

FAULT-TOLERANCE On THE CHEAP



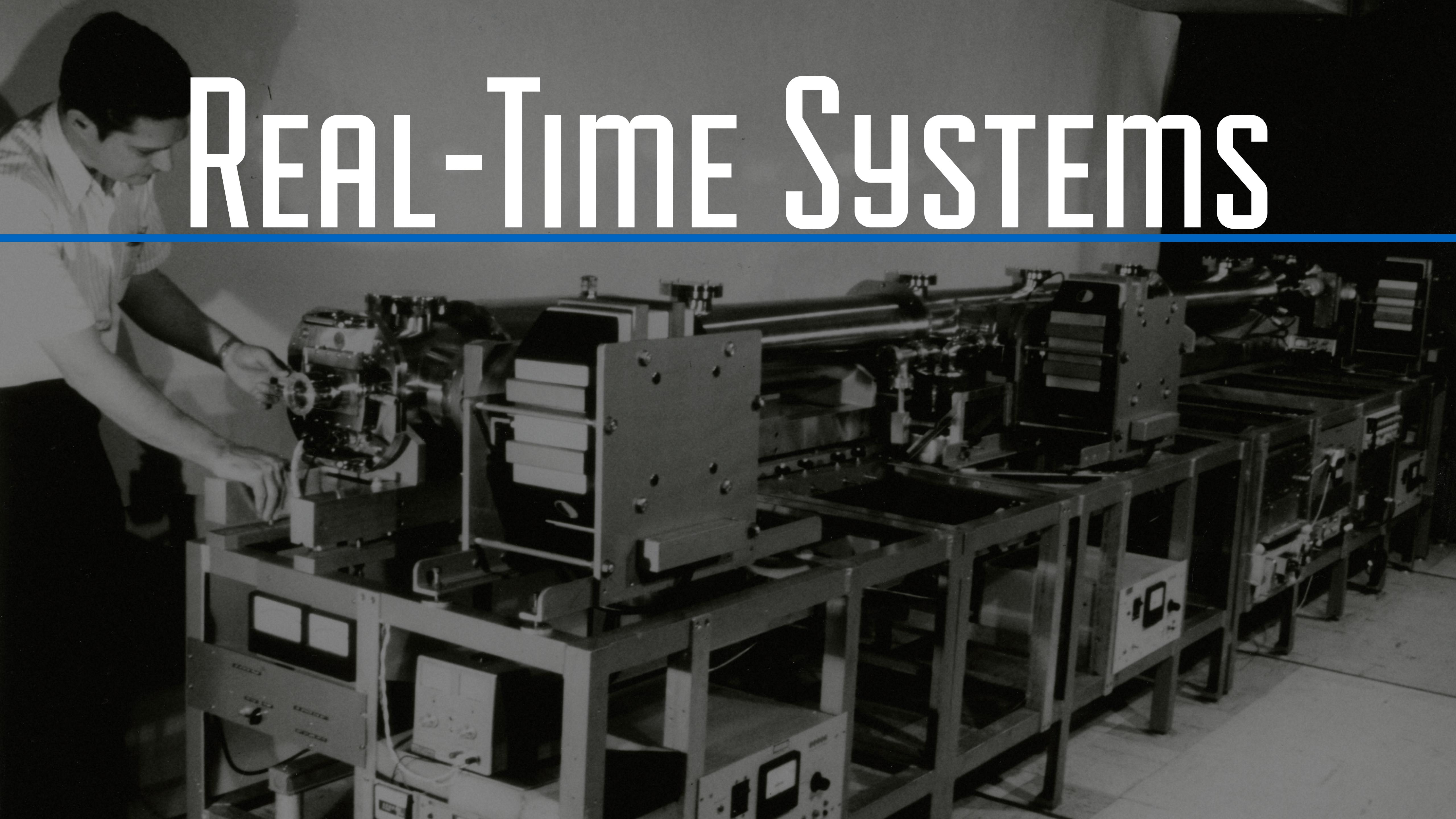
Making systems that *probably* won't fall over.

Hi, folks!

I do things
to/with
computers.

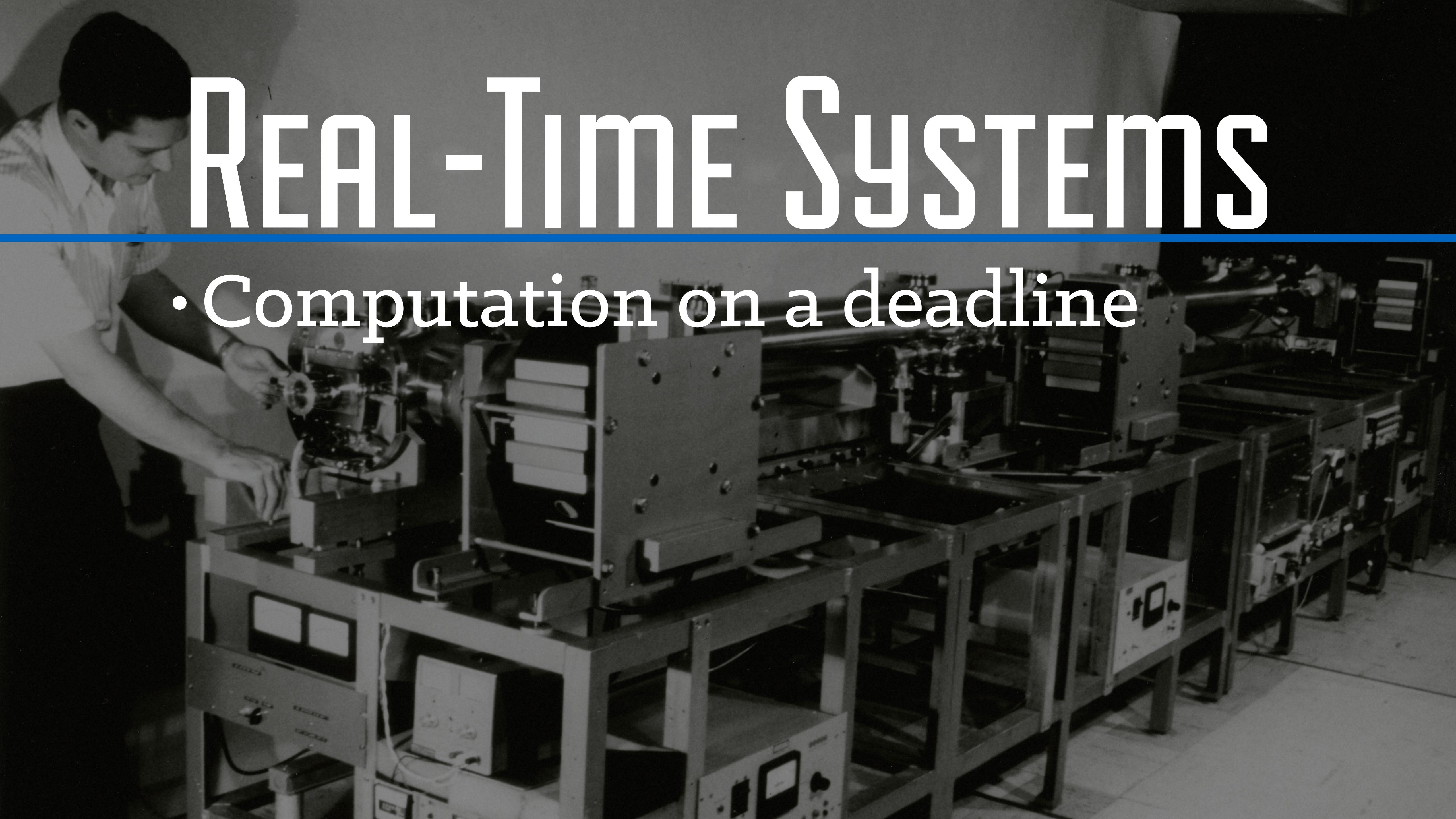
I'm a real-time,
networked
systems engineer.

