RETHINKING CYBER SECURITY

Eugene H. Spafford
Purdue University CERIAS
http://ceri.as

WHAT IS SECURITY?

- If we talk about a system being "secure" what do we really mean?
- If we talk about "security features," what are they?

CYBER SECURITY

Let’s start with an intuitive definition: a system is secure if it is protected against all forms of threat.

This is what most current commercial entities claim they do: make your systems secure.

CYBER SECURITY

Random hackers?

Check!

(Well, usually)

CYBER SECURITY

Malware?

Probably.

CYBER SECURITY

Nation State Hackers?

Probably not.

CYBER SECURITY

UFO Invasion?

What? No!

CYBER SECURITY

Extinction Event Meteor Impact

Definitely not.

CYBER SECURITY

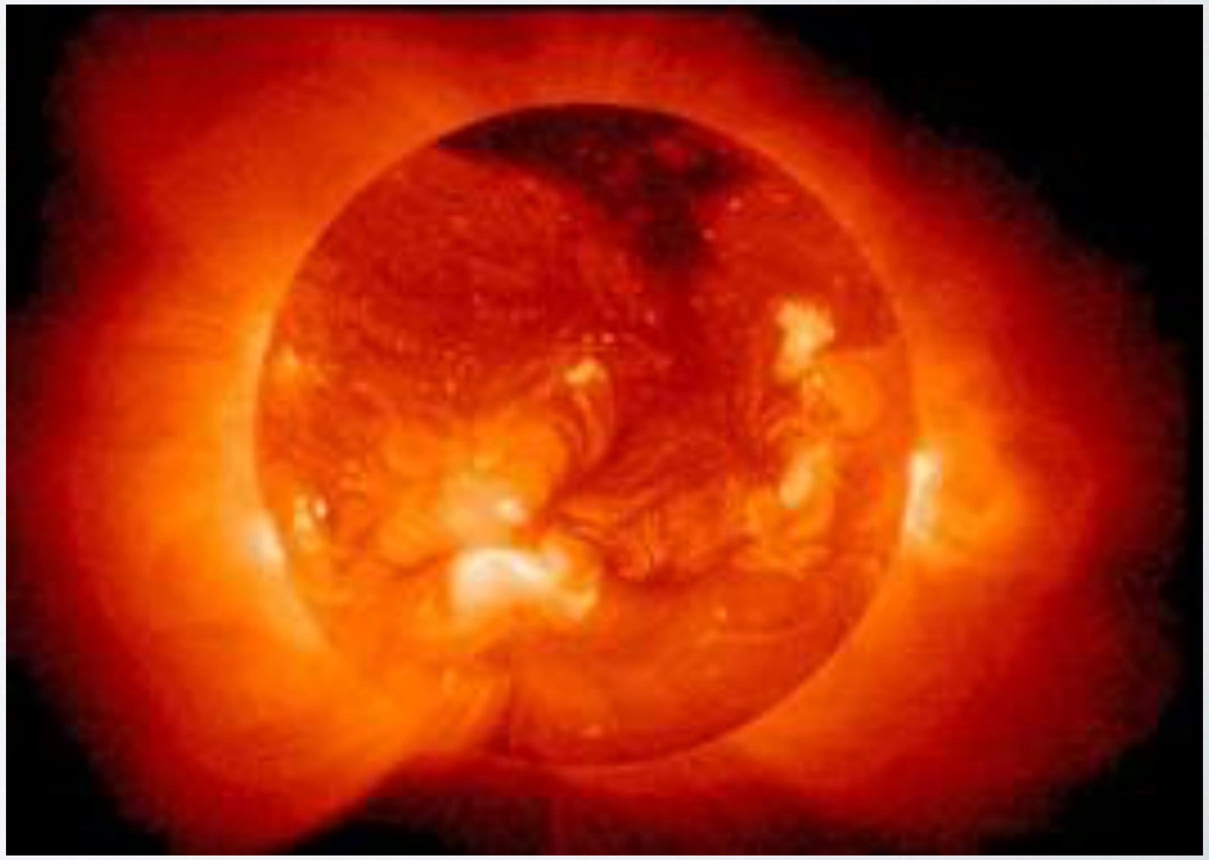
Maybe if we set up colonies on Mars and give them backup copies?

(In "the cloud" to a greater degree)

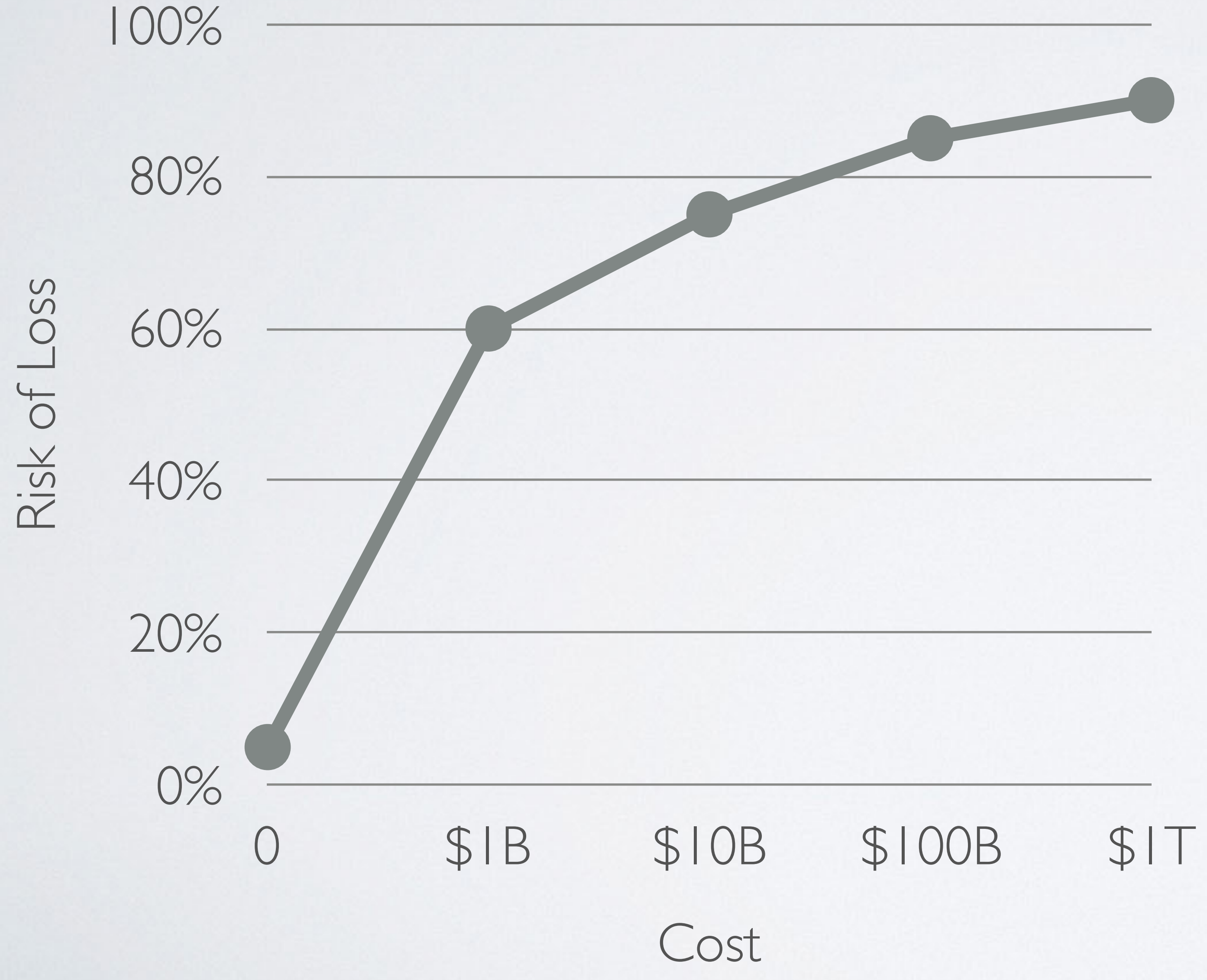
CYBER SECURITY

Maybe if we set up colonies on Mars and gave them backup copies?

No, eventual death of the Sun will mean end of the inner planets.



CYBER SECURITY



- As a definition, maybe that isn't helpful — we can't ever achieve it.
- Actually, this exposes an issue: security is, at its heart, an economics issue — managing risk.

IN REALITY…

Absolute security is unattainable. It is also dependent on context and resources.

- Robert Courtney articulated this with his 3 laws back in the 1970s:
  - Nothing useful can be said about the security of a mechanism except in the context of a specific application and environment.
  - Never spend more mitigating a risk than tolerating it will cost you.
  - There are management solutions to technical problems but no technical solutions to management problems.

ANOTHER ATTEMPT

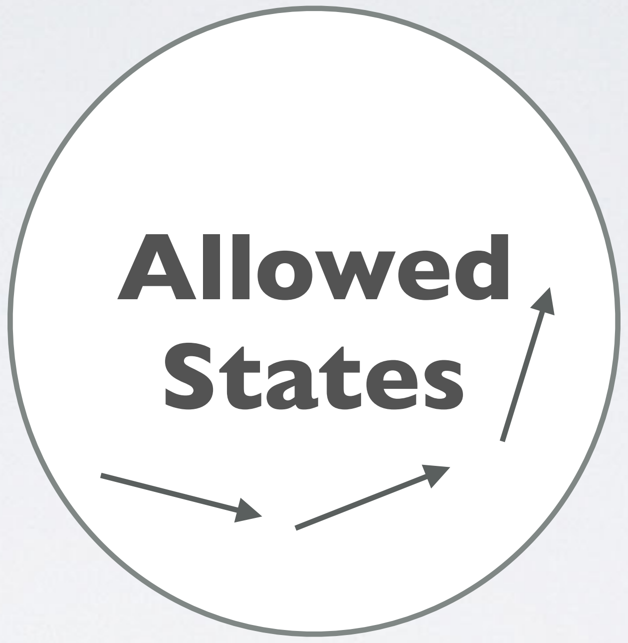
Let’s approach this as a problem of software and hardware design. Can we do a better job?

