

A Paradigm Change for Safety and for System Engineering

Prof. Nancy G. Leveson

Aeronautics and Astronautics
MIT



Bottom-Line Up Front (BLUF)

- Our methods and tools are being swamped by the complexity in the systems we are trying to build.
- New causes of losses appearing (not just component failures).
- To make progress, we need a paradigm change in
  - Engineering for safety
  - MBSE models and tools
  - General system engineering approaches (FUSE)

General Definition of “Safety”

Accident = Mishap = Loss: Any undesired and unplanned event that results in a loss

- Loss of human life or injury
- Property damage,
- Environmental pollution,
- Mission loss,
- Loss of protected information,
- Negative business impact (damage to reputation, etc.),
- etc.

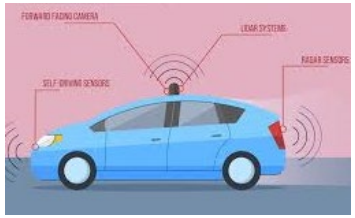
- Includes inadvertent and intentional losses (security)

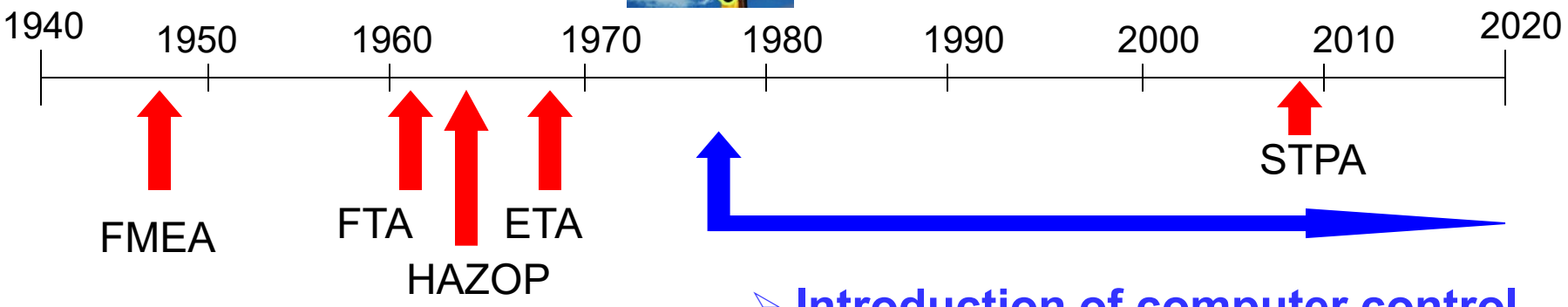
What is the Problem?

The first step in solving any problem is understanding it.

"It's never what we don't know that stops us. It's what we do know that just ain't so."

Our current safety tools are all 50-75 years old but our technology is very different today





Assume accidents caused by component failures

- **Introduction of computer control**
- **Exponential increases in complexity**
- **New technology**
- **Changes in human roles**

Warsaw A320 Accident

- Software protects against activating thrust reversers when airborne
- Hydroplaning and other factors made the software think the plane had not landed
- Pilots could not activate the thrust reversers and ran off end of runway into a small hill.

Boeing 787 Lithium Battery Fires

Models predicted 787 battery thermal problems would occur once in 10 million flight hours…but two batteries overheated in just two weeks in 2013

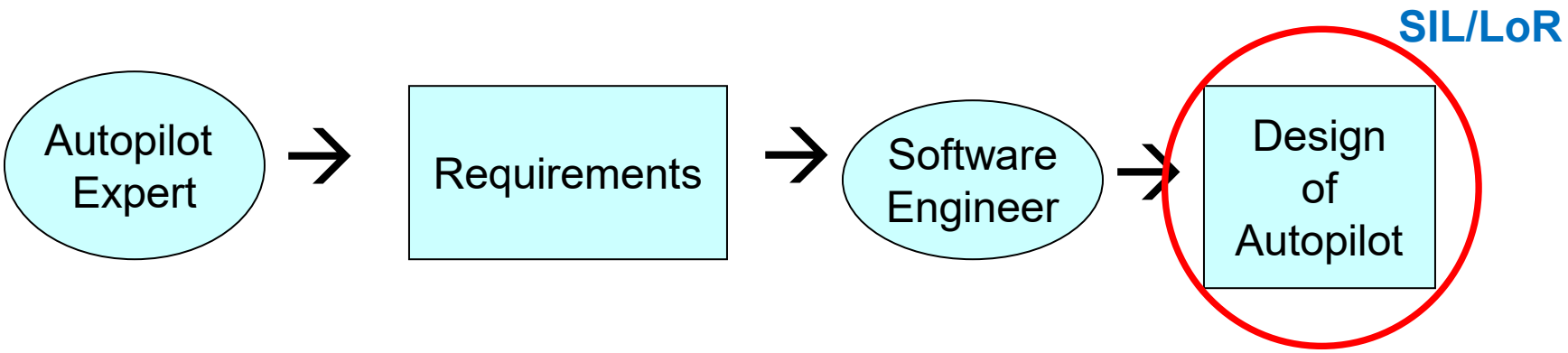
Boeing 787 Lithium Battery Fires

- A module monitors for smoke in the battery bay, controls fans and ducts to exhaust smoke overboard.
- Power unit monitors for low battery voltage, shut down various electronics, including ventilation
- Smoke could not be redirected outside cabin

All software requirements were satisfied!
The requirements were unsafe

Software engineering focuses on implementing the requirements and validating it

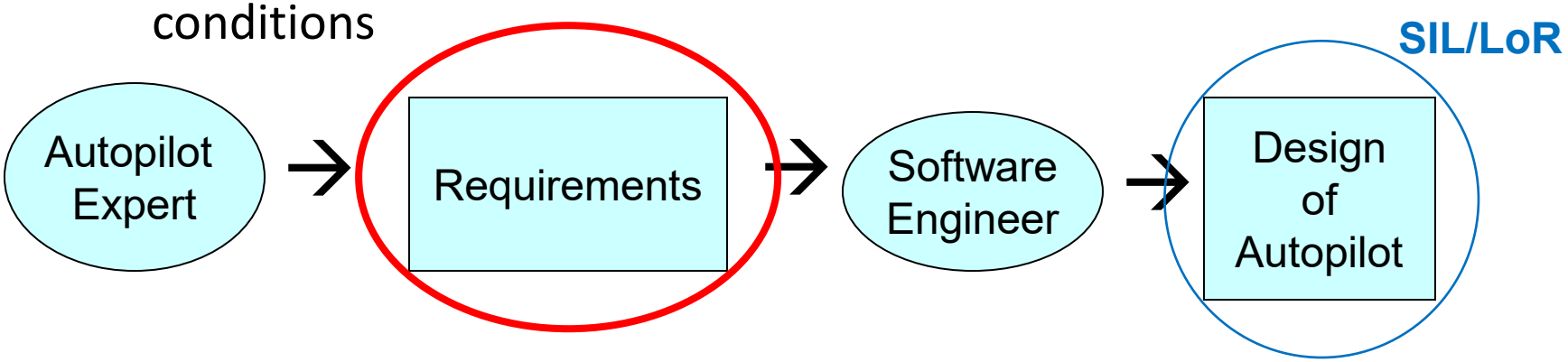
- Ensure rigor placed on design and test



- Level of rigor in producing the software design or DAL (design assurance level) has almost nothing to do with system safety.