REAL-TIME SYSTEMS



REAL-TIME SYSTEMS

- Computation on a deadline



REAL-TIME SYSTEMS

- Computation on a deadline
- Fail-safe / Fail-operational



REAL-TIME SYSTEMS

- Computation on a deadline
- Fail-safe / Fail-operational
- Guaranteed response / Best effort

REAL-TIME SYSTEMS

- Computation on a deadline
- Fail-safe / Fail-operational
- Guaranteed response / Best effort
- Resource adequate / inadequate

LONDON

PARIS

SHIP-TO-SHORE

BERMUDA

Buenos Aires

Rio de Janeiro

LONDON

NETWORKED SYSTEMS



LONDON

PARIS

SHIP-TO-SHORE

BERMUDA

Buenos Aires

Rio de Janeiro

LONDON

NETWORKED SYSTEMS

- Out-of-order messages

NETWORKED SYSTEMS

- Out-of-order messages
- No legitimate concept of “now”

NETWORKED SYSTEMS

- Out-of-order messages
- No legitimate concept of “now”
- High-latency transmission

NETWORKED SYSTEMS

- Out-of-order messages
- No legitimate concept of “now”
- High-latency transmission
- Lossy transmission

AdRoll





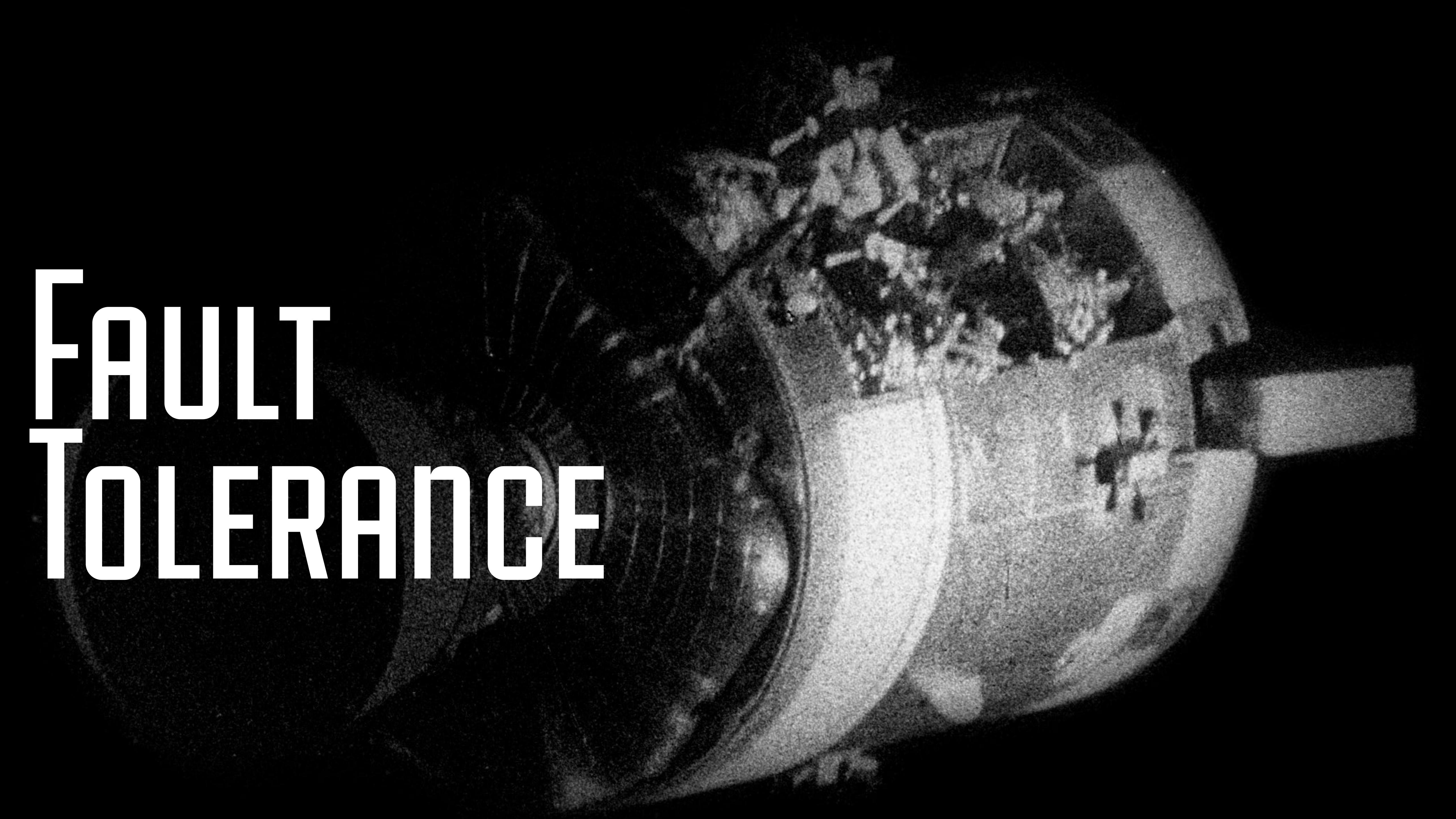
REALTIME

BIDDING

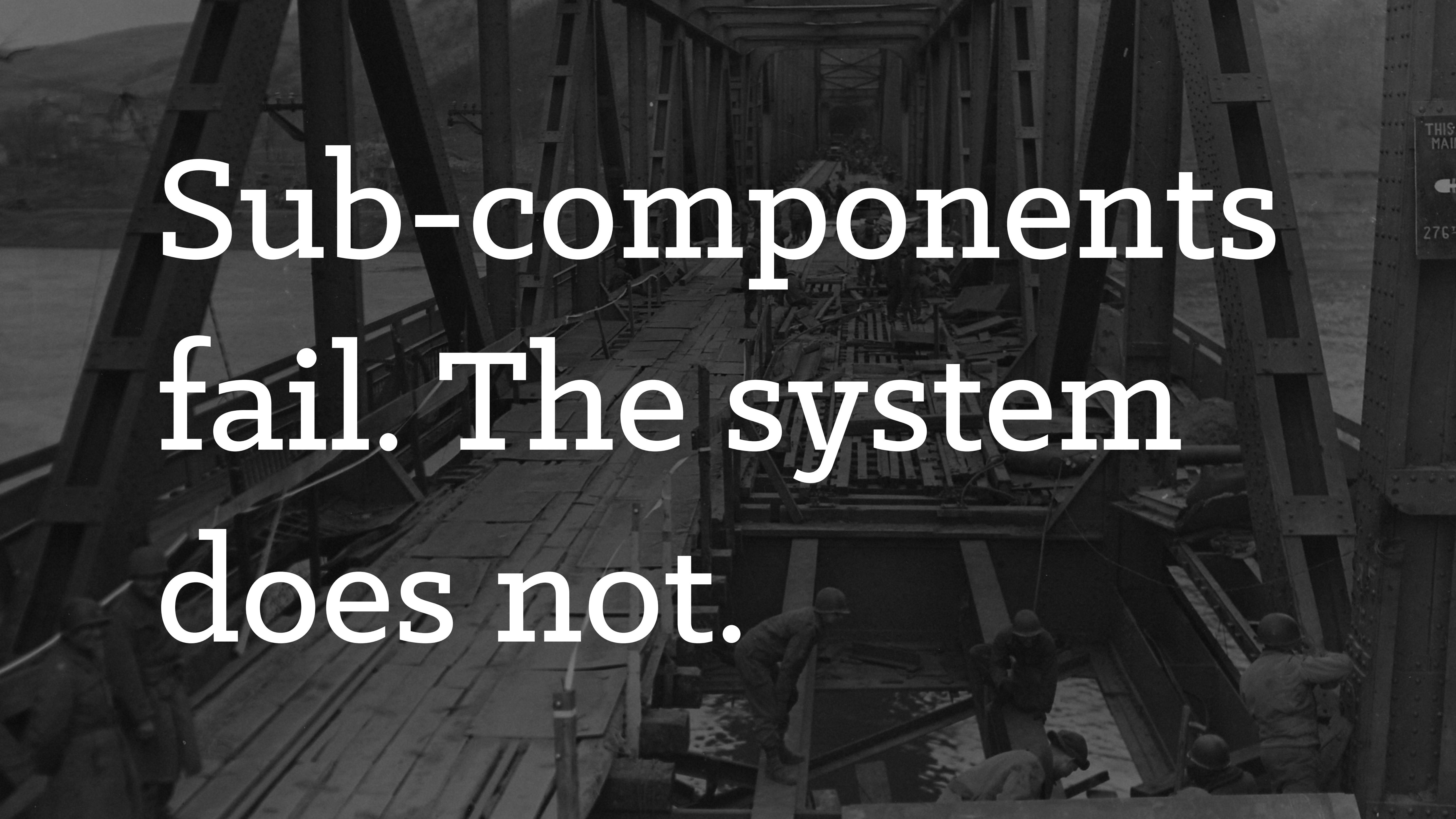
A black vintage telephone is shown from a slightly elevated angle. The phone has a dark, shiny finish. A red watermark with the word "Erlang" in a large, stylized font is overlaid across the center of the device. The word "Erlang" is written in a bold, sans-serif font, colored bright red.