Initial research in the 1970s and 1980s looked at system state.

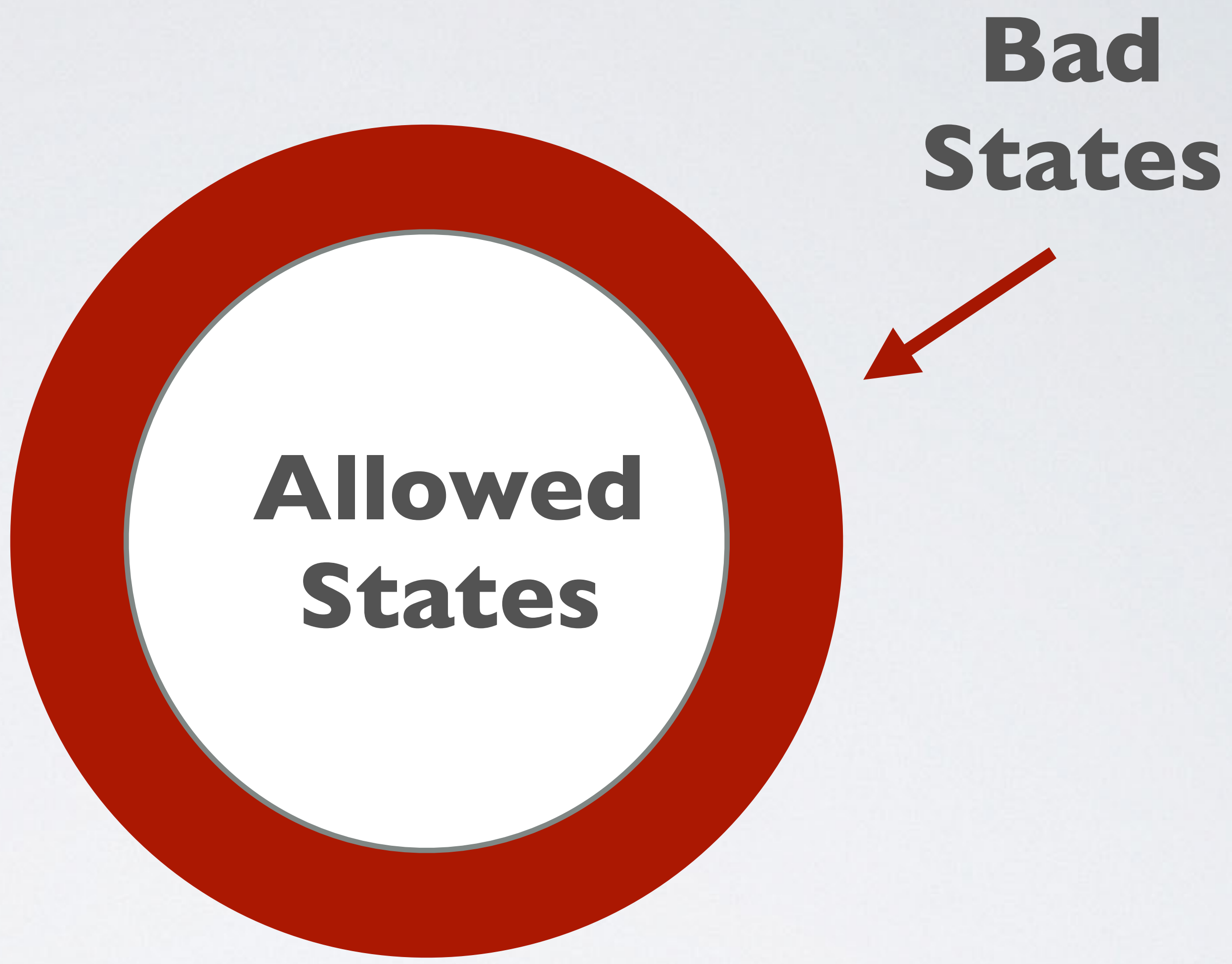
Allowed
States

There are a set of states that are known to be "okay" or "safe."

As a system executes, it changes state.

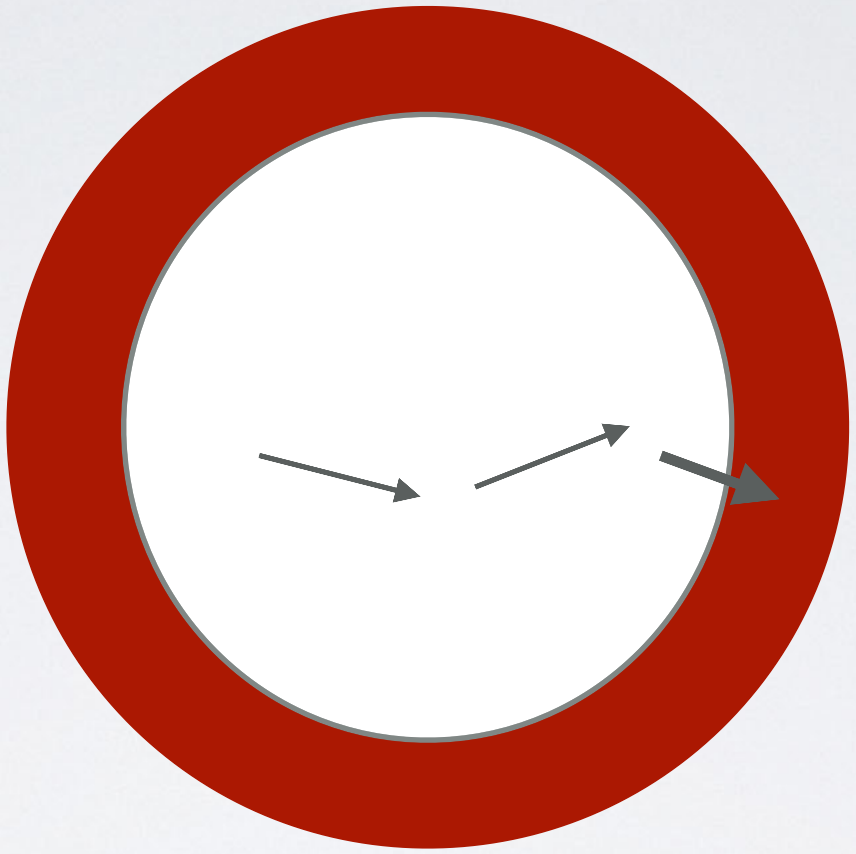


Each valid operation results in a state of the system that is also defined to be "okay."



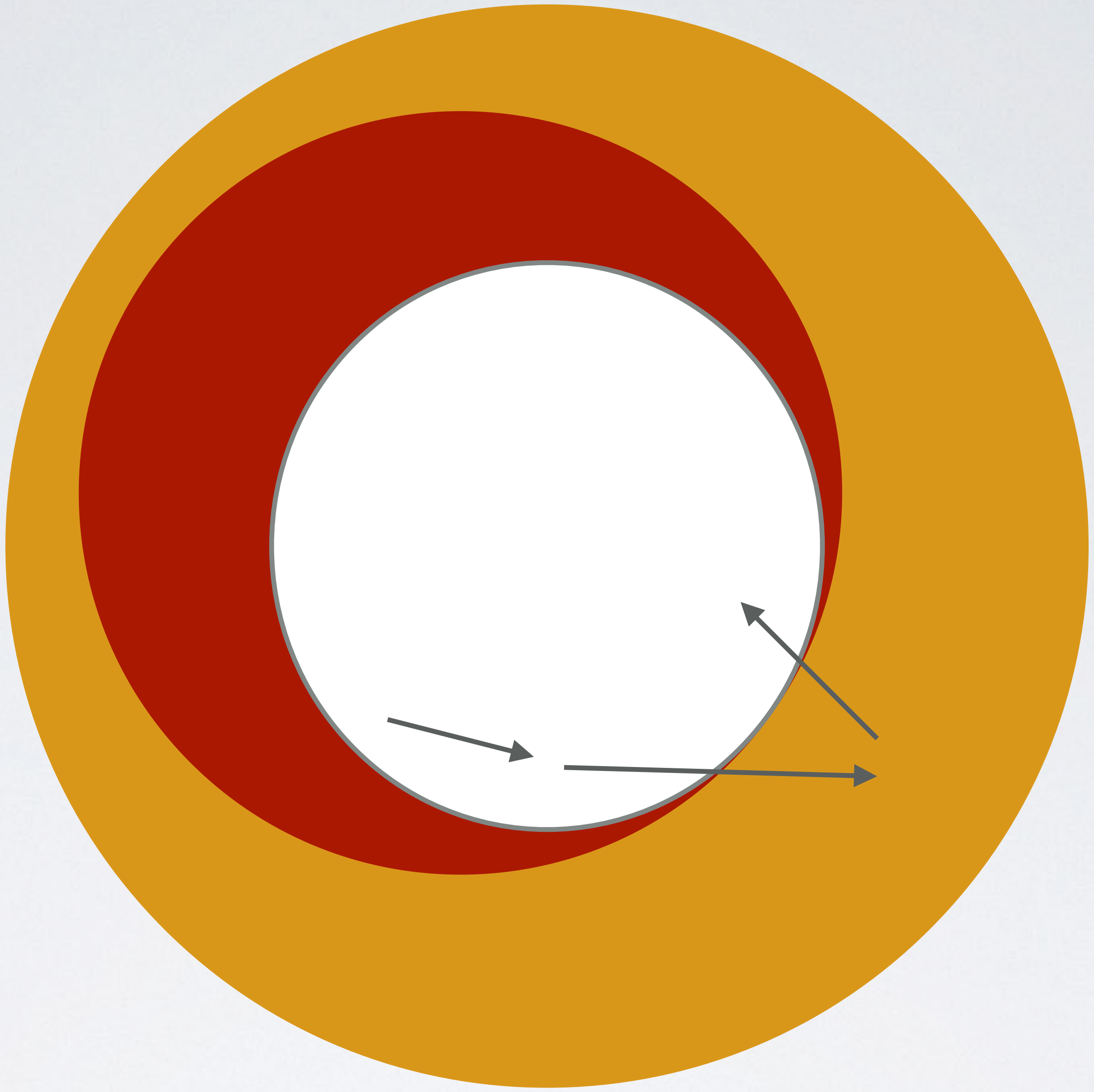
We also have "bad" states. We don't want these to occur.