The role of software in accidents almost always involves flawed requirements

- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions

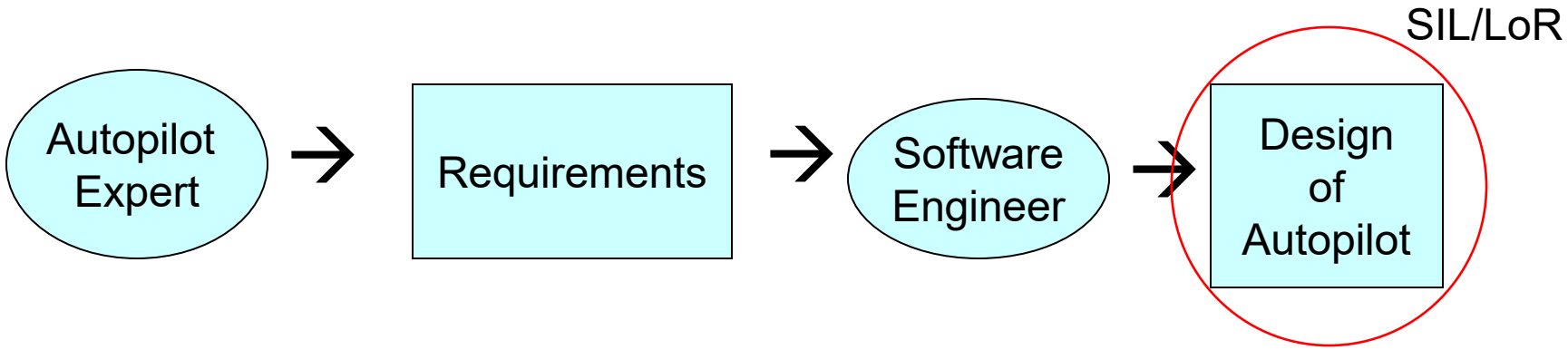


- Level of rigor in producing the software design or DAL (design assurance level) has almost nothing to do with system safety.
- The problem is **context**

Software has Revolutionized Engineering (2)

3. The role of software in accidents almost always involves flawed requirements

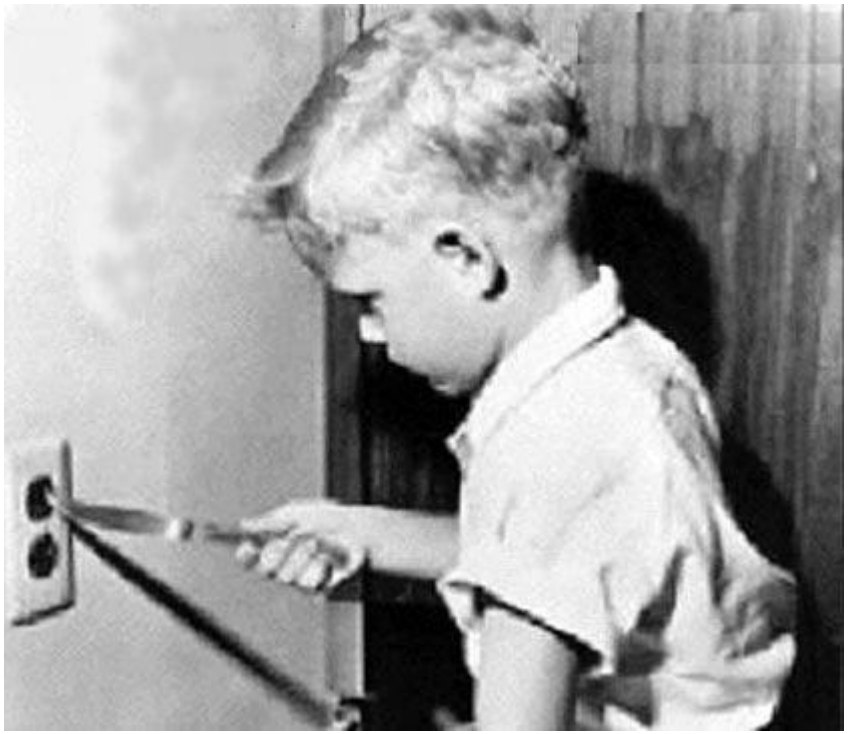
- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions



- Only trying to get the software “correct” or to make it reliable will not make it safer under these conditions

Safe or Unsafe?

Safety Depends on Context



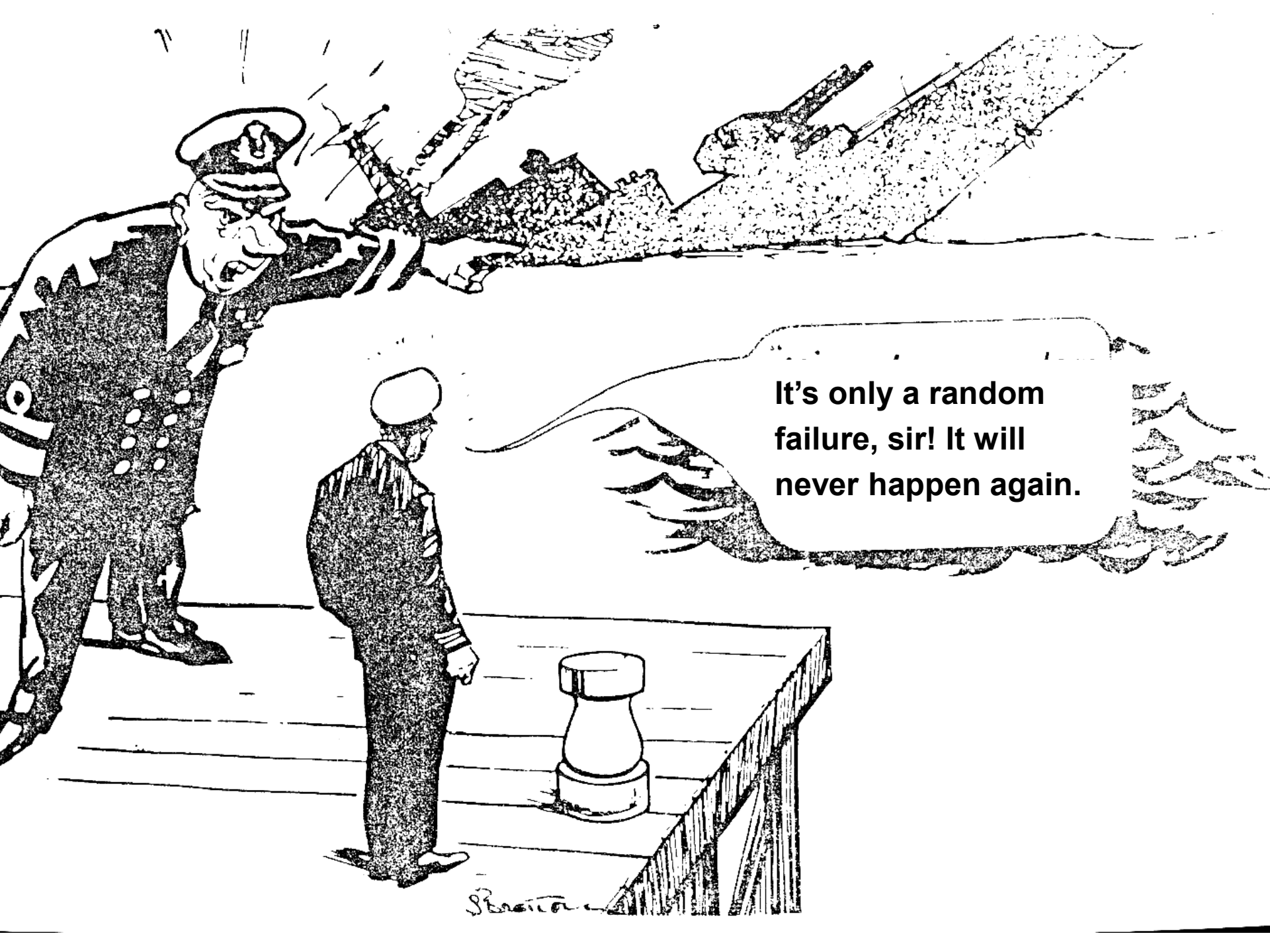
Example: Safety Always Depends on Context