Erlang

1777



**FAULT
TOLERANCE**

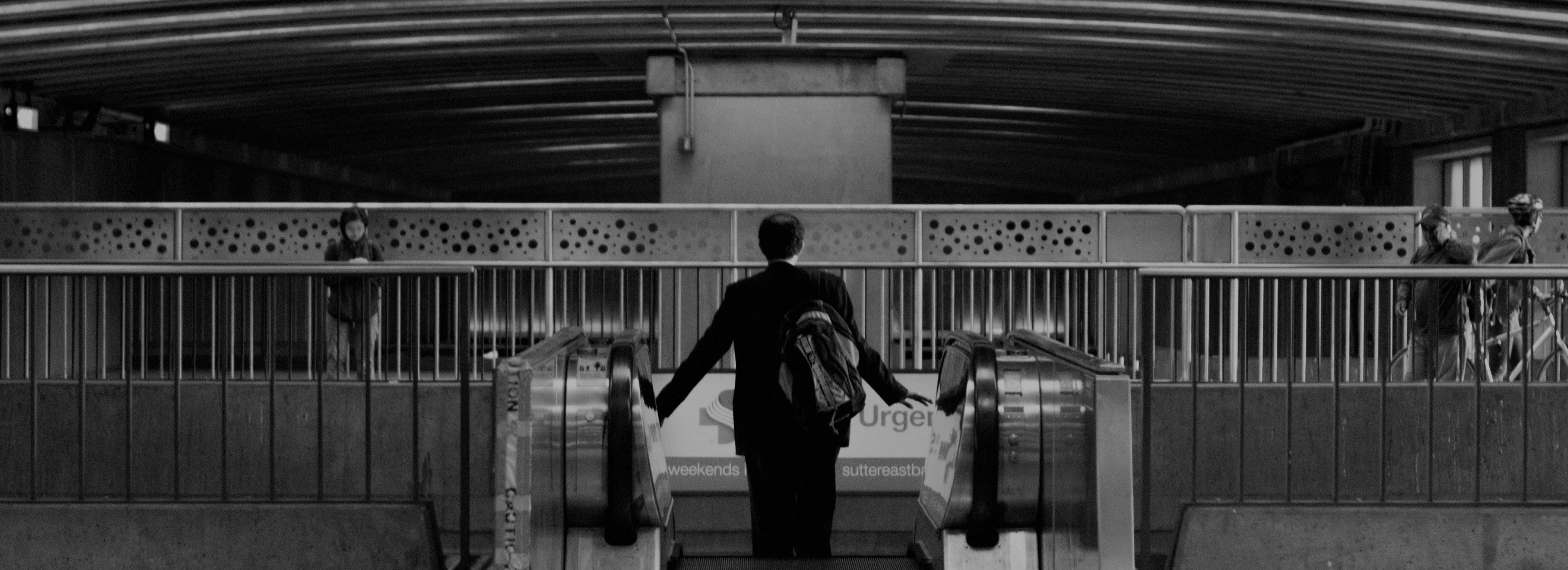


Sub-components
fail. The system
does not.



Well, not right
away...

What's it take?



Option 1: Perfection



Option 1: Perfection

- Total control over the whole mechanism.

Option 1: Perfection

- Total control over the whole mechanism.
- Total understanding of the problem domain.

Option 1: Perfection

- Total control over the whole mechanism.
- Total understanding of the problem domain.
- Specific, explicit system goals.

Option 1: Perfection

- Total control over the whole mechanism.
- Total understanding of the problem domain.
- Specific, explicit system goals.
- Well-known service lifetime.

Option 1: Perfection

“They Write the Right Stuff”
Fast Company, 2005

Option 1: Perfection

- Extremely expensive.

Option 1: Perfection

- Extremely expensive.
- Intentionally stifles creativity.

Option 1: Perfection

- Extremely expensive.
- Intentionally stifles creativity.
- Design up front.

Option 1: Perfection

- Extremely expensive.
- Intentionally stifles creativity.
- Design up front.
- Complete control of the system is not complete.

Option 2: Hope for the Best



Option 2: Hope for the Best

- Little up-front knowledge of the problem domain.

Option 2: Hope for the Best

- Little up-front knowledge of the problem domain.
- Implicit or short-term system goals.

Option 2: Hope for the Best

- Little up-front knowledge of the problem domain.
- Implicit or short-term system goals.
- No money down.

Option 2: Hope for the Best

- Little up-front knowledge of the problem domain.
- Implicit or short-term system goals.
- No money down.
- Ingnutity under pressure.

Option 2: Hope for the Best

“Move fast and
break things!”

Option 2: Hope for the Best

- Ignorance of problem domain leads to long-term system issues.

Option 2: Hope for the Best

- Ignorance of problem domain leads to long-term system issues.
- Failures do propagate out toward users.

Option 2: Hope for the Best

- Ignorance of problem domain leads to long-term system issues.
- Failures do propagate out toward users.
- No, money down!

Option 2: Hope for the Best

- Ignorance of problem domain leads to long-term system issues.
- Failures do propagate out toward users.
- No, money down!
- Hard to change cultural values.

Option 3: Embrace Faults



Option 3: Embrace Faults

- Partial control over the whole mechanism.

Option 3: Embrace Faults

- Partial control over the whole mechanism.
- Partial understanding of the problem domain.

Option 3: Embrace Faults

- Partial control over the whole mechanism.
- Partial understanding of the problem domain.
- Sorta explicit system goals.

Option 3: Embrace Faults

- Partial control over the whole mechanism.
- Partial understanding of the problem domain.
- Sorta explicit system goals.
- Able to spot a failure when you see one.

Option 3: Embrace Faults

“Fail fast. Either do the right thing or stop.”

“Why Do Computers Stop and What Can Be Done About it?”, Jim Gray, 1985 (paraphrase)

Option 3: Embrace Faults

- Faults are isolated but *must* be resolved in production.

Option 3: Embrace Faults

- Faults are isolated but *must* be resolved in production.
- Must carefully design for introspection.

Option 3: Embrace Faults

- Faults are isolated but *must* be resolved in production.
- Must carefully design for introspection.
- Moderate design up-front.

Option 3: Embrace Faults

- Faults are isolated but *must* be resolved in production.
- Must carefully design for introspection.
- Moderate design up-front.
- Pay a little now, pay a little later.

Let's talk
embracing
faults.





There are *four*
conceptual stages to
consider.