We don't want execution to enter "bad" states.
We especially don't want to remain in them.



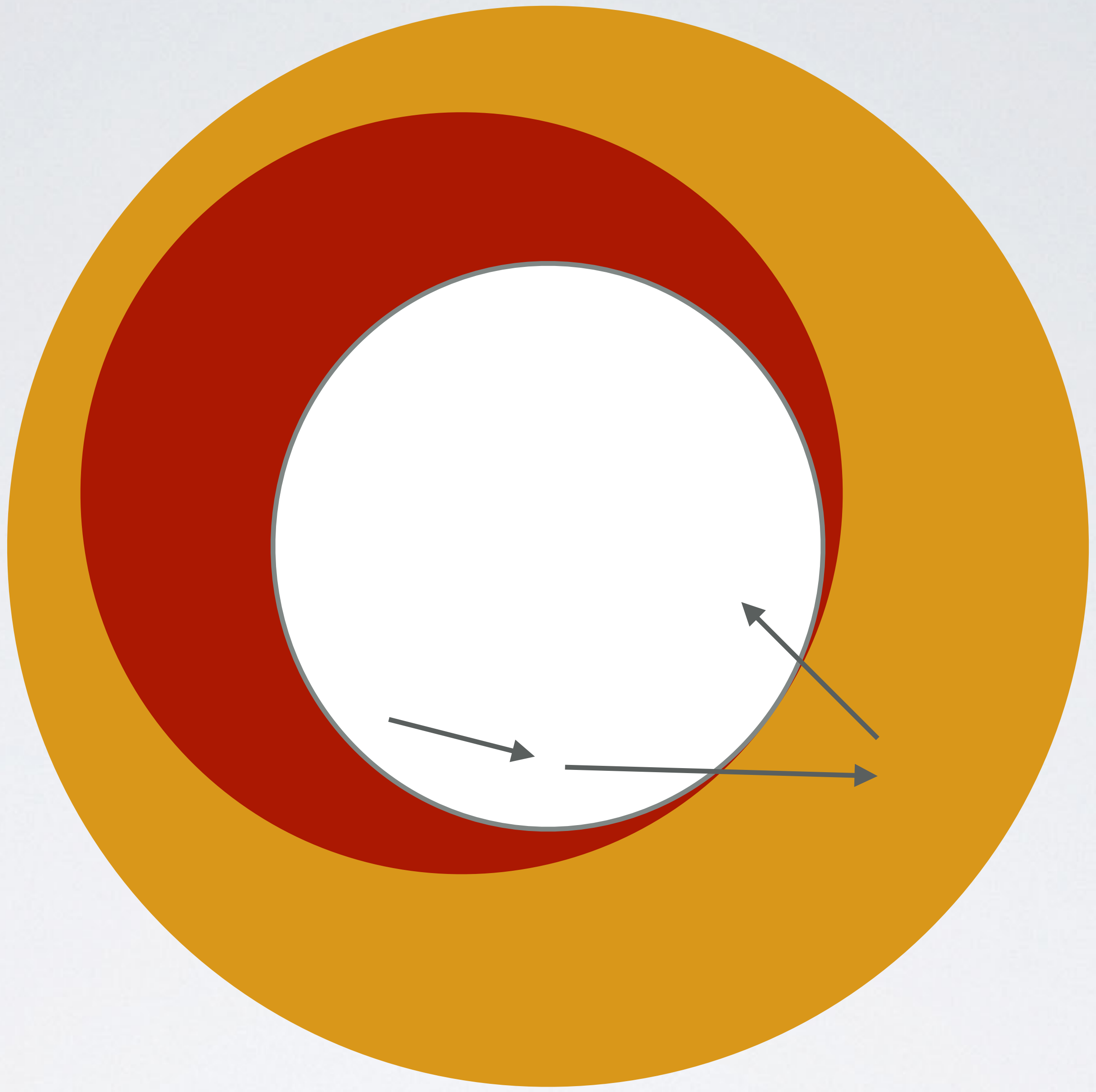
- This notion of "allowed states" is a match to the concept of "system requirements" in software engineering. Next step execution coresponds to "system specifications."

- Execution of a state not in the specification is a "fault" that can result in a "failure." A failure in a protected system is a security failure.



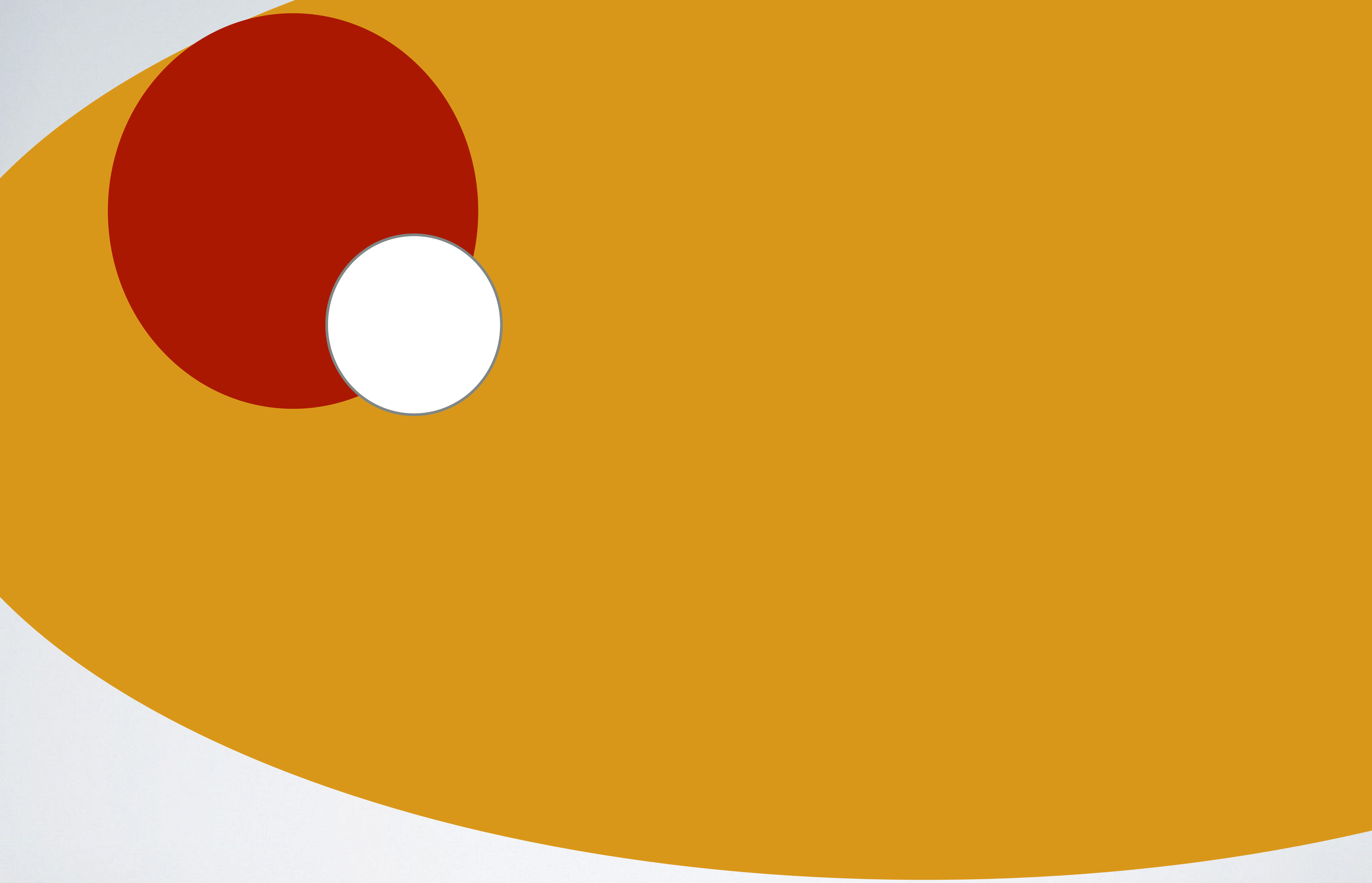
We also have "undefined" states. These aren't specified.

Entering undefined states is problematic. This may lead to a fault, or it could lead back into a defined state.



Undefined states might not be “bad” states.

