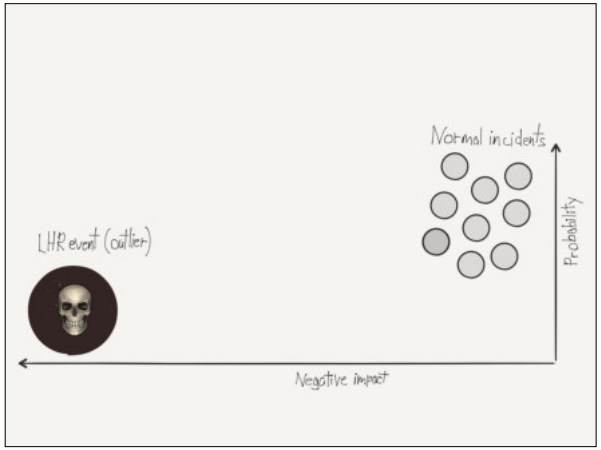
22

We assume that single points of failures are removed from the systems we concentrate on positive feedback loops

EXTREME BEHAVIOR—LHR EVENT

- A large impact, hard-to-predict, and rare (**LHR**) event is an outlier in the left tail of the probability distribution
- While "normal" events occur multiple times during a period of say ten years, LHR events are non-recurrent, that is, they occur at most once during the period

23



The figure illustrates the difference in probability and impact between non-recurrent LHR incidents and normal recurrent incidents.

LHR INCIDENT IN NORWEGIAN PAYMENT SERVICES

- In August 2001, computer systems providing services to about one million Norwegian bank customers ceased to function
- It took 7 days to get the services back in normal operation
- Multiple points of failure—causing transaction data on 288 disks to become inaccessible

25

EDB Fellesdata AS ran the computer services of about half of Norway's banks. On Thursday 2 Aug 2001, the company installed 288 new disks in its Hitachi storage. Then, instead of initializing the new disks, EDB initialized 288 existing disks, crashing the whole data storage system. According to news reports, about half of all Norwegian bank customers were denied access to online banks and ATMs by this LHR event. `https://news.hitb.org/node/3129`

LHR EVENT IN A LARGE NORWEGIAN BANK

- In March 2007, malware infected 11 000 PCs and 1 000 servers belonging to a Norwegian bank
- More than two weeks were needed to completely remove the malware
- An error in the anti-virus software and a vulnerability in the OS led to this LHR event

26

Viking.gt spread because there was a problem with the anti-virus software running on the bank's computers, and because the malware was able to exploit a vulnerability in the computers' OS. During the attack several branch offices were unable to assist their customers. About 50 employees and 150 external consultants were involved in the clean-up. It's estimated that this LHR event cost the bank more than 110 million NOK.

`www.idg.no/bransje/bransjenyheter/article45271.ece`
`www.dagensit.no/arkiv/article1339181.ece?WT.svl=article_title&jgo=`
`www.dagensit.no/arkiv/article1339199.ece?WT.svl=article_title&jgo=`

LHR EVENT: CONFICKER

- It is estimated that the Conficker worm has infected 12 million PCs world wide
- Conficker severely affected hospitals (Helse Vest) and the police in Norway
  - the Norwegian police spent 30–50 million NOK to "clean up" after Conficker attacked operational control centres and the system for passport control

27

The Conficker computer worm infected millions of government, business, and personal computers in more than 190 countries, threatening to overpower the computer networks that controlled health care, air traffic, and banking systems over the course of several weeks.

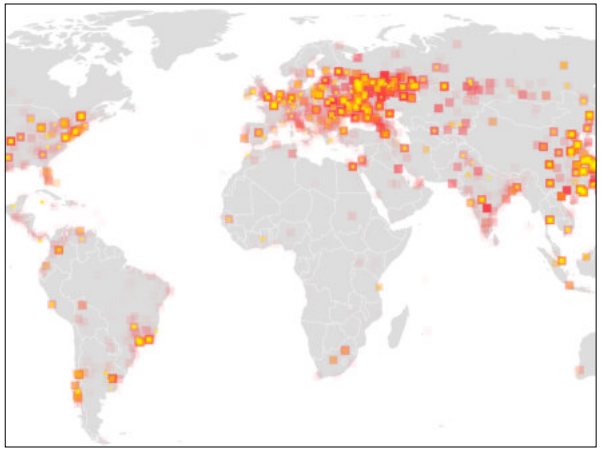
`www.digi.no/817553/dataorm-kostet-politiet-30-50-millioner-kroner`
`www.nrk.no/nyheter/distrikt/rogaland/1.6428721`