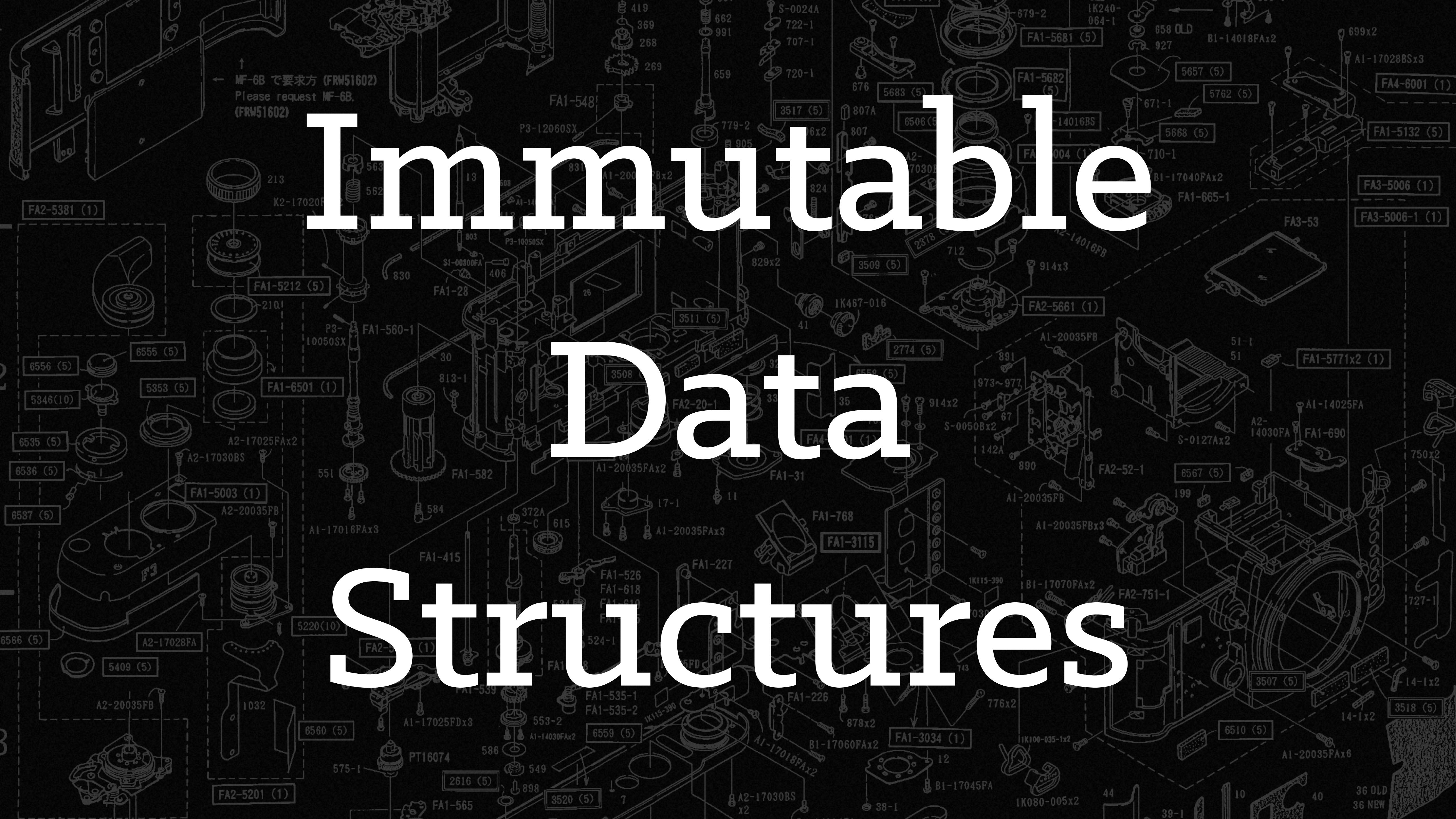
COMPONENT

The most
atomic level
of the system.



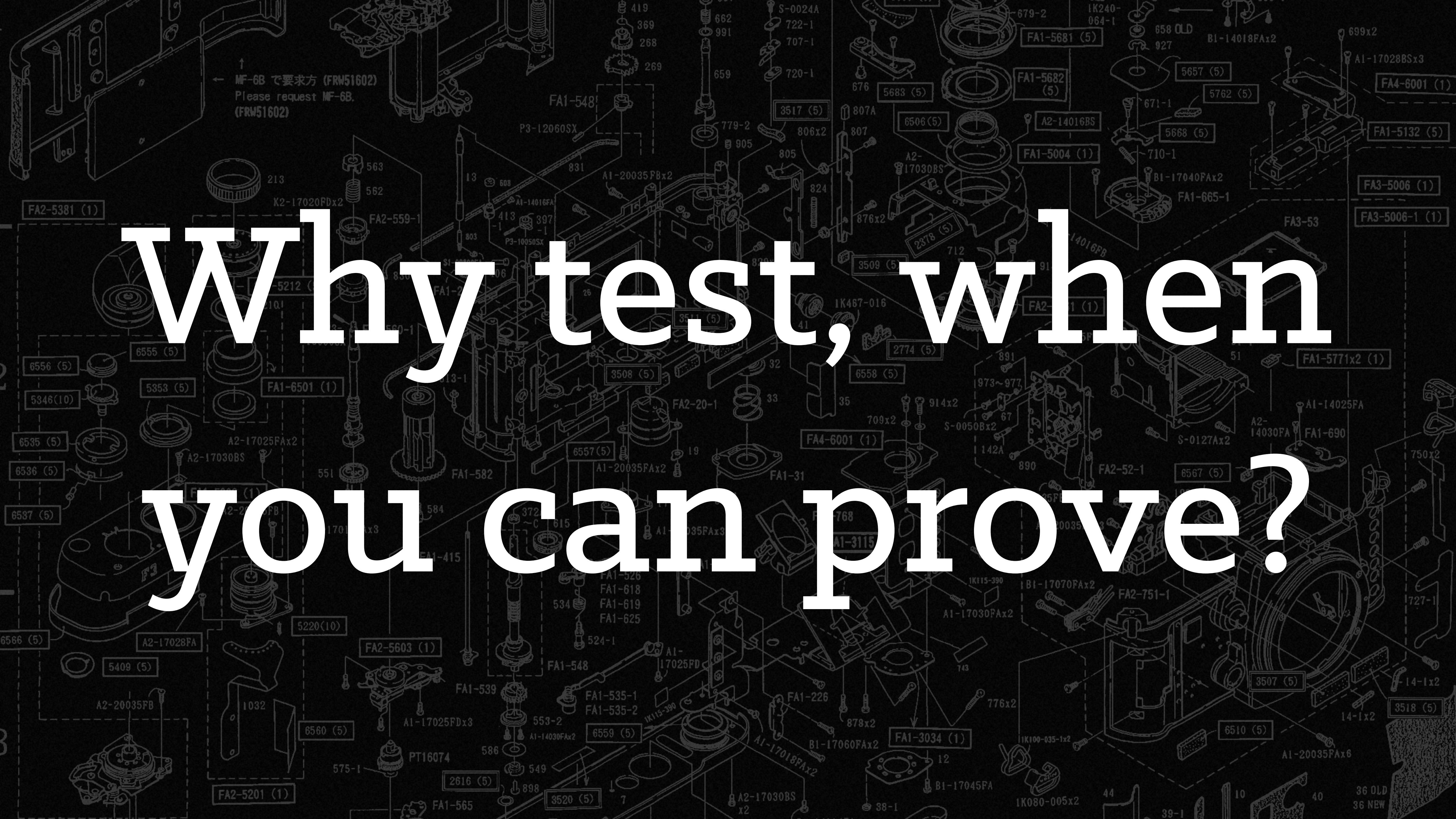
Progress here
has an outsized
impact.