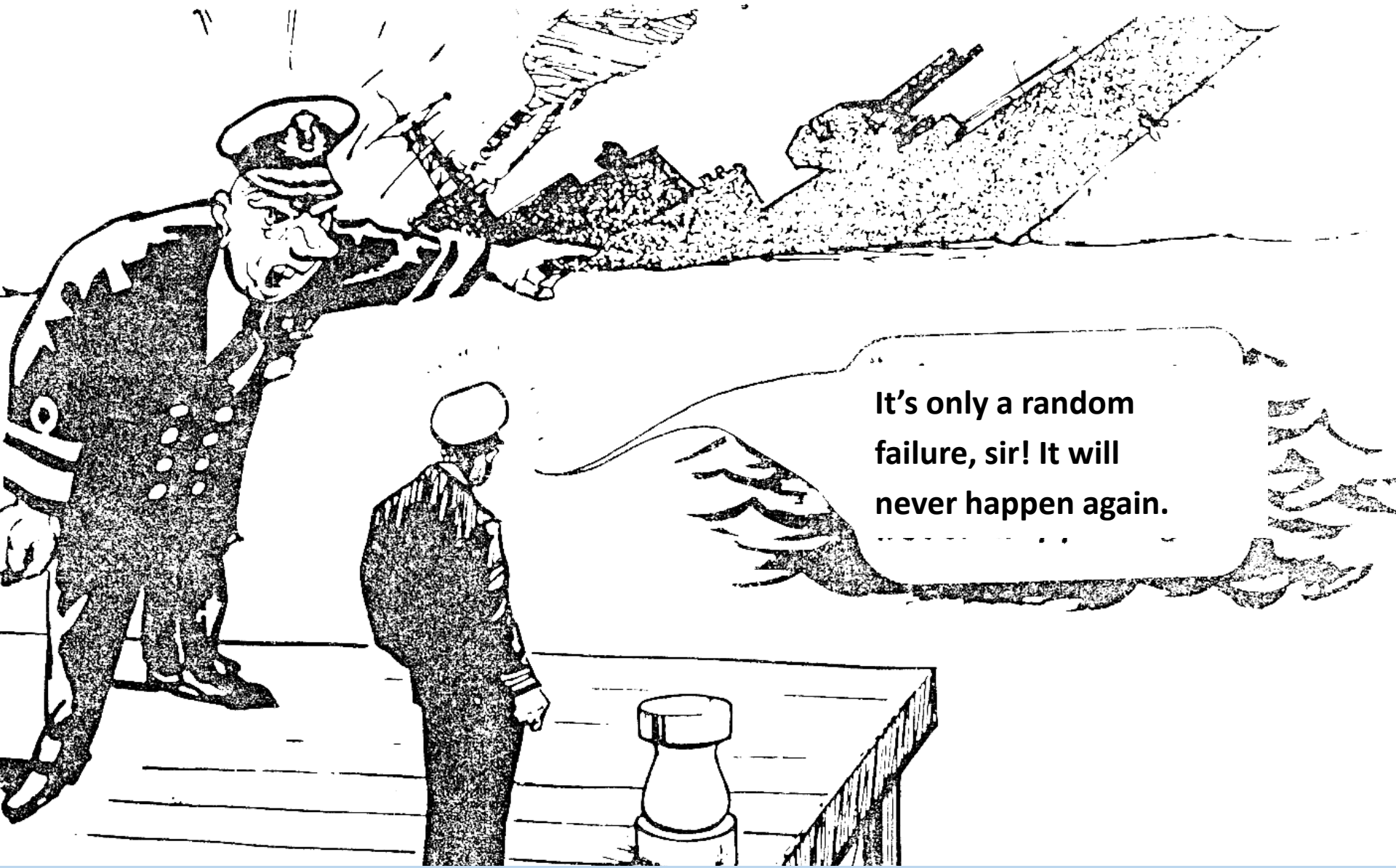
Ariane 4 IRS (Inertial Reference Software)

Ariane 5 IRS (reused same software)