MASSIVE RANSOMEWARE ATTACK FROM NORTH KOREA

- Self-replicating ransomware infected 200,000 systems in more than 150 countries on May 12th, 2017
- First attack to use a stolen cyberweapon developed by NSA
  - Many targets in Russia, Ukraine, India, and Taiwan
  - 48 hospitals in Britain were affected by the outbreak
  - Renault had to stop production in some factories
  - Telefónica, a Spanish telecommunications firm was affected
  - FedEx

28

Ransomware is malicious software that encrypts a computer's files and then demands payment to unlock them. Cybercriminals have discovered that ransomware is the most effective way to make money in the shortest amount of time. Bitcoin has given the cybercriminals an easy and anonymous way to get their profits, and it is much harder to track than credit cards or wire transfers.



The map shows tens of thousands of Windows computers that were taken hostage by a variant of the WannaCry ransomware. This worm exploited a vulnerability in Microsoft servers first discovered by NSA. Europe was more affected than the United States because a British cybersecurity researcher inadvertently activated a kill switch when he bought a domain used by the attackers. The domain was hard-coded into the worm.
https://www.nytimes.com/interactive/2017/05/12/world/europe/wannacry-ransomware-map.html

---

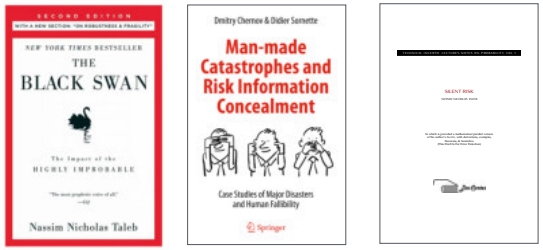
SUMMARY OF DISCUSSION ON EXTREME BEHAVIOR

Complex ICT systems are vulnerable to LHR events

30

To summarize, a typical complex adaptive ICT system is vulnerable to LHR events because single points of failure (SPFs) can take down the whole system and because local failure propagation can cause extreme global behavior with, perhaps, intolerable impact. The number or SPFs and the interplay between positive and negative feedback loops determine the frequency and impact of LHR events.

FURTHER READING



The two leftmost books are quite easy to read, while a solid math background is needed to understand the rightmost book. Nassim N. Taleb introduces and analyses LHR events (also called black and grey swans), while Dmitry Chernov and Didier Sornette provide 45 case studies to document how concealment of risk leads to LHR events.

LIMITS OF RISK ANALYSIS

(OR SHIT HAPPENS IN THE FOURTH QUADRANT)

This section discusses the limits of classical predictive risk analysis.

RISK IN COMPLEX ADAPTIVE ICT SYSTEMS

- We talk about risk when we do not know what will happen
- Risk means that more things can happen that will happen

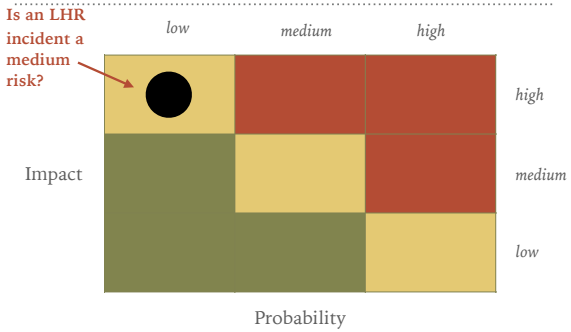
RISK ANALYSIS

- A classical risk analysis **predicts** incidents during a future time period by
  1. describing all possible incidents,
  2. estimating the probabilities that they will actually occur, and
  3. determining the incidents' impact on a group of stakeholders

34

This type of risk analysis is highly problematic when it is applied to complex adaptive ICT systems. First, risk analysts are not able to describe all possible LHR incidents in such systems. Second, analysts are not able to estimate the probabilities of identified LHR incidents.

CLASSICAL RISK MATRIX



35

A possible LHR event can be that a national banking system goes down for a week (see previous example). Since such an event has a low probability and a high impact, it is a medium risk according to the risk matrix. Obviously, a week of downtime is intolerable in any advanced society. Something is clearly wrong with this risk matrix.