immutable Data Structures



compile time
Guarantees

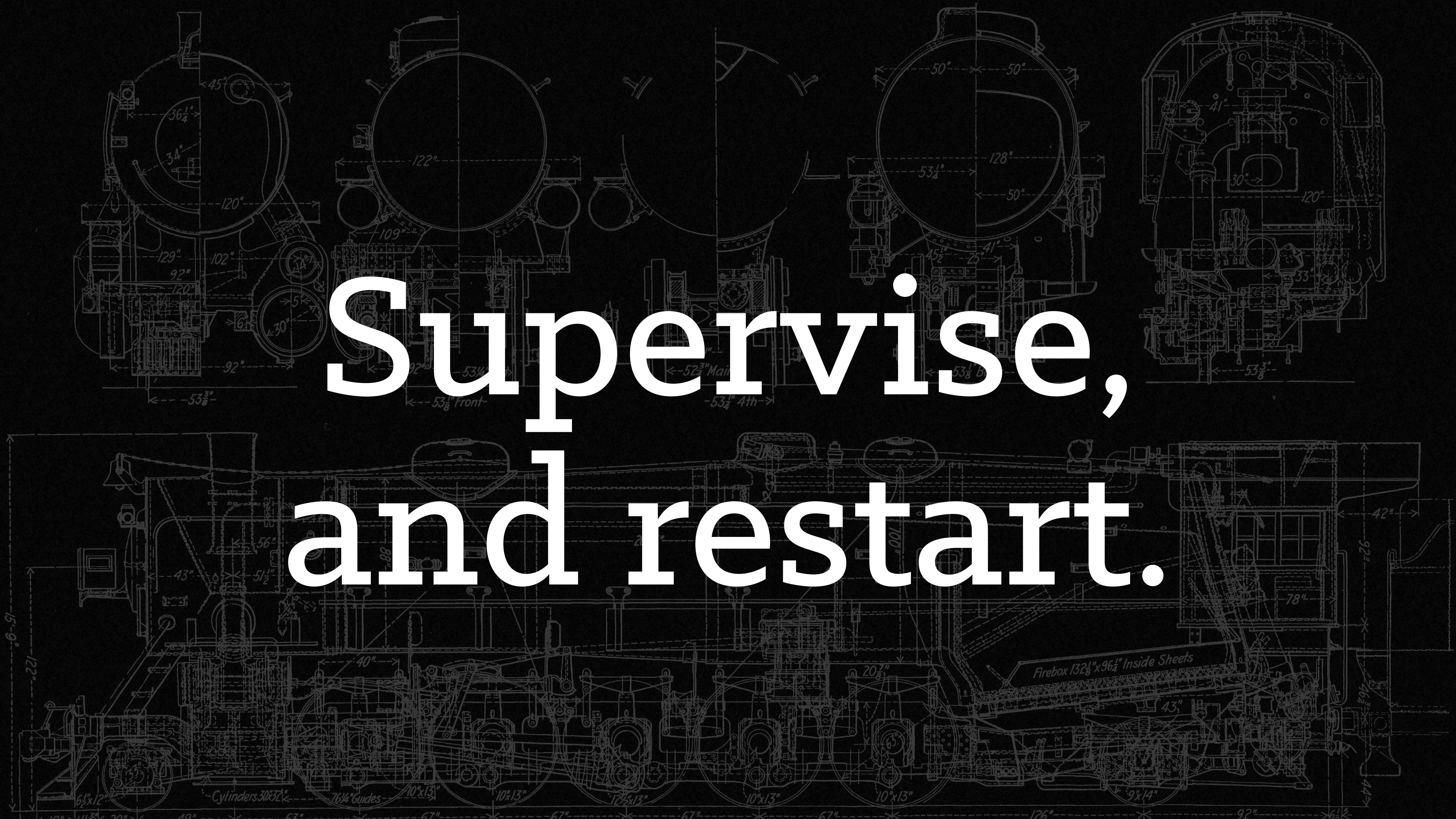
Why test, when
you can prove?



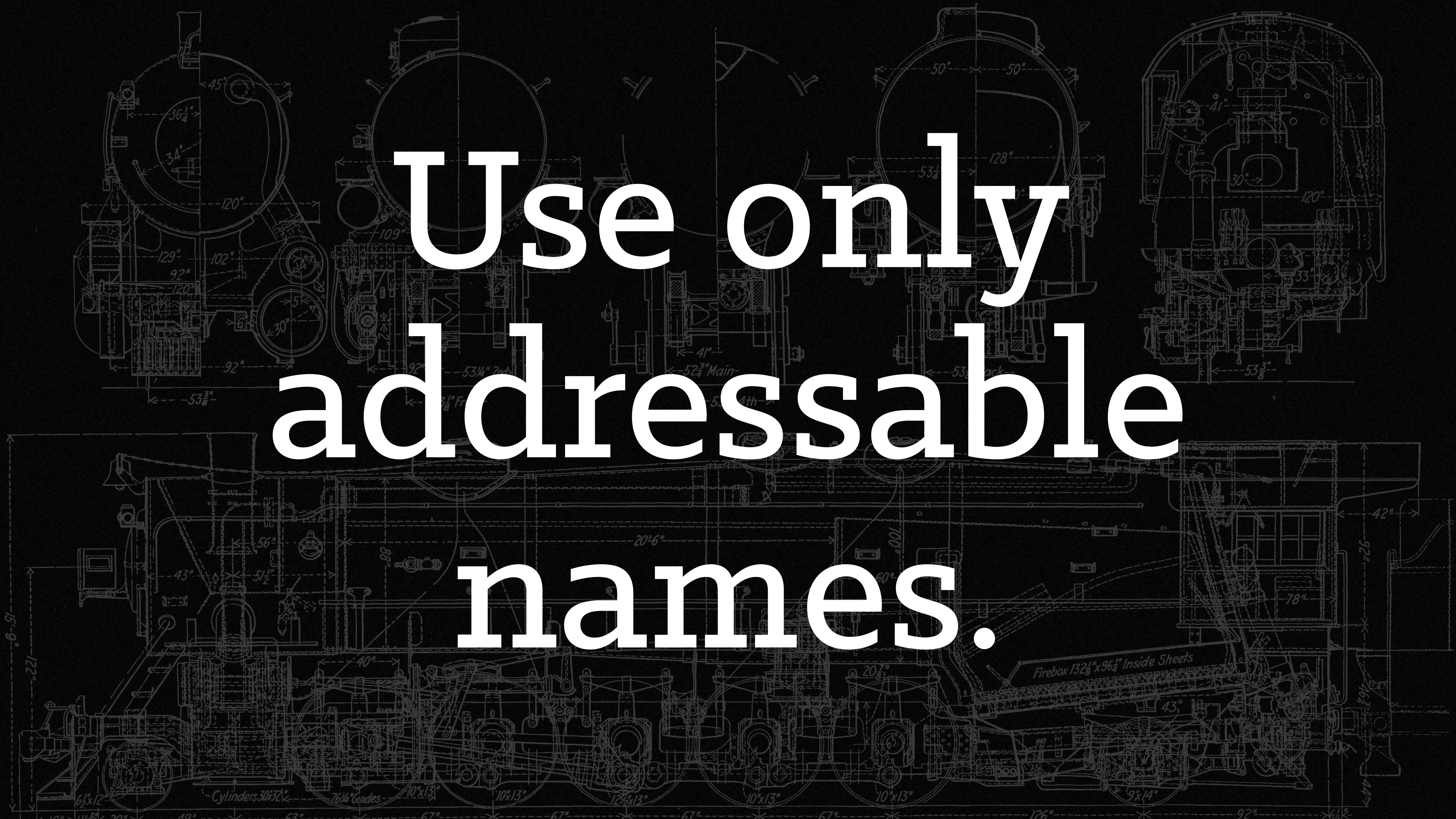
This is
Functional
Programming

Faults in
components are
exercised here.

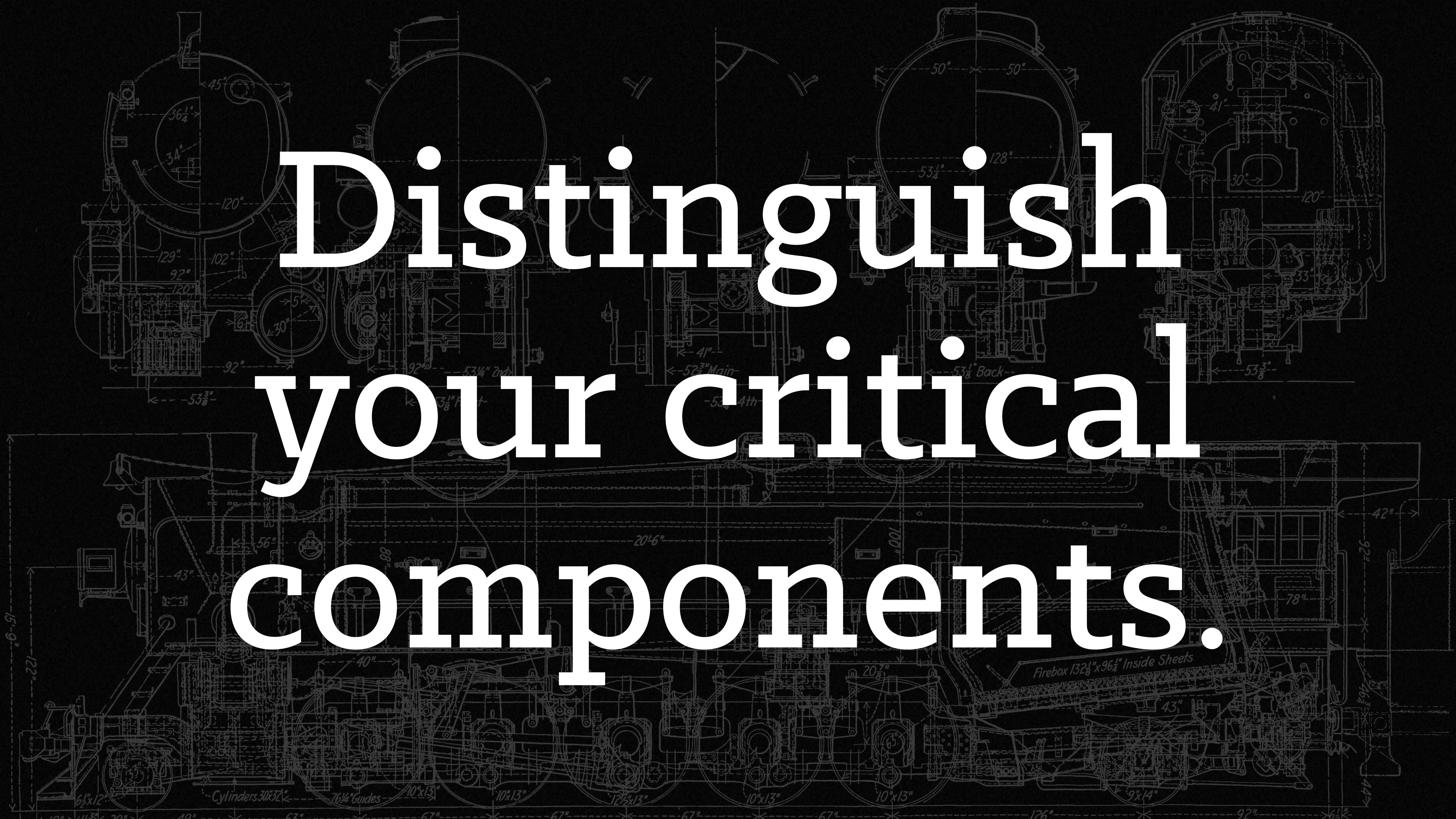
Faults in interactions are exercised here.