Two Types of Accidents

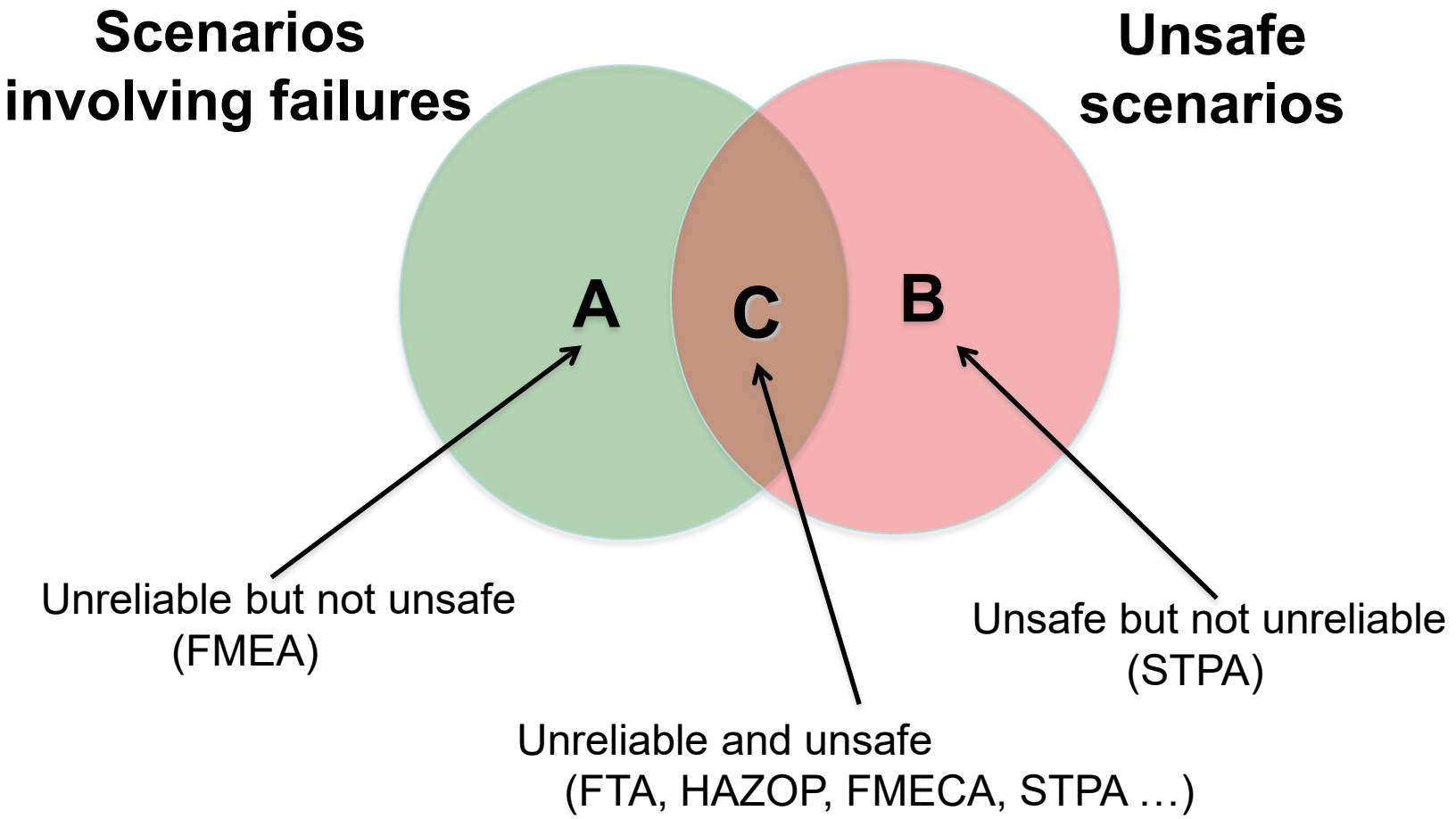
- **Component Failure Accidents**
  - Single or multiple component failures
  - Usually assume random failure
- **Component Interaction Accidents**
  - Arise in interactions among components
  - Related to complexity (coupling) in our system designs, which leads to system design and system engineering errors
  - No components may have “failed”
  - Exacerbated by introduction of computers and software but the problem is system design errors
    - Software allows almost unlimited complexity in our designs


It’s only a random failure, sir! It will never happen again.



Reliability and safety are different properties today

Confusing Safety and Reliability



Preventing Component or Functional Failures is Not Enough

Human factors concentrates on the “screen out”

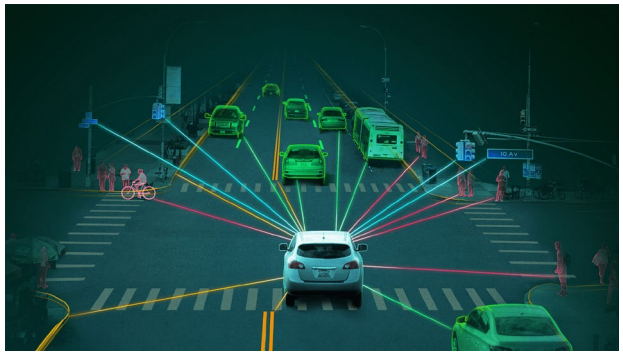
Hardware/Software engineering concentrates on the “screen in”

Not enough attention on integrated system as a whole

(e.g, mode confusion, situation awareness errors, inconsistent behavior, etc.

The Problem

- Traditional safety approaches do not work on today's systems
  - Don't handle complex systems, software, new roles for humans, management, social systems
  - Start too late – need a design first
  - Hardware, humans, software all treated separately
- No way to extend them as the underlying assumptions do not fit today's systems
- We need a paradigm change



We Need Something New

- New levels of complexity, software, human factors do not fit into a reliability-oriented world.

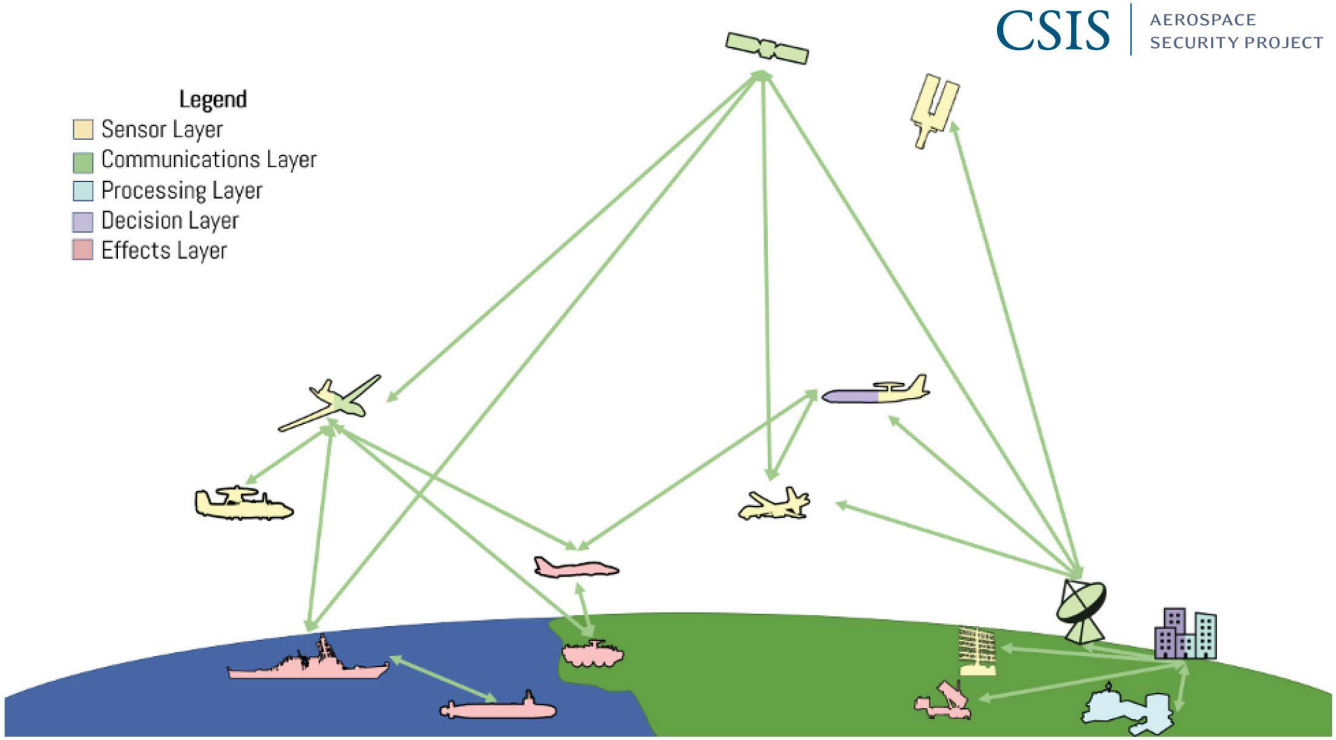
- Two approaches being taken now:

Pretend there is no problem

Shoehorn new technology and new levels of complexity into old methods

The Problem is Complexity

- We need:
  - New theoretical foundation(s)
  - New modeling languages and tools (based on that foundation)



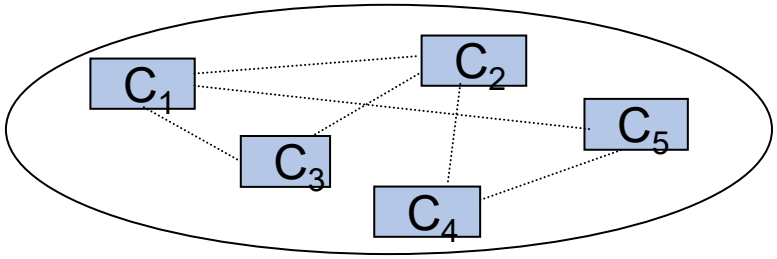
Source: Based on author's own creation

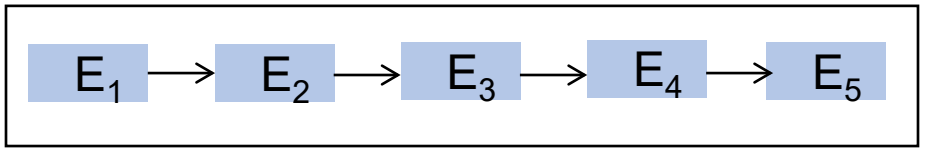
1. New Theoretical Foundation

Three ways to cope with complexity:

1. Analytical Decomposition
2. Statistics
3. Systems Theory

Analytic Decomposition ("Divide and Conquer")