LIMITS OF CLASSICAL RISK ANALYSIS

- A classical risk matrix is created with the implicit assumption that stochastic events in a system have a probability distribution with a thin left tail
  - **LHR incidents (outliers) are ignored**
- ***Observation*** Classical risk analysis severely underestimate the risk associated with complex adaptive ICT systems because LHR incidents dominate the impact on stakeholders

36

The observation that classical risk analysis severely underestimates the risk associated with today's ICT systems partly explains why we continue to build systems with fragility to LHR events. In fact, current risk analyses of ICT systems lead stakeholders to take more risk than they should because the stakeholders are confident that the risk is manageable.

TALEB'S FOUR QUADRANTS

Impact / Distribution	Only local impact	Global impact
Thin left tail	*1* Only limited local impact	*2* Global impact possible but tolerable
Thick left tail	*3* Large local impact possible, good risk management needed	*4* **Intolerable global impact is inevitable PREDICTIVE RISK ANALYSIS DOES NOT WORK**

37

Following Taleb, we divide complex adaptive ICT systems into four categories, or quadrants, depending on incurred financial costs and distribution of events.
We then determine the quadrants' robustness to LHR incidents. Note that risk management works in the first three quadrants, but not in the fourth.

AVOID THE 4TH QUADRANT

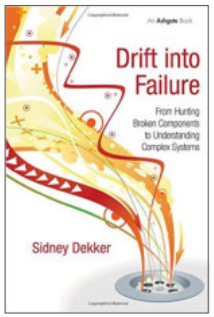
- We want to create systems in the first quadrant but may end up in the third quadrant
- The important thing is to avoid the fourth quadrant with its intolerable LHR incidents

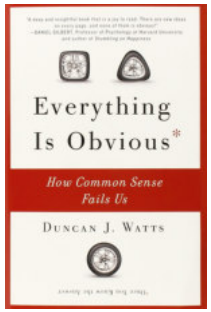
We need to develop and operate complex adaptive ICT systems that limit the impact of unforeseen incidents

38

`https://www.edge.org/conversation/nassim_nicholas_taleb-the-fourth-quadrant-a-map-of-the-limits-of-statistics`

FURTHER READING





39

Sidney Dekker explains why hidden risk in complex systems will eventually lead to serious failures. Duncan J. Watts explains why we believe we know much more about (economic) systems and the world in general than we actually do.

COMPLEXITY IS THE ENEMY

Subjective view

40

This section discusses the consequences of our inability to understand extreme behavior in complex adaptive systems.

COGNITIVE COMPLEXITY

- **Cognitive complexity** is the mental effort needed by a single individual, or a team, to understand a given functionality of a system
- Cognitive complexity is subjective in the sense that it depends on an individual's energy, mood, skill set, and ability to concentrate
- ***Claim*** Large cognitive complexity seriously affects our ability to analyze incidents in complex systems, leading to oversimplified explanations and downright wrong conclusions

41

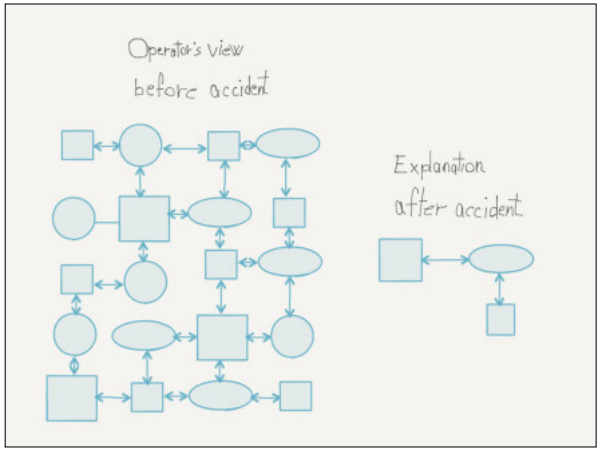
FLAWED INVESTIGATIONS

- Investigators of incidents in complex systems tend to
  - ignore that many scenarios lead to the same incident
  - focus on explanations where the incident is the last in an ordered (linear) sequence of events
  - look only for well-defined root causes
  - view technical systems as reliable and humans as unreliable
- An investigation often concludes that an incident was due to "human error," but says nothing about why individuals acted the way they did

42

HINDSIGHT BIAS

- The **hindsight bias** or the I knew-it-all-along effect is the tendency, after an incident occurred to
  - conclude that it was foreseeable,
  - find a single initiating event,
  - ignore all but one simple explanation, and
  - blame one or a few individuals for the incident