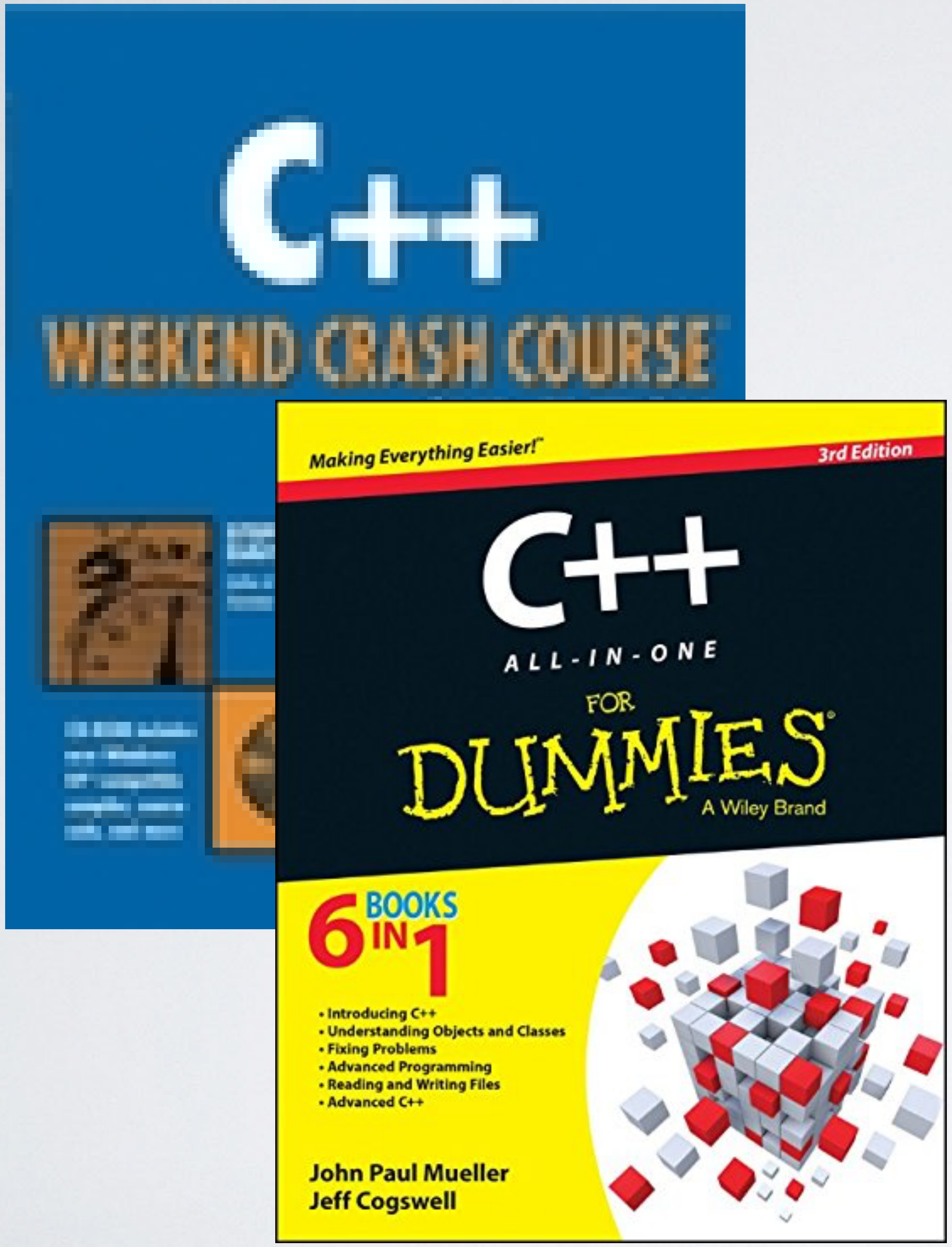
They might even lead back to “okay” states.
Because they are undefined, we do not know.



What the typical state-space probably really looks like

- Most software today operates in the "undefined" state space because we have never defined its proper behavior.
- We may have general requirements, but no completely-defined requirements, and definitely no specifications.
- Formal specifications are time-consuming and expensive. They also require expertise to define, and to build software to match.

INDUSTRY PRACTICE



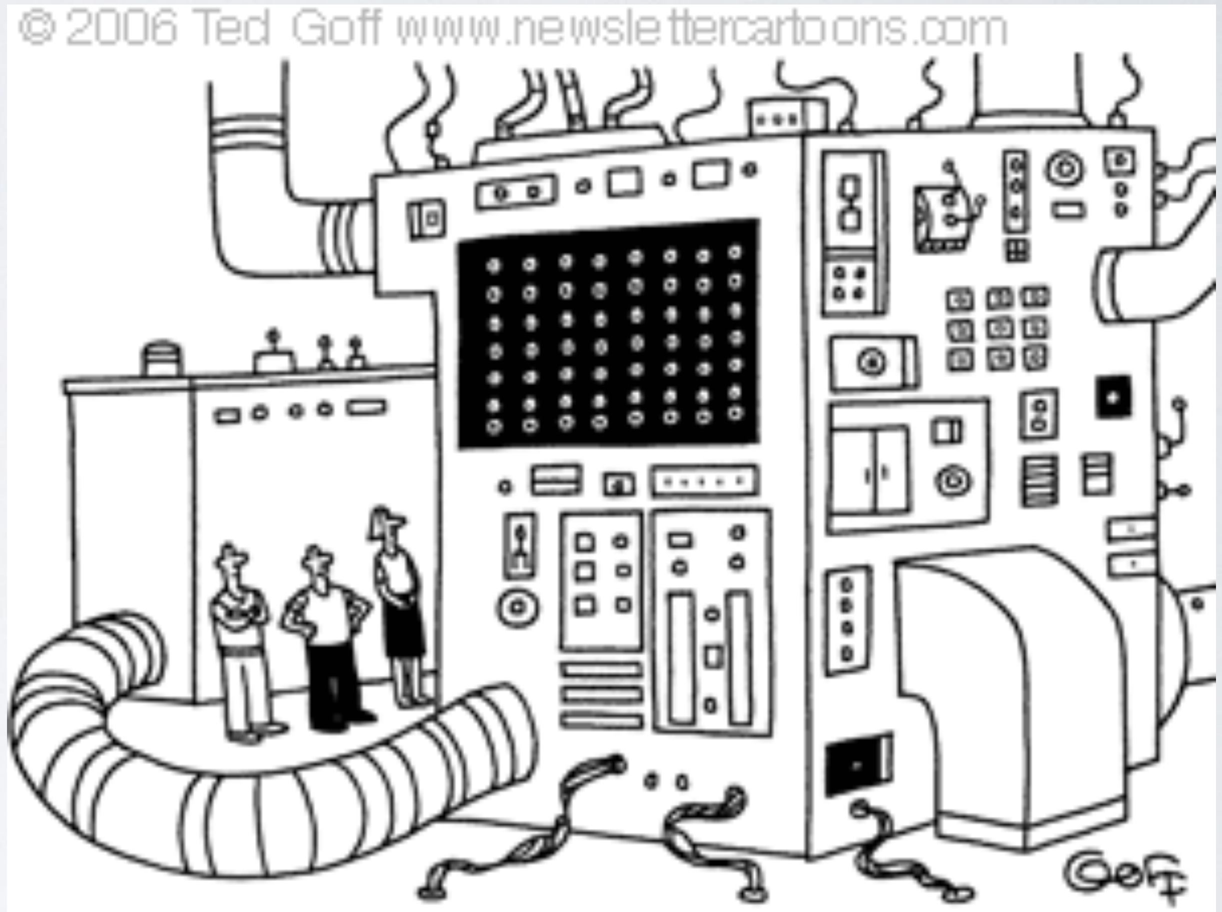
The writers got it in *Jurassic Park*

Minimum training

THE CONSEQUENCE OF "DESIGN"

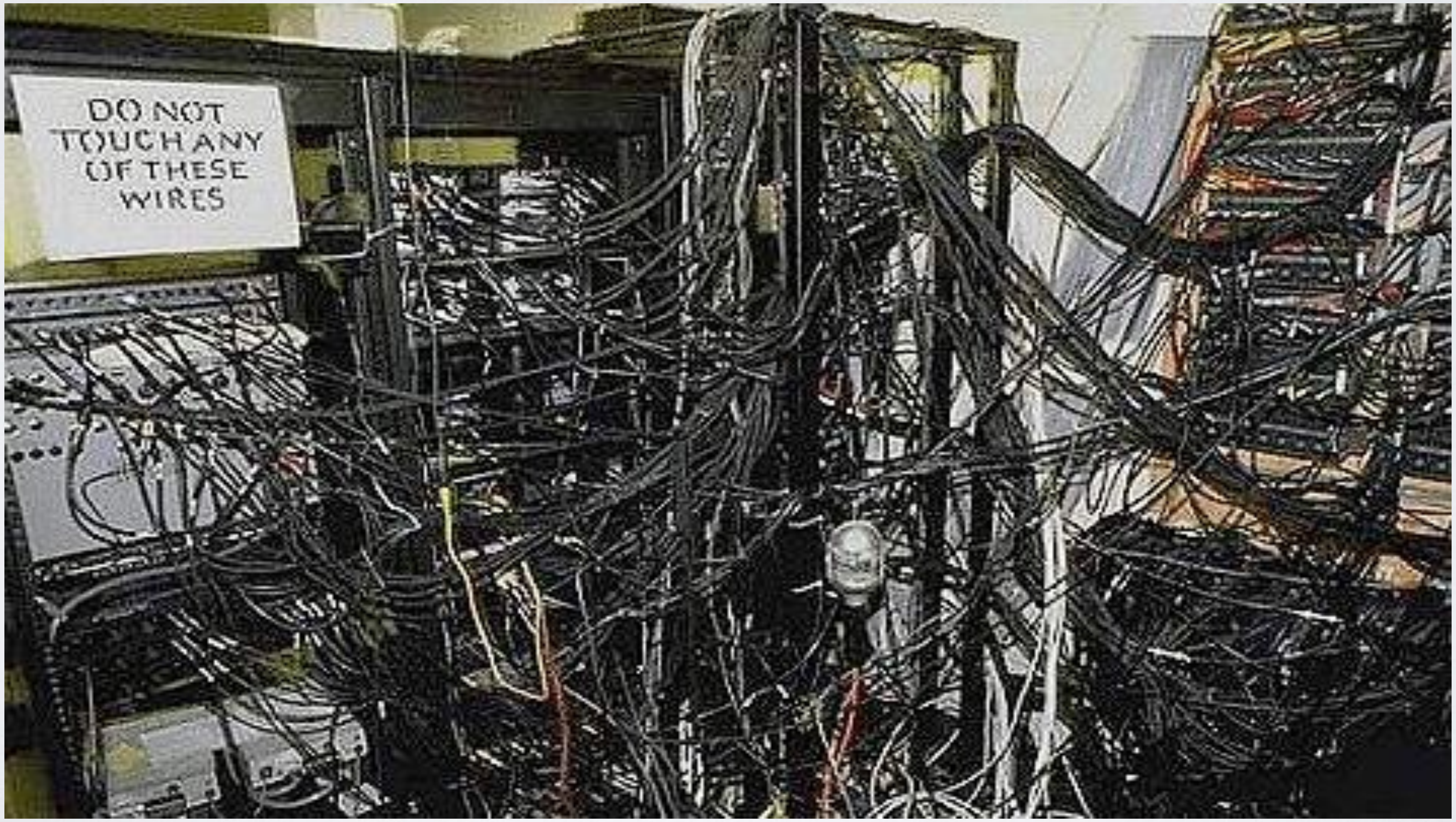
A program that has not been specified cannot be incorrect; it can only be surprising.