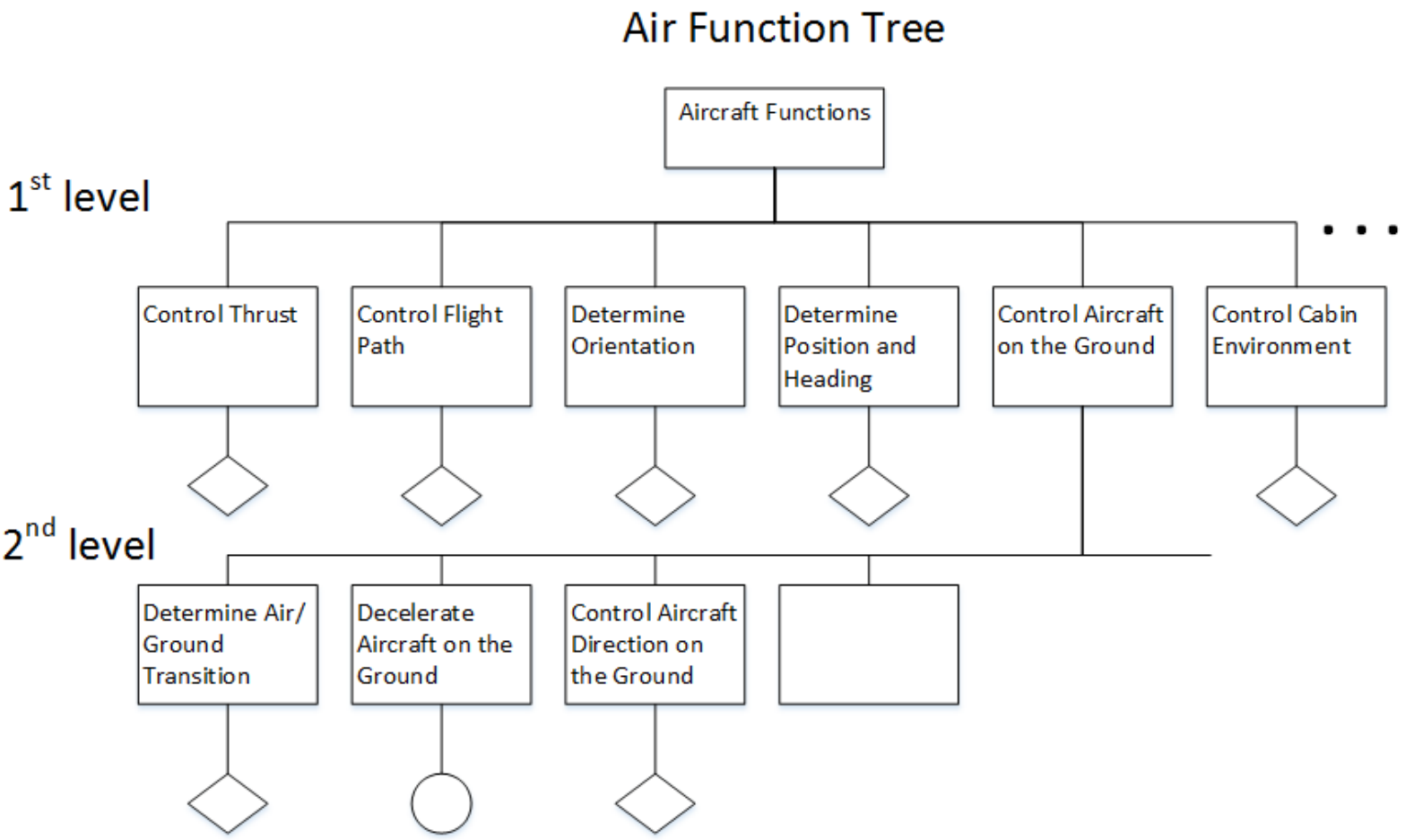




Analyze/examine pieces separately and combine results

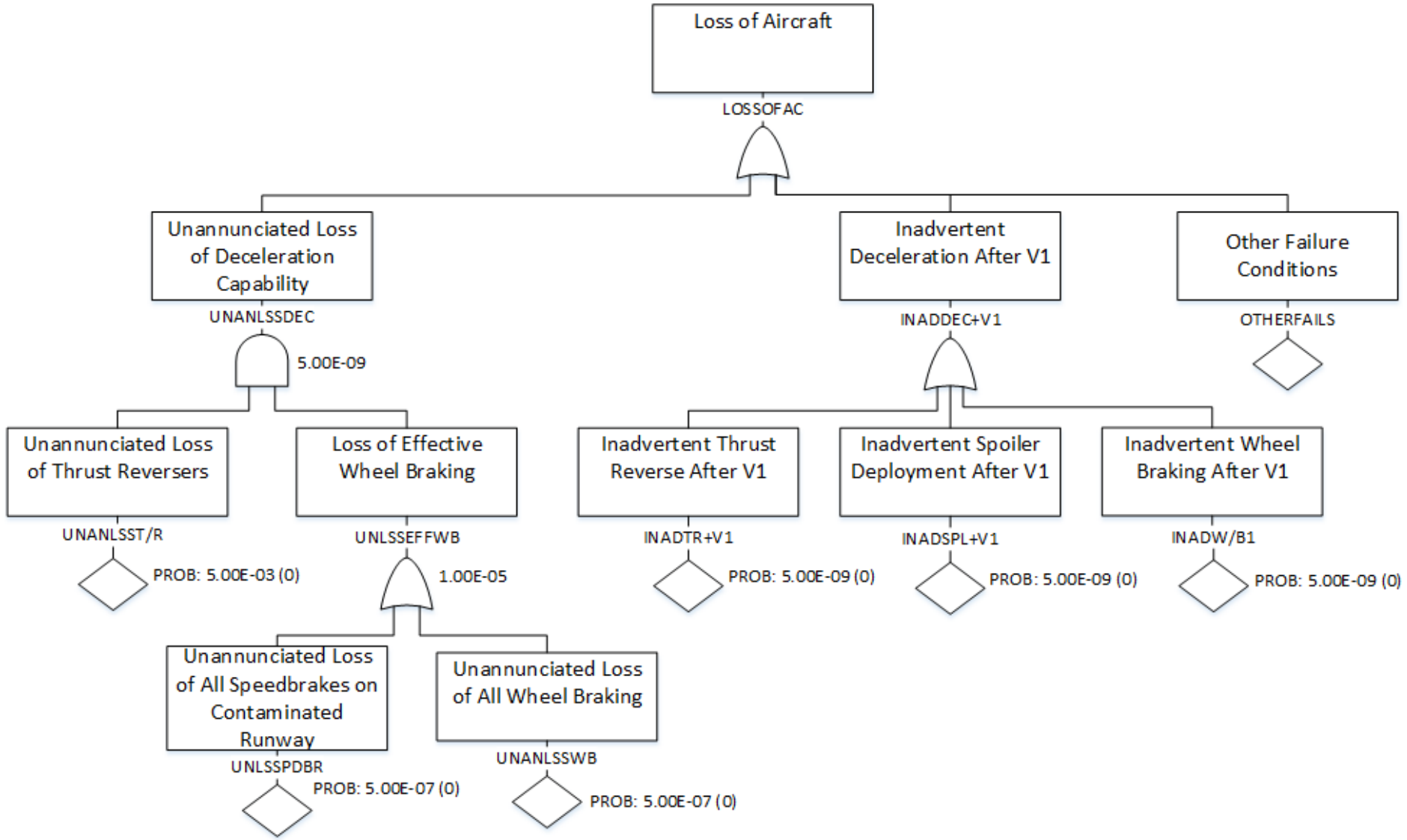
- Assumes such separation does not distort phenomenon
  - ✓ Each component or subsystem operates independently
  - ✓ Components act the same when examined singly as when playing their part in the whole
  - ✓ Components/events not subject to feedback loops and non-linear interactions
  - ✓ Interactions can be examined pairwise

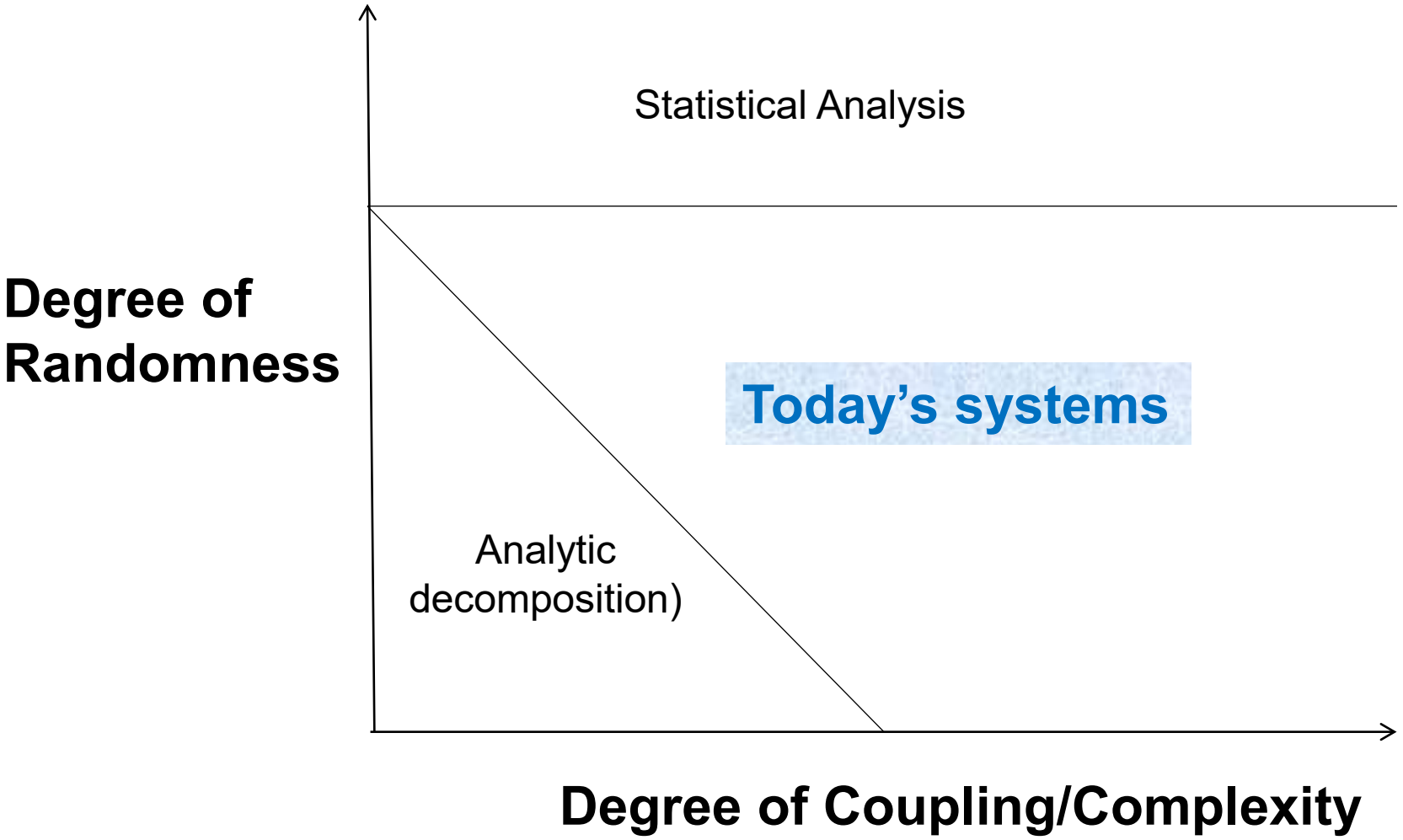
Typical Decomposition Approach (ARP 4761)



First, decompose top-down into components

Then combine individual component analyses bottom up (omit software and humans)





[Credit to Gerald Weinberg]

Here comes the paradigm change!