DECISION MAKING: LOCAL RATIONALITY

- Failures occur when the ability to understand and handle complexity breaks down
- While there is a view that technical systems are reliable and humans are unreliable, technical systems fail on a regular basis
  - algorithms are brittle and fail on unanticipated inputs
  - hardware fail all the time in large systems
- ***Observation*** Humans are reliable in general but have to make decisions with limited information and understanding when the complexity is high

Local rationality is the idea that during the events leading up to accidents, people are acting in a way that makes sense to them at the time. All of their knowledge, training, experience, organizational culture, and input from the environment combine to influence the decisions people make and the actions they take.

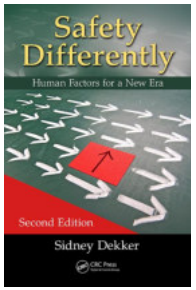
"HUMAN ERROR" IS JUST A CONVENIENT LABEL

- Investigator of incidents in complex systems usually blame
  - the technical system,
  - management, or
  - the operators
- While it is both most convenient and least expensive to blame operators, it often hides the real reasons for incidents, namely
  - many strong dependencies leading to high cognitive complexity

THE ENEMY

- The enemy is not unreliable humans but large complexity that makes it too hard for humans to understand a system and make good decisions
- ***Observation*** Blaming a small group of operators for a serious incident, without really understanding why they acted the way they did, prevents us from learning how to design and operate systems
  - erroneous actions and assessment are symptoms, not causes

FURTHER READING



This book provides a critique of the way we analyze accidents and suggest a better way forward.

FROM FRAGILE TO ANTI-FRAGILE SYSTEMS

49

Here, we discuss how to build ICT systems that do not depend on highly accurate risk analyses to operate as desired.

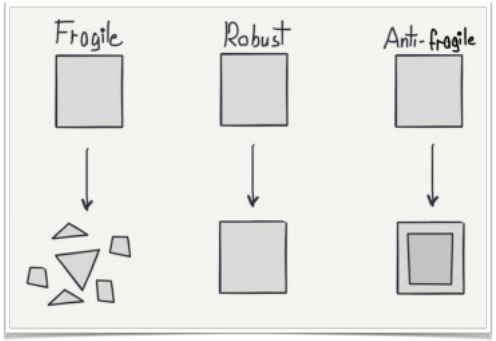
FRAGILE, ROBUST, AND ANTIFRAGILE SYSTEMS

- A property of a complex adaptive system is
  - **fragile** if it is easily damaged by internal or external perturbations,
  - **robust** if it can withstand perturbations (up to a point), and
  - **antifragile** if the system learns from incidents how to make the property increasingly robust over time

50

As an example, a system can be antifragile to targeted attacks, malware, downtime, scaling, real-time response, and fraud.

TYPES OF SYSTEMS



51

The human immune system is a prime example of an antifragile system. Another example is muscles. It could also be argued that the brain is antifragile.

FRAGILE

Handle with care

52

If the content of the box is fragile, then the box must be handled with care.

ROBUST

53

If the context of the box is robust, then we need not write anything on the box because we do not care how it is handled (up to a point).

ANTIFRAGILE

Please mishandle

54

If the content of the box is anti-fragile, then stressors will make it stronger. In fact, the content needs stressor to maintain its antifragility.

FRAGILE SYSTEMS

- When a ***fragile*** system fails, its fragility is not blamed; instead, bad risk analysis is said to be the cause
- ***Observation*** The real problem is not bad risk analysis, but that the fragile system was created in the first place

55

ROBUST SYSTEMS

- We need to create systems that are much less dependent on our very limited ability to predict LHR incidents
- ***Observation*** It is not enough to create a system that is robust to known incidents because it will become fragile over time as the system and its environment change

56

ANTIFRAGILE ICT SYSTEMS

- Antifragile systems fail locally with limited impact and prevent failure propagation. The systems
  1. avoid silent failures,
  2. detect failures early, and
  3. learn from failures how to better handle future incidents

57

No antifragile system can scale forever because the growing complexity will introduce unwanted positive feedback loops that increase the fragility of the system, The increase in hidden risk will eventually materialize as an intolerable incident.