Supervise,
and restart.



Use only
addressable
names.

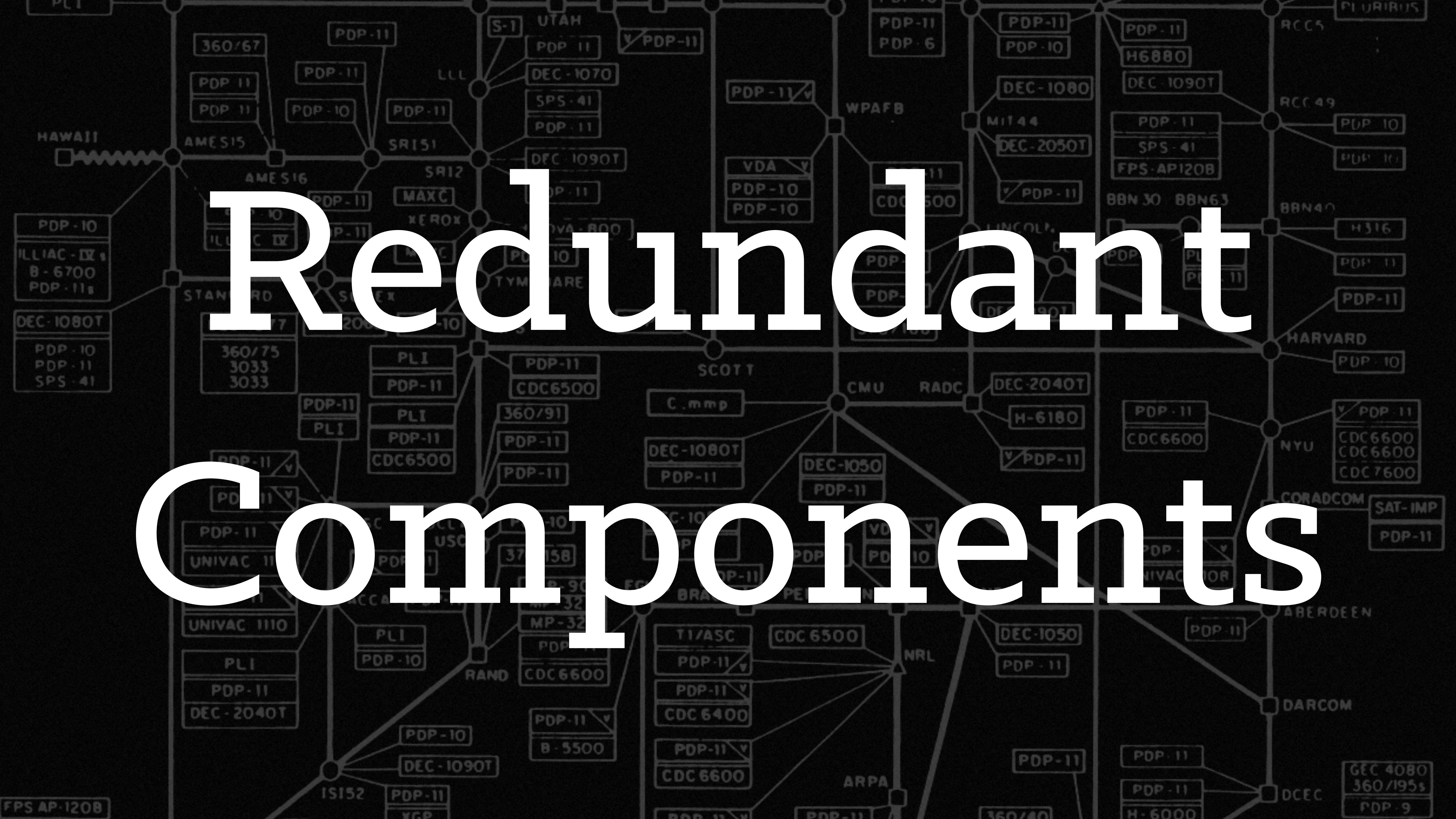


Distinguish
your critical
components.

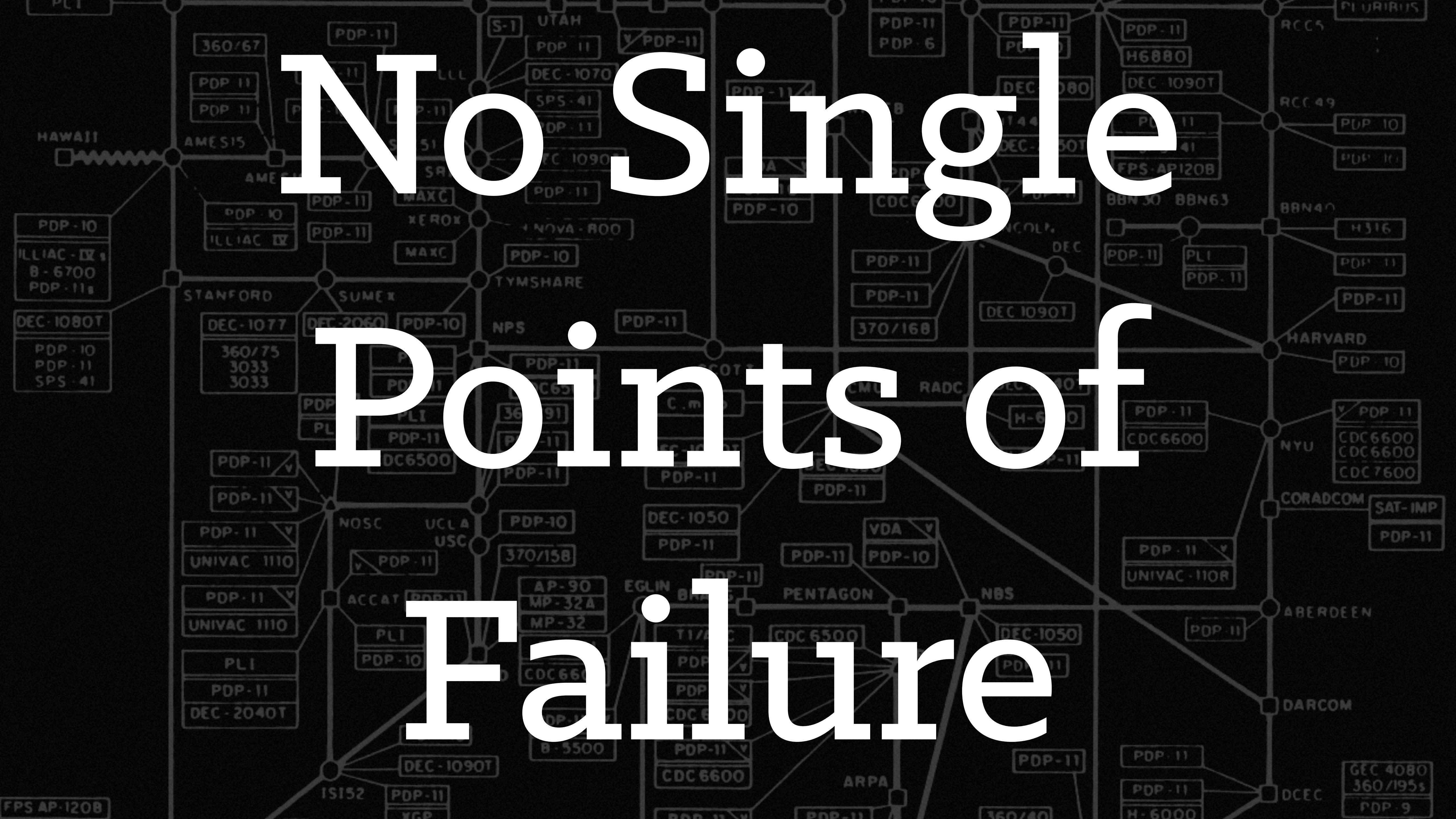
This historical network diagram illustrates the interconnectedness of several computer systems during the early days of the Internet. The nodes represent different computer systems, and the lines represent their connections. The labels on the nodes indicate the specific computer models or names of the organizations.