<u>Proving a Computer System Secure</u>, W. D. Young, W.E. Boebert and R.Y. Kain, The Scientific Honeyweller (July, 1985), vol. 6, no. 2, pp. 18-27.

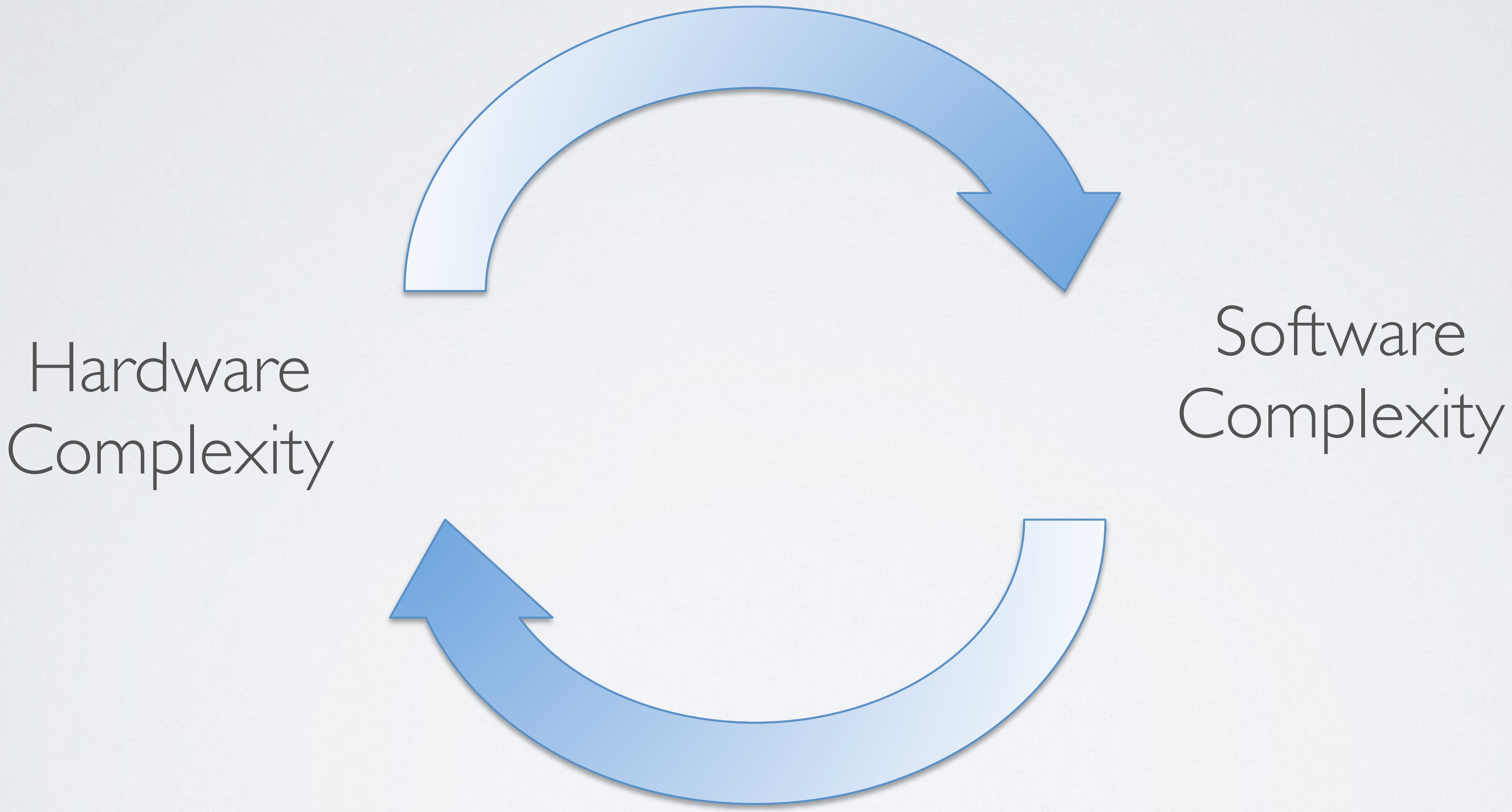


"It was just going to be a laser printer before we started adding features."