ANTIFRAGILITY TO CLASSES OF INCIDENTS

- No ICT system is antifragile to all possible types of incidents
- Our approach is to develop and operate systems that are antifragile to particular classes of incidents
- These classes can be defined in different ways by focusing on
  - results of incidents, e.g., downtime
  - type of attacks, e.g., malware
  - type of threats, e.g., Advanced Persistent Threats

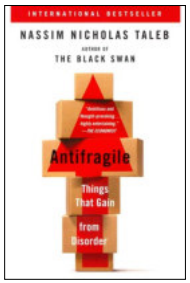
58

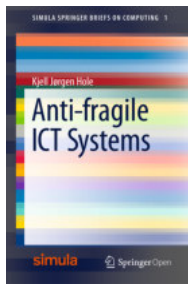
Fragile Robust Antifragile

59

Fragility, robustness, and anti-fragility are degrees on a spectrum. A system is only robust or anti-fragile up to a point. A fragile agent is one who must control the environment to maintain its normal condition. A slight shift in the environment can result in devastating consequences. In contrast, the robust agent maintains its normal condition in response to changes in the environment. But an anti-fragile agent always maintains or improves its current condition as the environment changes, without any preordained sense of normality.
`http://blogs.lse.ac.uk/lsereviewofbooks/2013/02/23/book-review-antifragile-how-to-live-in-a-world-we-dont-understand/`

FURTHER READING





Download Kjell's free e-book from
`link.springer.com/book/10.1007/978-3-319-30070-2`

60

To fully understand the concept of antifragility, you should to read the book by Nassim N. Taleb. The book by Kjell Hole can also be downloaded from `https://againstfragile.com`

DESIGN AND OPERATIONAL PRINCIPLES

61

This section introduces design and operational principles for antifragile systems.

CORE PRINCIPLES

- To design and operate antifragile ICT systems, we first study
  - four design principles and
  - two operational principles
- The six principles are not new, but together they outline a novel way to develop and operate ICT systems
- ***Observation*** The principles' common goal is to mitigate tail risk, that is, to ensure that the impact PDF has a thin left tail

62

In computer science, design principles are often called design patterns

PRINCIPLES AND ANTI-PRINCIPLES

	Principles	Anti-principles
Design	separate processes	deployment monolith
	isolatable processes	inseparable
	diversity	uniformity
	redundancy	uniqueness
Operational	fail fast	fail slow
	skin in the game	no skin in the game

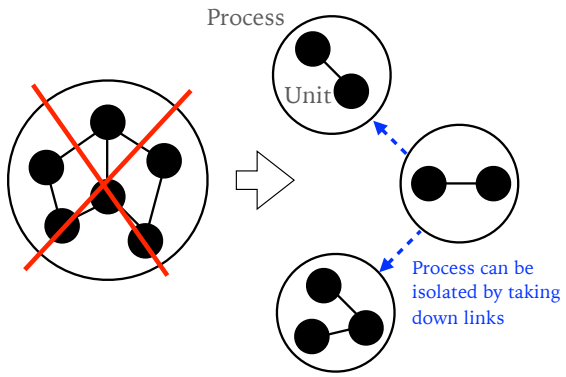
63

The table contains six principles and the corresponding anti-principles. To understand the importance of a principle, it is useful to study the anti-principle. In our case, the anti-principles can be used to detect fragility in systems.

DESIGN PRINCIPLES

- **Separate processes** A system must consist of separate processes running on multiple physical machines
  - first step to avoid failure propagation
- **Isolatable processes** A process must be isolated from other processes when it develops problems
  - second step to avoid failure propagation

64



65

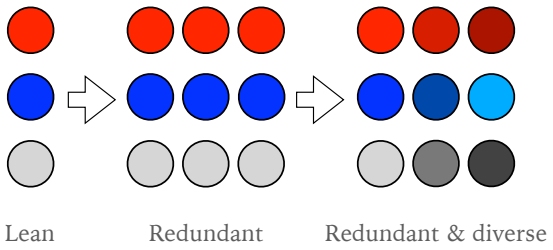
Virtual machines are often used to implement processes. Links that can be disconnected and reconnected without creating problems for the rest of the system are sometimes referred to as **weak links**.

MORE DESIGN PRINCIPLES

- **Redundancy** Use multiple identical copies of processes
  - limits the impact of process failure
- **Diversity** use differently designed and implemented processes
  - makes it less likely that multiple processes fail at the same time

66

DISTRIBUTED SYSTEMS



Processes with red, blue, and gray colors in the lean and redundant systems have different functionality. Processes with different shades of the same color in the redundant & diverse system have different design and/or implementation, but the same functionality.