- CMU**: A central node connected to several other systems.
- DEC-1050**: A system located near CMU.
- PDP-11**: A system located near CMU, also connected to the VAX-11 system.
- VAX-11**: A system located near CMU, connected to the PDP-11 system.
- DEC-1050**: Another system located near CMU.
- NBS**: A node at the bottom center, connected to the VAX-11 system.
- C**: A node on the left side, connected to the DEC-20 system.
- DEC-20**: A system located near the C node.
- H-6**: A system located near the C node.
- VAX-11**: A system located near the C node, connected to the NBS node.
- PDP-11**: A system located near the C node, connected to the VAX-11 system.
- UNIVAC-110**: A system located at the bottom right, connected to the PDP-11 system.
- NYU**: A node on the right side, connected to the ABERACOM and SABRE systems.
- ABERACOM**: A system located near the NYU node.
- SABRE**: A system located near the NYU node.

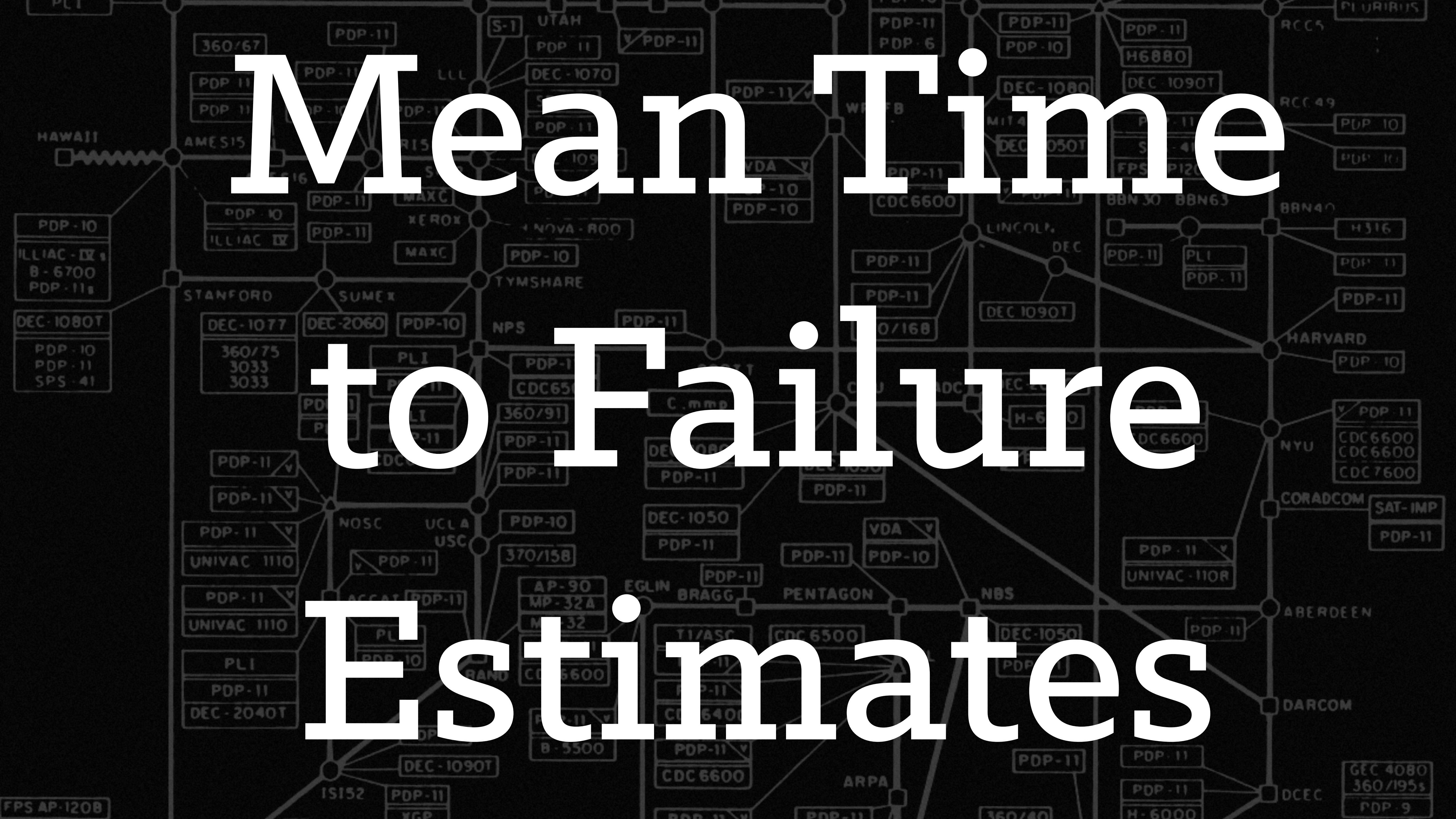
redundant components



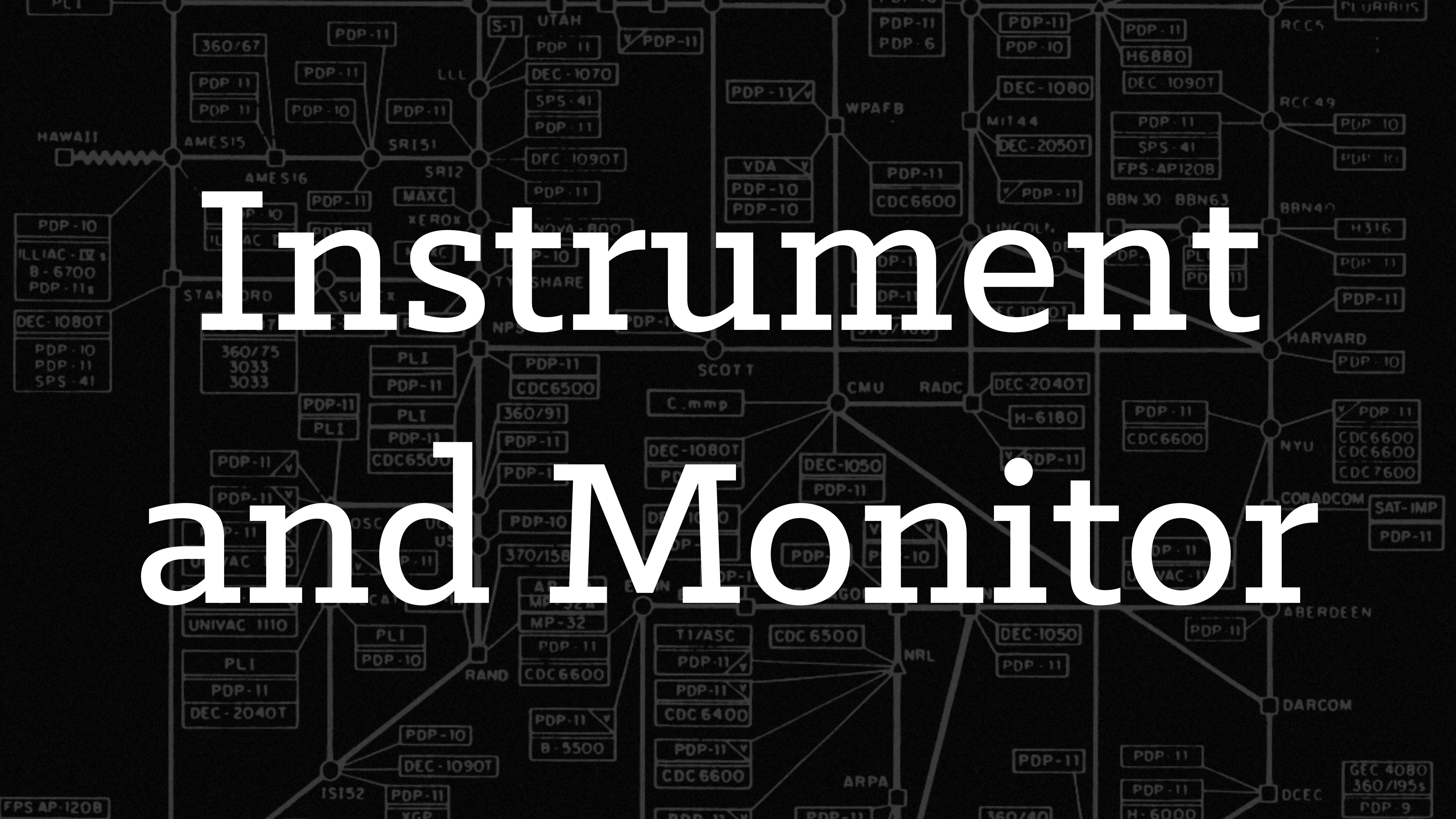
No single
points of
failure



Mean time
to failure
estimates



Instrument and Monitor



A black and white photograph of a person walking away from the camera on a path, with a cloudy sky and trees in the background.

Organization

A finely built machine
without a supporting
organization is a disaster
waiting to happen.

Chernobyl
Chevron Refinery

STS-51-L

Deepwater Horizon

Magnitogorsk

Damascus Incident to New Orleans Levee

BART ATC

Asiana #214

Therac-25

Correct the conditions
that allowed mistakes,
as well as the mistake.

A black and white photograph of a control room, likely from the Apollo era. Several operators are seated at their respective control panels, which are filled with numerous buttons, switches, and indicator lights. The room is filled with the complex machinery of a space mission control center. On the walls behind the operators, there are large world maps and several circular mission patches or insignia, including the NASA logo and various flight crew patches.