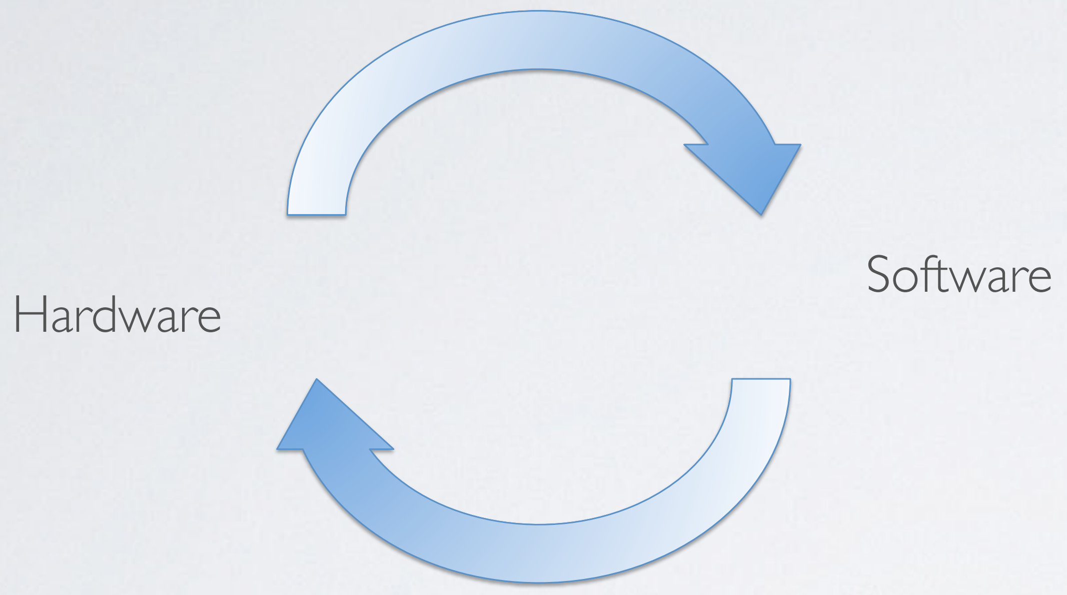
METAPHORS FOR CURRENT SOFTWARE



BAD FEEDBACK



BAD FEEDBACK

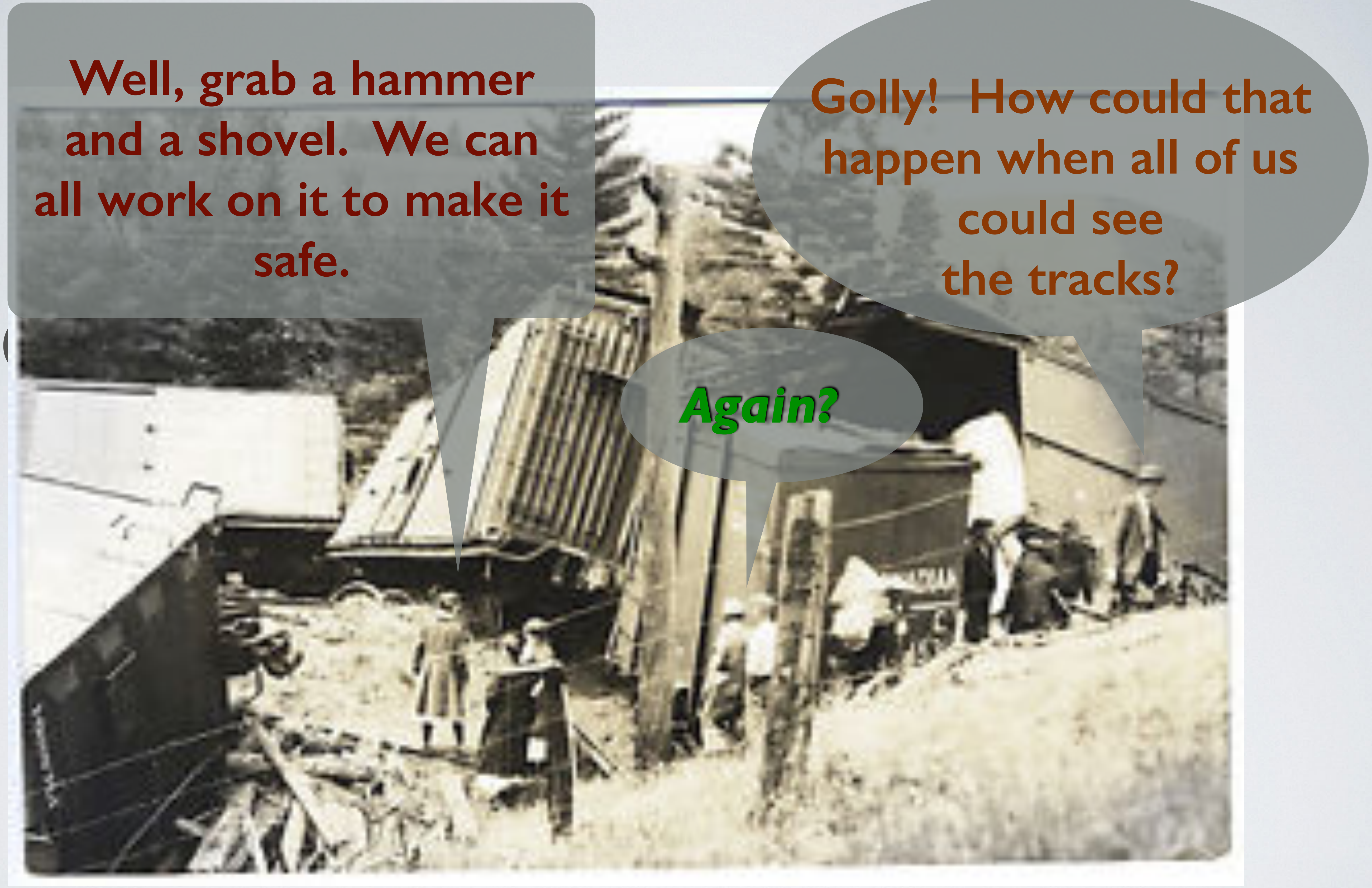


We design and invest in new hardware because our software is too slow.

We add to the software because we now have capacity to execute new features.

Rinse, repeat.

We stick with the same basic systems because of sunk costs.


Well, grab a hammer and a shovel. We can all work on it to make it safe.
Again?
Golly! How could that happen when all of us could see the tracks?

SO, NEXT BEST: TRUST

- “believe in the reliability, truth, ability, or strength of”
- “allow someone to have, use, or look after something of importance or value with confidence”
- “commit something to the safekeeping of”
- “place reliance on (luck, fate, or something else over which one has little control)”

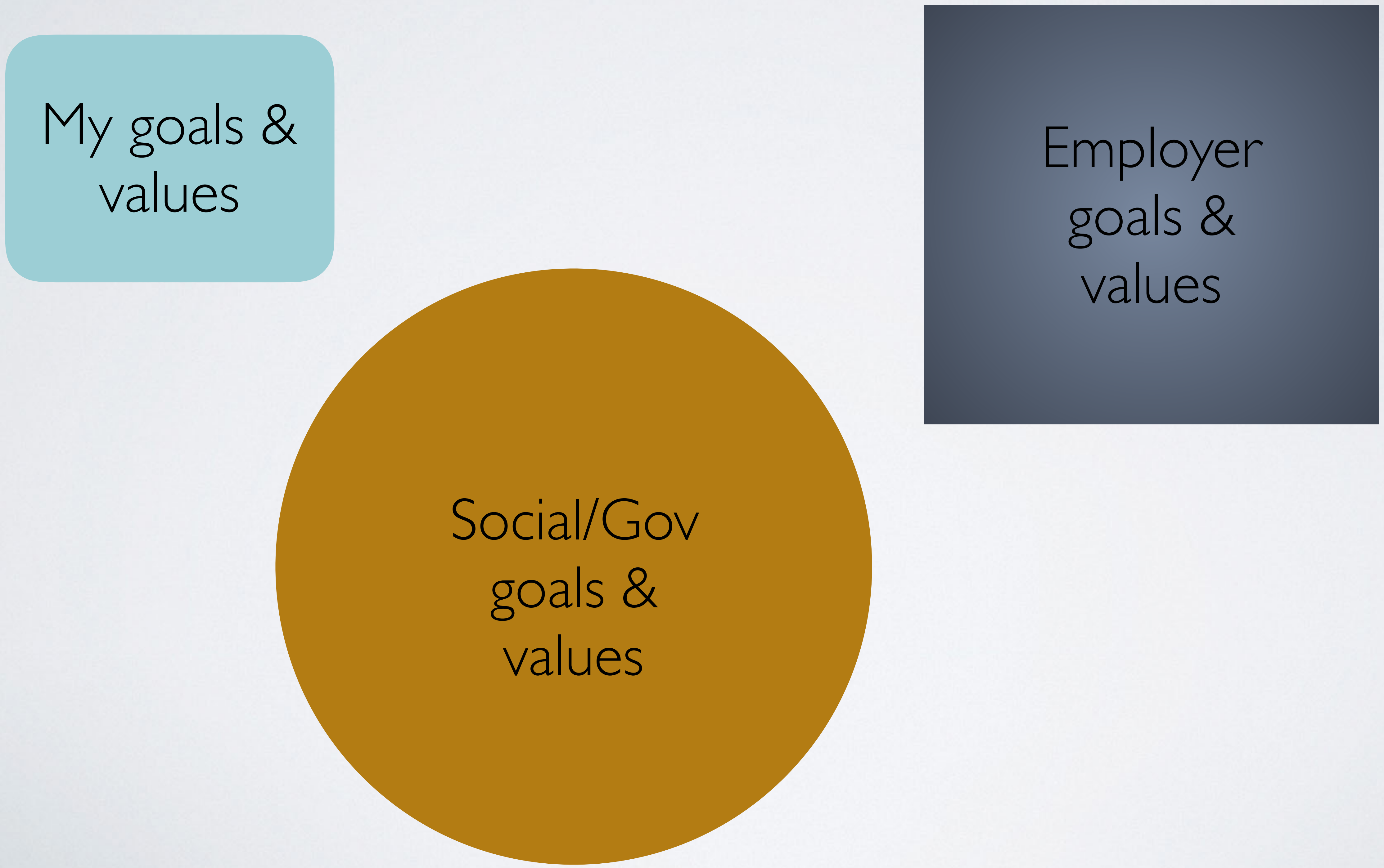
CYBER… WHAT?

- Cyber security is the science and practice of protecting information and information processing components from misuse during their design, creation, transmission, storage, transformation, use, and disposal.

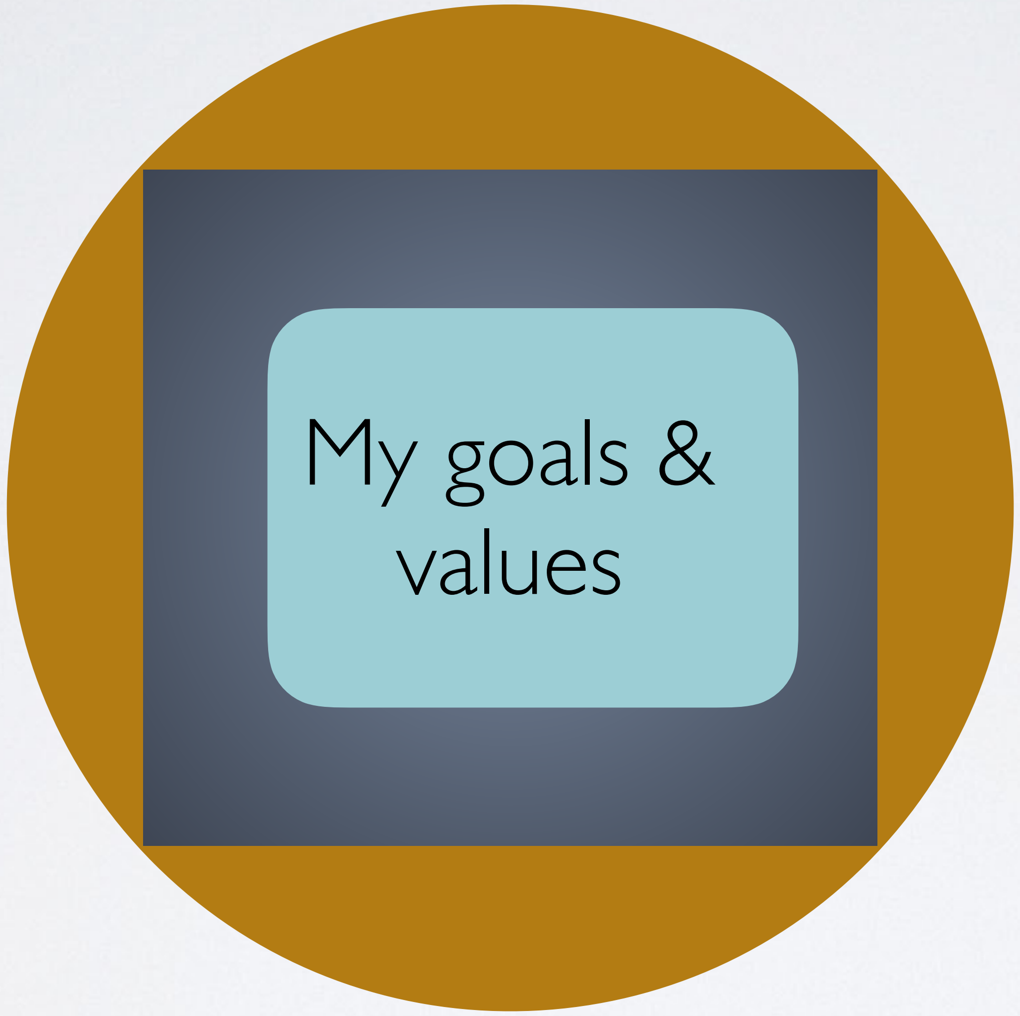
- Information assurance is the science and practice of increasing our confidence (trust) in the information security of a system.