FAIL FAST OPERATIONAL PRINCIPLE

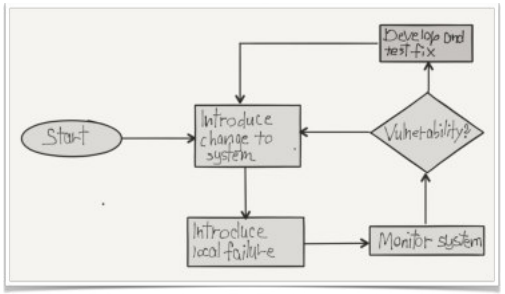
- It is necessary to discover failures early to limit their consequences and learn how to avoid the same, or similar, failures in the future
  - remember that we are not able to predict all future incidents

What is fragile should break early, while it's still small [Nassim Taleb, Black Swan]. Nothing should ever become too big to fail.

REAL-TIME MONITORING

- ***Observation*** Accurate real-time monitoring of behavior at different system levels is crucial to detect problems and learn how to improve the system
  - rather than waiting for failures to occur, Netflix injects artificial failures into their production system to speed up the learning process

LEARNING FROM INJECTED FAILURES



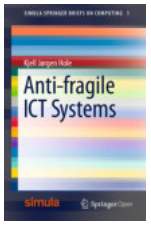
SKIN IN THE GAME

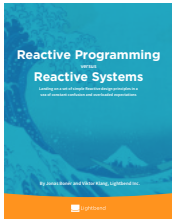
- A person with “skin in the game” has something to lose like ownership, money, property, or respect
- Major stakeholders that benefit from an ICT system should share at least some of the downside when the system misbehaves

SOFTWARE DEVELOPERS WITH SKIN IN THE GAME

- A team of software developers creating a system should be responsible for mitigating problems with their own code and make sure that the system runs without serious hiccups
- Developer teams should have operational responsibilities not to punish them when things go wrong, but to make sure the teams learn from their own mistakes how to maintain and improve the system

FURTHER READING





ANTIFRAGILE MICROSERVICE SYSTEMS

NETFLIX

- Netflix has developed a distributed system of **microservices** in the Amazon Web Services (AWS) cloud for streaming movies and TV series
- We study how Netflix has utilized the six design and operational principles to create a system that is antifragile to downtime

MICROSERVICES

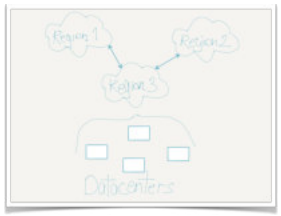
- A microservice does one thing well
- Manage its own data
- Runs as a separate process
- Fast shutdown and startup times
- A single developer can quite easily understand the functionality of a service
- Services can be changed independently of each other
  - can be written in different languages

76

Microservices avoids much of the maintenance problems associated with tightly integrated legacy systems because it easy to update individual services, including changing the technologies they are based on.

CLOUD

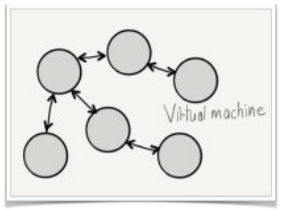
- A cloud infrastructure is divided into **regions** situated in different parts of the world
- Each region consists of multiple **zones** or data centers
- Each data center has a large number of servers and storage units



77

MODULARITY VIA MICROSERVICES

- Virtual machines run well-defined and self-contained services
- A microservice solution may have many hundred services

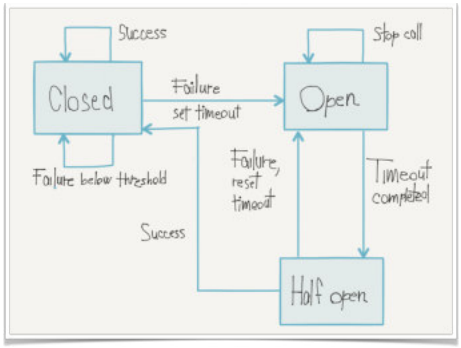


78

ISOLATION WITH CIRCUIT BREAKERS

- Any service is called via a circuit breaker
- A circuit opens when it detects problems with a service
- The circuit breaker provides a default response
- It closes when the problem is fixed
  - has logic to test if the problem is gone

GENERIC CIRCUIT BREAKER



REDUNDANCY PROVIDED BY THE CLOUD

- The cloud supports redundancy at the virtual machine (VM), zone, and region layers

VM REDUNDANCY (1)

Redundant services with timeout and failover

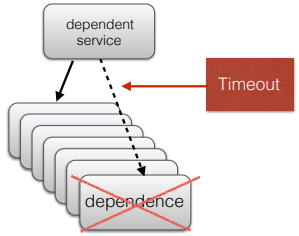


Fig. from Netflix

82

When a dependency times out, another is queried. Failure of an instance is often due to power outage in the hosting rack, a disk failure, or a network partition that cuts off access.