Prevent failures or errors

Enforce constraints on behavior:
- **components**
- **interactions among components**

Treat Safety as a **Reliability** Problem

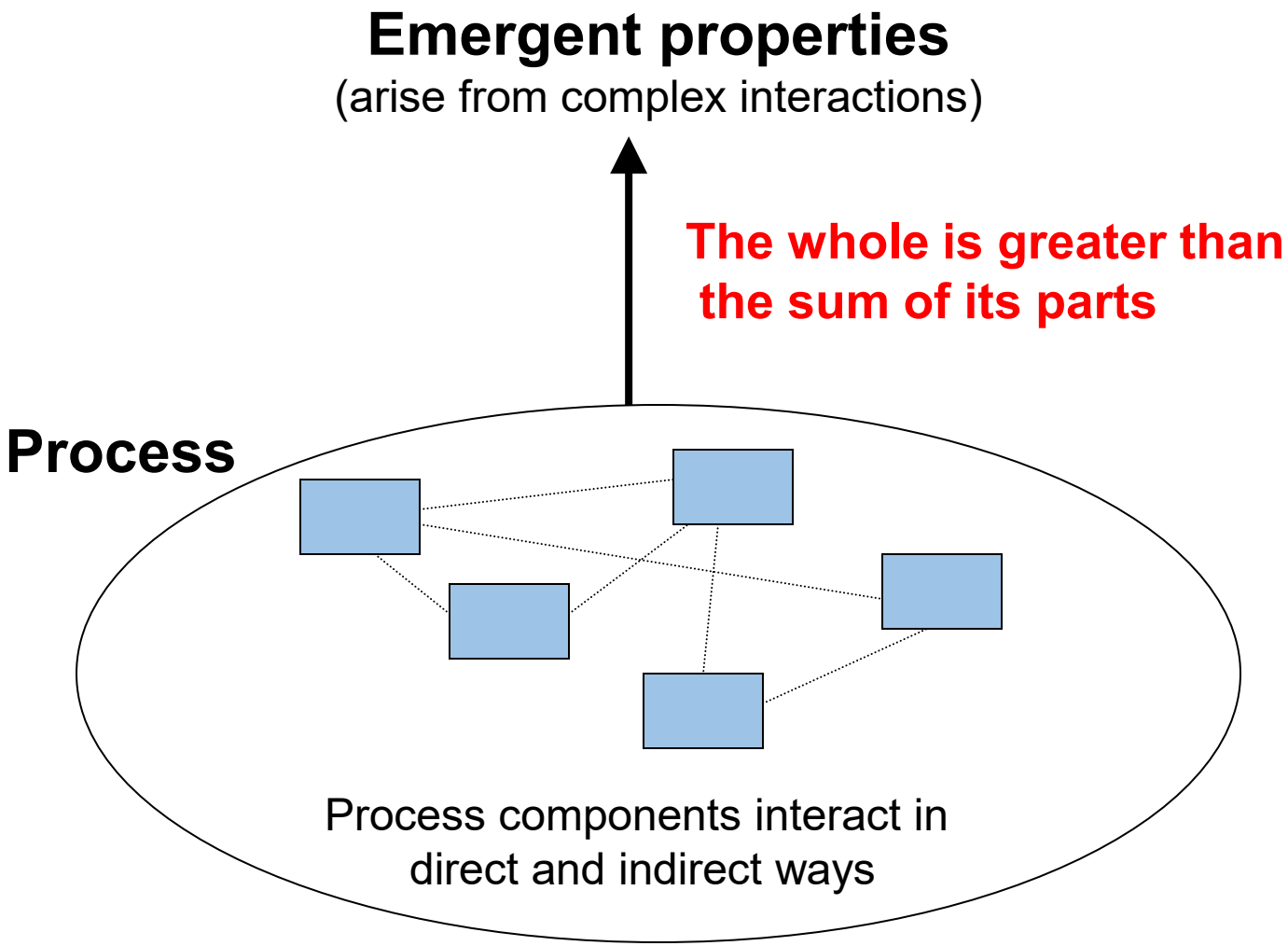
Treat Safety as a **Control** Problem

Systems Theory

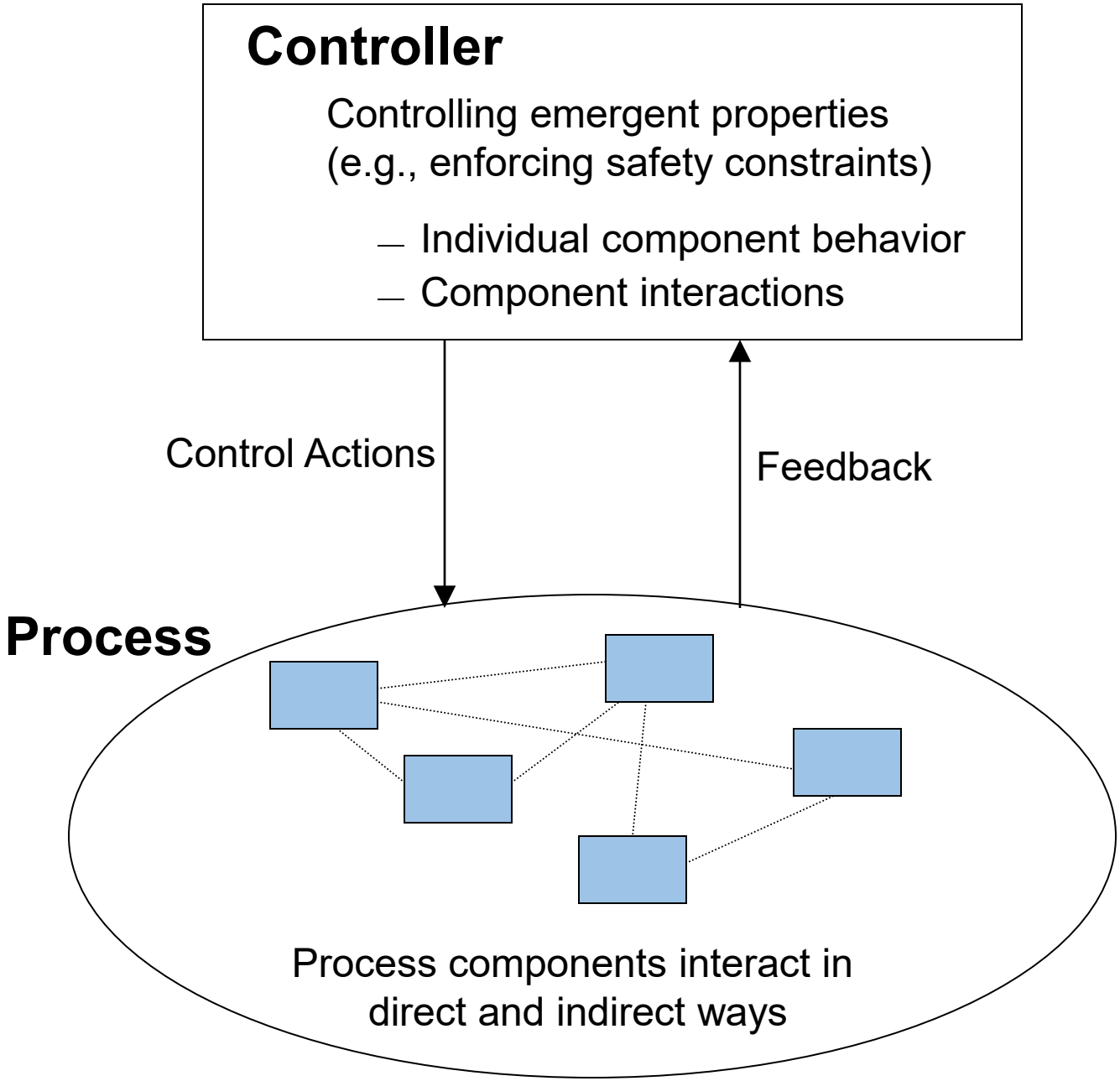
- Developed for systems that are
  - Too complex for complete analysis
    - Separation into (interacting) subsystems distorts the results
    - The most important properties are emergent
  - Too organized for statistics
    - Too much underlying structure that distorts the statistics
    - New technology and designs have no historical information
- First used on ICBM systems of 1950s/1960s