- We need to use these two together.

FIRST ELEMENT: TRUST ALIGNMENT

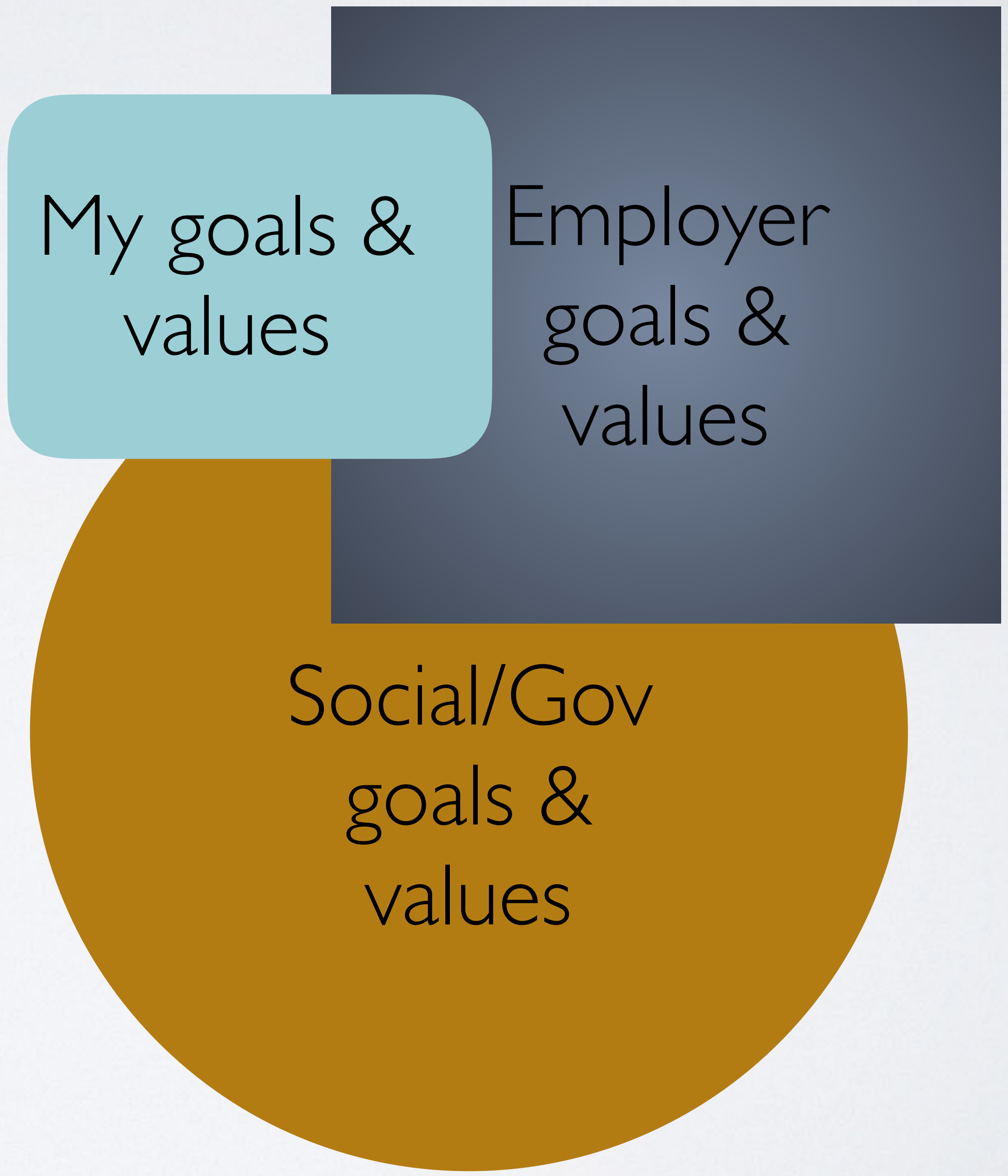


IDEAL TRUST ALIGNMENT



DYSFUNCTIONAL TRUST ALIGNMENT

Whose trust do we support?



COMPOUNDED TRUST

What are the limits of trust?

Supply chain…

Perhaps we can define tunable attributes — decompose security & trust?

Lord Kelvin (William Thompson) wrote:

"When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science."

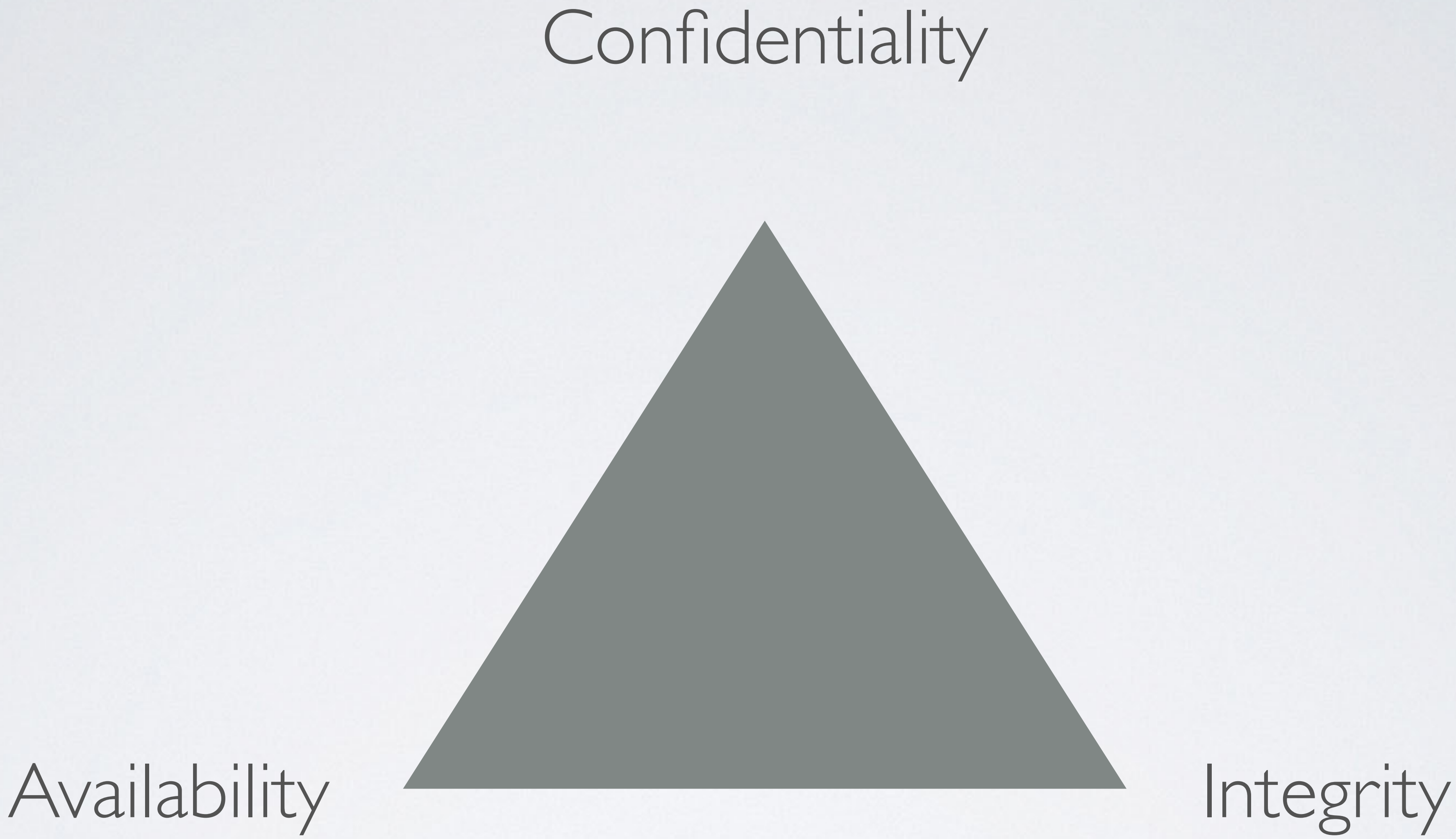
SO WHERE ARE OUR SECURITY METRICS?

We don’t have useful ones.

We advertise gigabytes of storage, MIPS, # of processors, $$ per system, MB/sec transfers …

Where are the measures of useful security properties? Privacy properties?