VM REDUNDANCY (2)

Timeout with fallback default response used when *all* instances are affected

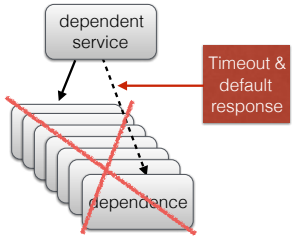


Fig. from Netflix

83

When there is a software bug or network failure, all instances are affected and it is necessary to use a (non-personalized) default response to contain the error. A careful analysis is needed to determine the appropriate response.

MULTIPLE ZONES

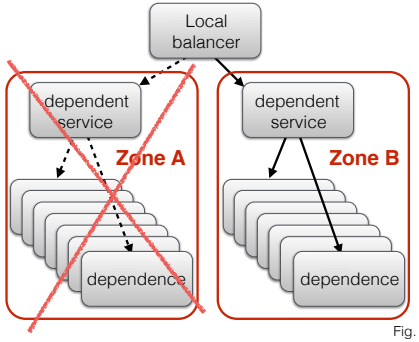


Fig. from Netflix

84

Netflix uses three zones in each region to limit the consequences of firmware failures, certain serious software bugs, power failures, and severe network failures. The zones correspond to different data centers. Note that the load balancer is a single point of failure.

MULTIPLE REGIONS

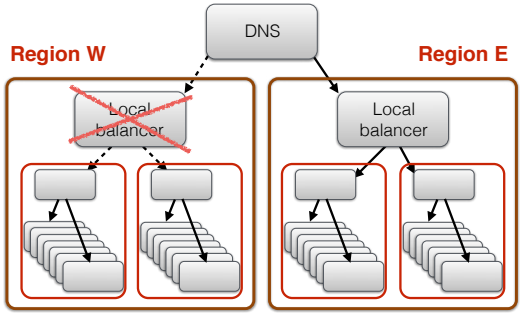


Fig. from Netflix

85

DNS splits traffic load in two halves. To handle infrastructure failures, Netflix uses three regions and switch users to a new region when needed. Failures of a whole region are caused by configuration issues, bugs in infrastructure code, bugs in application code, and failures in load balancer.
`http://techblog.netflix.com/2013/12/active-active-for-multi-regional.html`

DIVERSITY PROVIDED BY THE CLOUD

- The cloud supports diversity at the VM layer
- Since a web-scale solution supports users all over the world, there is no good time to take down the system and upgrade its software
- An alternative is to introduce new code by keeping both old and new code running and switch user requests to new code

86

SIMPLE CANARY PUSH

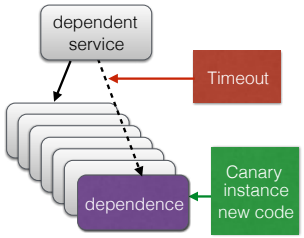


Fig. from Netflix

87

A “canary” is a new version of a service. The stability of a “canary” cannot be fully evaluated before it gets a heavy traffic load in the production system. A few copies of the “canary” is tested in the production system.

RED/BLACK DEPLOYMENT

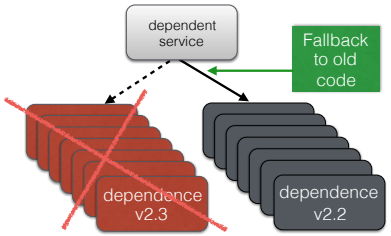


Fig. from Netflix

88

The default instance groupings that Chaos Monkey uses for selection is Amazon's Auto Scaling Group (ASG). We use enough copies of new code in an ASG to carry the load. Keep the old code to ensure that we can handle peak load if there is a problem with the new code.