Process is
Priceless

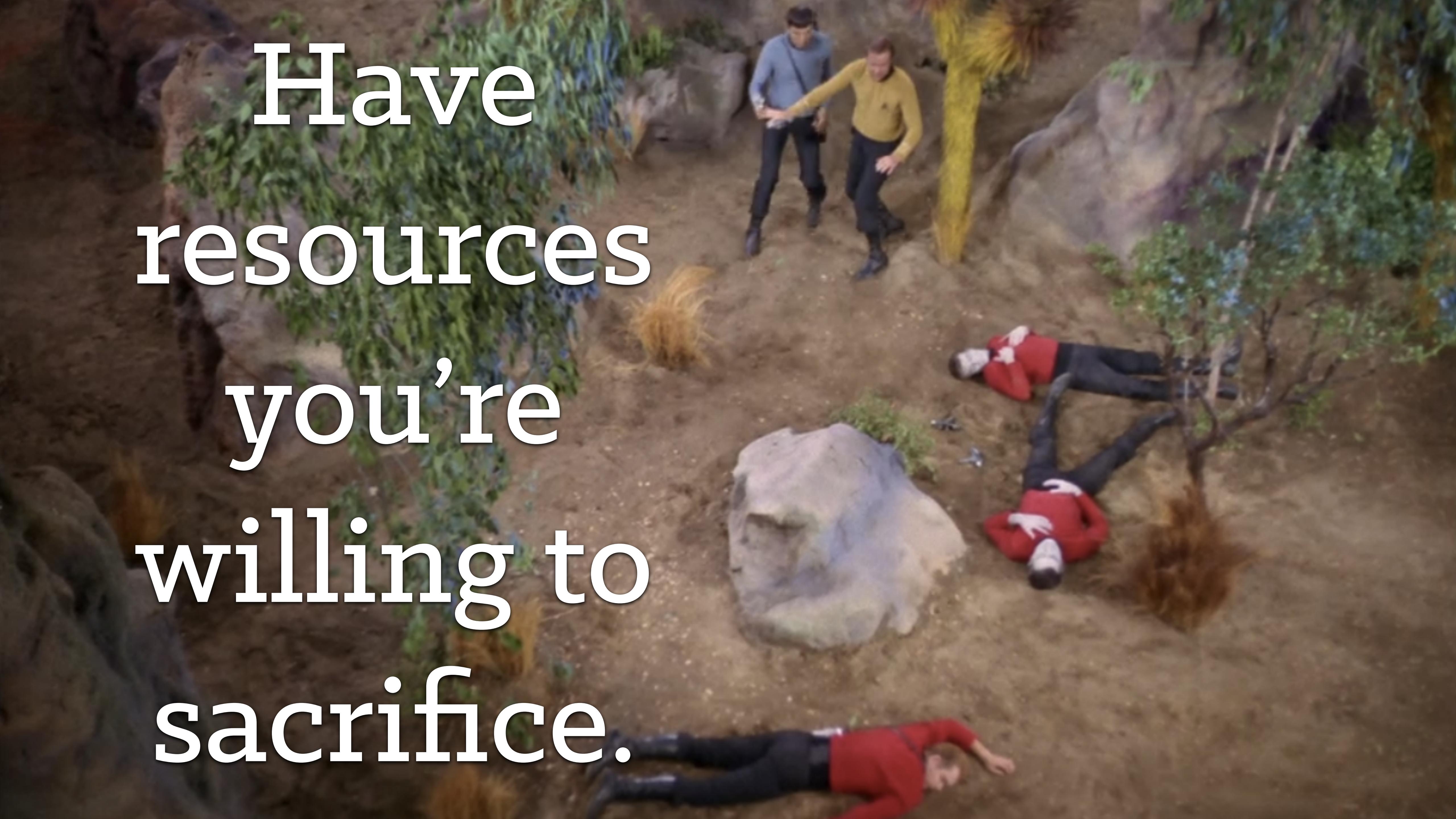
Build flexible
tools for experts.

A black and white photograph of a man standing on a large pile of rubble. He is wearing a light-colored trench coat over a dark shirt and trousers. He is pointing his right hand towards the horizon. The background shows more rubble and debris, suggesting a destroyed area.

Separate
Your
Concerns

Build with
Failure in mind.

Have
resources
you're
willing to
sacrifice.



Study
accidents.





Every system
carries the
potential for its
own destruction.

Some things aren't
worth building.



Understand Networks.



- o. The network is unreliable.
 - 1. Latency is non-zero.
 - 2. Bandwidth is finite.
 - 3. The network is insecure.
 - 4. Topology changes.
 - 5. There are many administrators.
 - 6. Transport cost is non-zero.
 - 7. The network is heterogenous.

Thanks
so
much!

@bltroutwine

RECOMMENDED READING

“Normal Accidents: Living with High-Risk Technologies”, Charles Perrow

“Digital Apollo: Human and Machine in Spaceflight”, David A. Mindel

“Command and Control: Nuclear Weapons, the Damascus Accident, and the Illusion of Safety”, Eric Schlosser

“Erlang Programming”, Simon Thompson and Francesco Cesarini

“Steeltown, USSR”, Stephen Kotkin

“The Truth About Chernobyl”, Grigorii Medvedev

“Real-Time Systems: Design Principles for Distributed Embedded Applications”, Hermann Kopetz

“The Apollo Guidance Computer: Architecture and Operation”, Frank O’Brien

“Why Do Computers Stop and What Can Be Done About It?”, Jim Gray

“Thirteen: The Apollo Flight That Failed”, Henry S.F. Cooper Jr.