TRADITIONAL VIEW

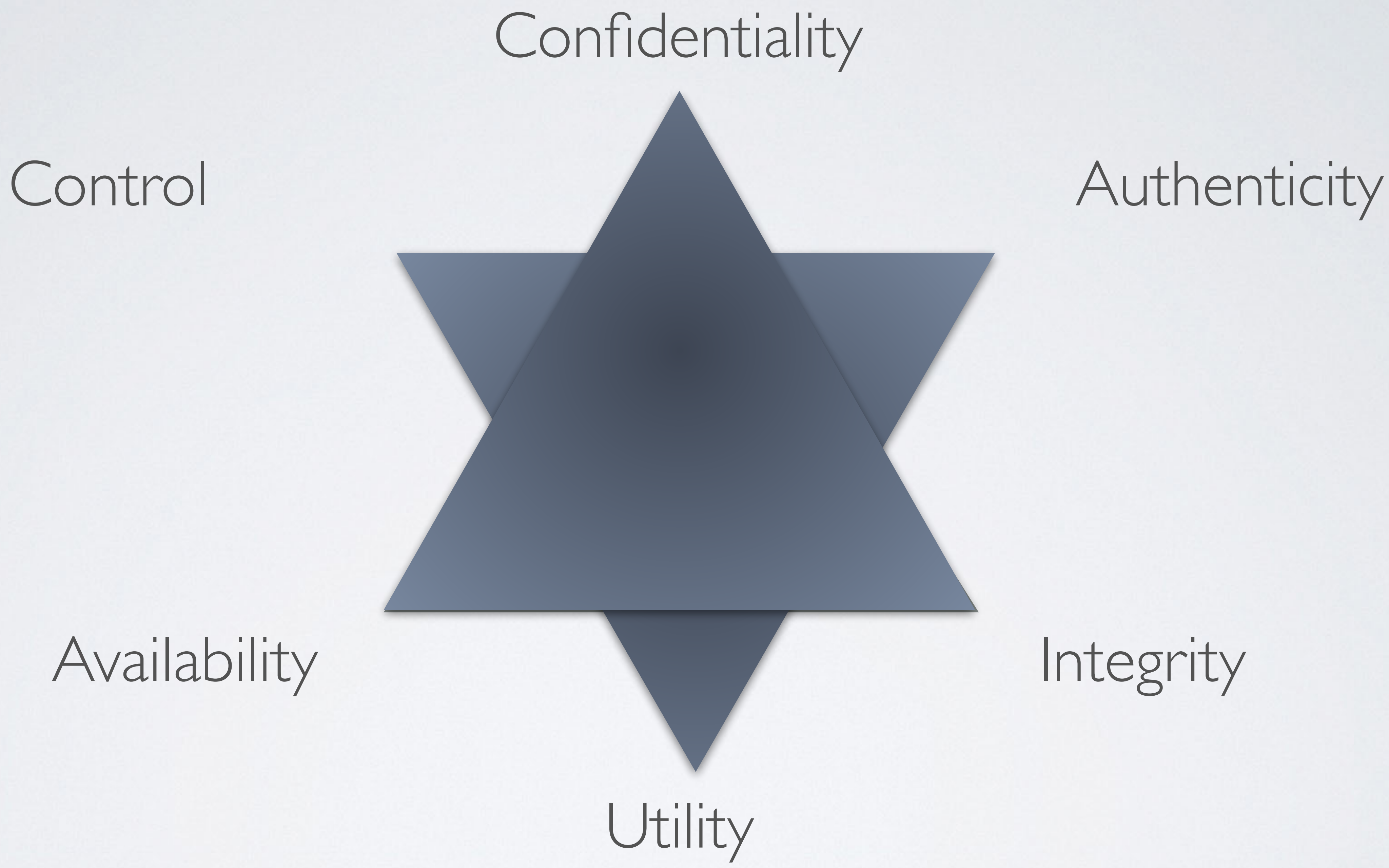


But consider where it came from….It was for a marketing event, not for designers. (Also created by Robert Courtney, btw.)

TRADITIONAL C-I-A VIEW

- Not a good model — measures aren’t orthogonal
  - Integrity overlaps availability.
  - Confidentiality assured by no availability
  - Any one of them can be used to disable the third.
- Might as well use rock, paper, scissors

DONN PARKER'S HEXAD



Some better insight, but not hugely better.

WHAT PROPERTIES DO WE NEED?

- Which properties are fundamental?
  - Correctness. Software & hardware should behave exactly as we specify it and do nothing more.
  - Without this, nothing else can be said
- So where do we start?
  - Composable, trusted components:
    - Simplicity — complexity breeds faults
    - Specificity — if we don't know what we want, we won't get it
    - Limited interactions — we can't control what we don' t know is happening

OTHER NEEDS

- Non-subvertable, parameterized access controls
- Non-interfering layering of authorities
- Intuitive, non-intrusive interface
- Useful, non-subvertable identification and tagging
- Standardized, hardened functions (e.g., crypto)
- Non-subvertable auditing

LIST OF PROPERTIES

- I can’t give you a more exact list. It’s a research agenda that isn’t funded and isn’t being pursued. (We’re too busy building Internet-enabled shoes and toasters.)

- Each property should be well-defined, achievable in some context, limited, and its output should be measurable. The measures should be composable.

PROPERTIES

We have come close with some developments:

- Type-safe languages
- Formal methods and verification
- Reduced microkernel systems
- Multilevel. gate-based capability architectures.

None are cheap. None will run Word, Excel, and Firefox. All will required new training, hardware, and more. Don't hold your breath.

KEY TAKEAWAY: ONE SIZE DOES NOT FIT ALL

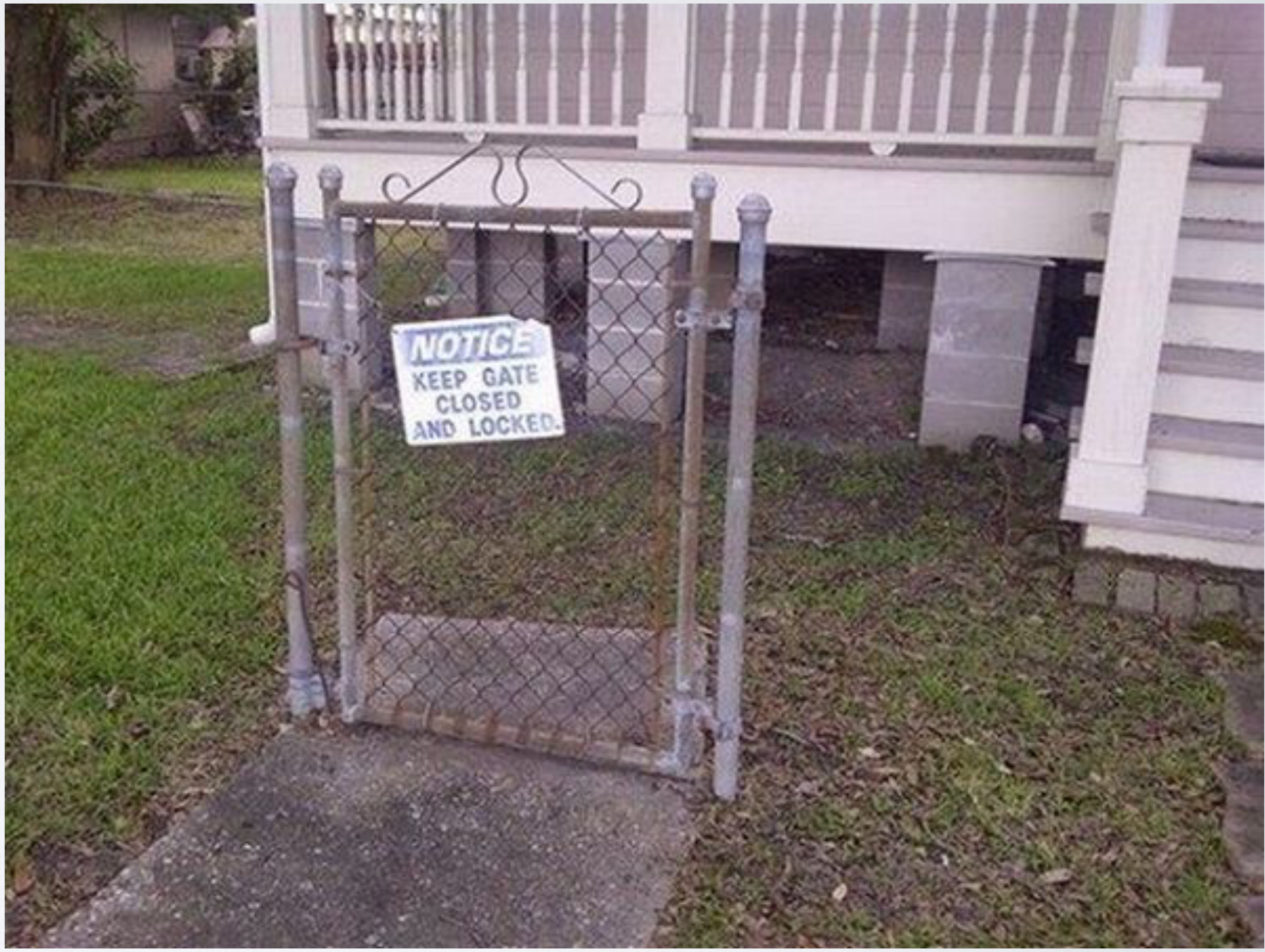
- Robert Courtney 's 3 laws:
- Nothing useful can be said about the security of a mechanism except in the context of a specific application and environment.
- Never spend more mitigating a risk than tolerating it will cost you.
- There are management solutions to technical problems but no technical solutions to management problems.

KEY TAKEAWAY: QUALITY, FIRST

KEY TAKEAWAY: IF YOU DON'T KNOW WHAT YOU'RE BUILDING, YOU'RE STUCK WITH WHAT YOU BUILD

TAKEAWAY: SECURITY MUST BE DESIGNED IN

- Adding it on afterwards results in gaps



HOW WILL ***WE*** DEFINE SECURITY?

Remember that doing the same thing over and over again expecting different results may be a sign of insanity.

Here's hoping you choose sanity.

Author's email: spaf@purdue.edu