STANDBY BLUE SYSTEM

- Software error in both red and black deployment
- Blue system delivers a minimal solution
- Used when all resent versions of the code fail

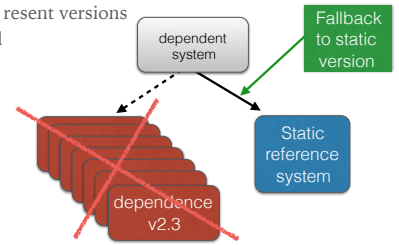


Fig. from Netflix

89

Several versions of the code may contain a “time bomb” that only goes off after a long period. There could also be a problem with the data causing several versions of the code to fail. Since the blue system is non-adaptive or static, it is easier to scale than the regular code.

FAIL FAST USING MONKEYS

- Netflix has created a collection of tools called the **Simian army** to deliberately introduce failures in their production system
  - **Chaos Monkey** disables randomly selected virtual machines
  - **Chaos Gorilla** simulates network partitions and total zone failures
  - **Chaos Kong** simulates region failures

To avoid intolerable impact, only introduce failures in systems that satisfy the four design principles

90

Within an ASG, Chaos Monkey will select an instance at random and terminate it. The ASG should detect the instance termination and automatically bring up a new instance.

LATENCY MONKEY

Latency Monkey tests what happens when the delay becomes too long

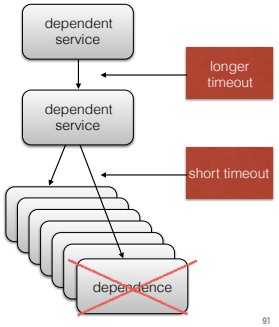
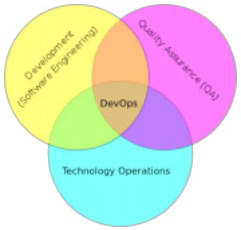


Fig. from Netflix

91

The shutdown of a low-level dependency can lead to a longer timeout at a higher layer, causing a cascading failure. There is no simple answer to this multi-level dependency problem, each case must be carefully studied.

SKIN IN THE GAME WITH DEVOPS



92

DevOps: combination of development and operations. See `http://en.wikipedia.org/wiki/DevOps`

DEVOPS

- The DevOps methodology combines software development and IT operations
- DevOps is a response to the interdependence of development and operations
- Breaks down silos of development, quality assurance, and operations
- Software teams develop, run, and update their own code
- DevOps team utilises iterative development with continuous delivery

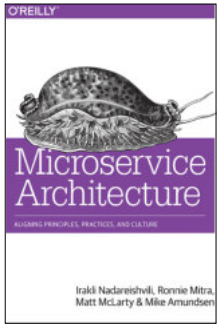
93

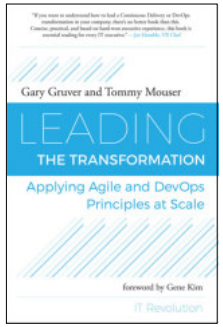
DEVOPS FACILITATES ANTIFRAGILITY

- Failures occur in a production system traditionally under the control of IT operations
- Software developers must fix the problems because IT operations lacks the needed programming skills
- If software development and IT operations are combined, then it is possible to learn from failures and introduce countermeasures much faster than before

94

FURTHER READING





95

There exist many books on microservice systems and DevOps. Here are two good examples.

SUMMARY

96

ANTIFRAGILE SYSTEMS ARE:

- Highly distributed systems of isolatable processes with much redundancy and diversity
- They avoid silent failures and fail fast with only local impact
- They learn from small-impact failures how to become more robust to future incidents

Antifragile systems are needed because we are very bad at predicting rare incidents with huge negative impact

97

Antifragility is all about continuous improvement. Stakeholders, especially software developers and system operators, must continue to improve a system over time to make it more robust to a changing environment. The introduction of artificial failures and the use if sophisticated system monitoring are crucial to achieve antifragility.

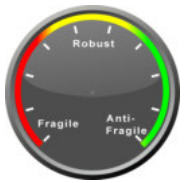
RESEARCH QUESTIONS

- How do we design antifragile systems from scratch?
  - How do we limit a system's cognitive complexity?
- What are the central design and operational patterns?
- How should we detect anomalies?
- How should we implement antifragile systems?
  - Erlang
  - Java/Scala & Akka

98

Since the concept of antifragile ICT systems is quite new (first discussed around 2012), there are many open research questions. Today, only a few researchers work in this area.

THANKS FOR LISTENING



99