System Theory was created to provide a more powerful way to deal with complexity

System Theory



Safety and security are emergent properties (but not the only ones)


Controller
Controlling emergent properties
(e.g., enforcing safety constraints)
– Individual component behavior
– Component interactions
Control Actions
Feedback
Process
Process components interact in
direct and indirect ways

A Broad View of "Control"

Component failures and unsafe interactions may be "controlled" through design

(e.g., redundancy, interlocks, fail-safe design, …)

or through process

- Manufacturing processes and procedures
- Maintenance processes
- Operational processes

or through social controls

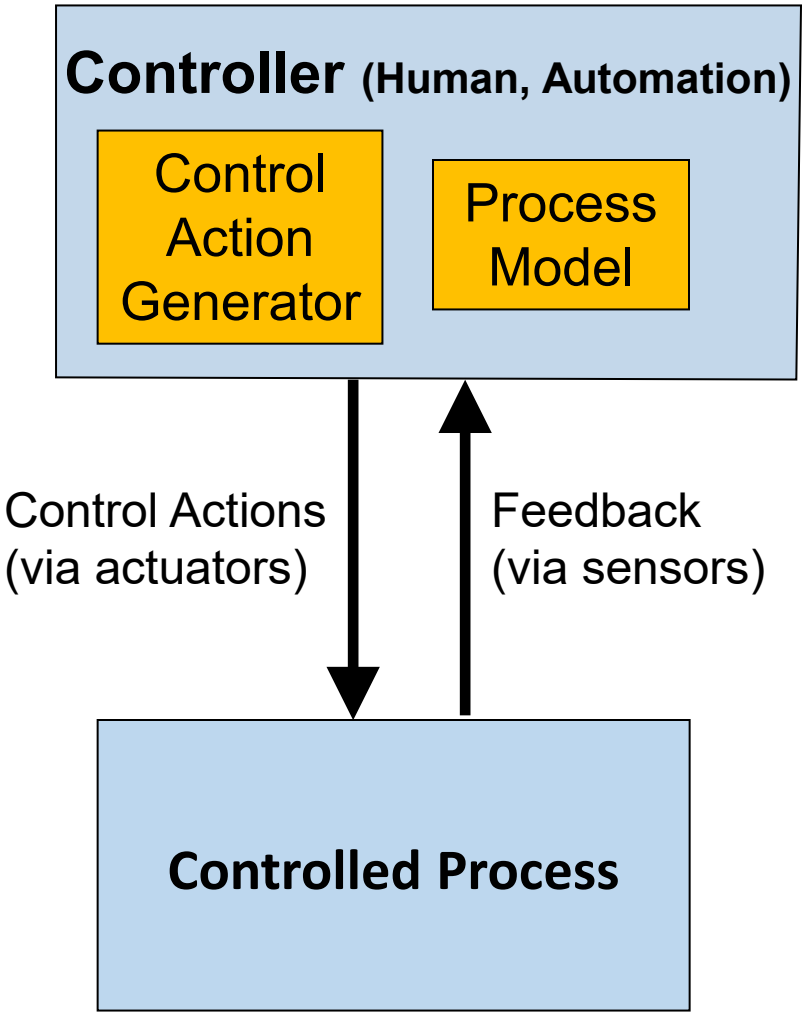
- Governmental or regulatory
- Culture
- Insurance
- Law and the courts
- Individual self-interest (incentive structure)

Controls/Controllers Enforce Constraints

- Aircraft must maintain sufficient lift to remain airborne
- Aircraft, autos must maintain minimum separation
- Public health system must prevent exposure of public to contaminated water, food products, and viruses
- Pressure in a offshore well must be controlled
- Integrity of hull must be maintained on a submarine
- Toxic chemicals/radiation must not be released from plant
- Workers must not be exposed to workplace hazards

These represent the system-level requirements on the sociotechnical system

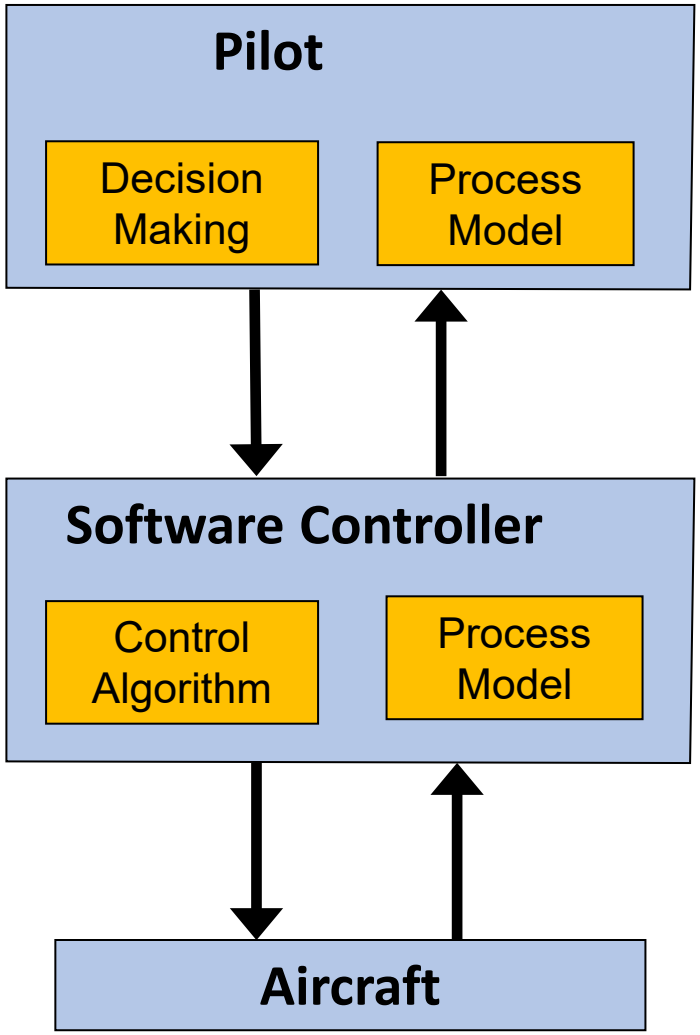
Treating Safety as a Control Problem



- Controllers use a **process model** to determine control actions
- Software/human related accidents usually occur when the process model is incorrect (inconsistent with real state of process)
- Captures software errors, human errors, flawed requirements …

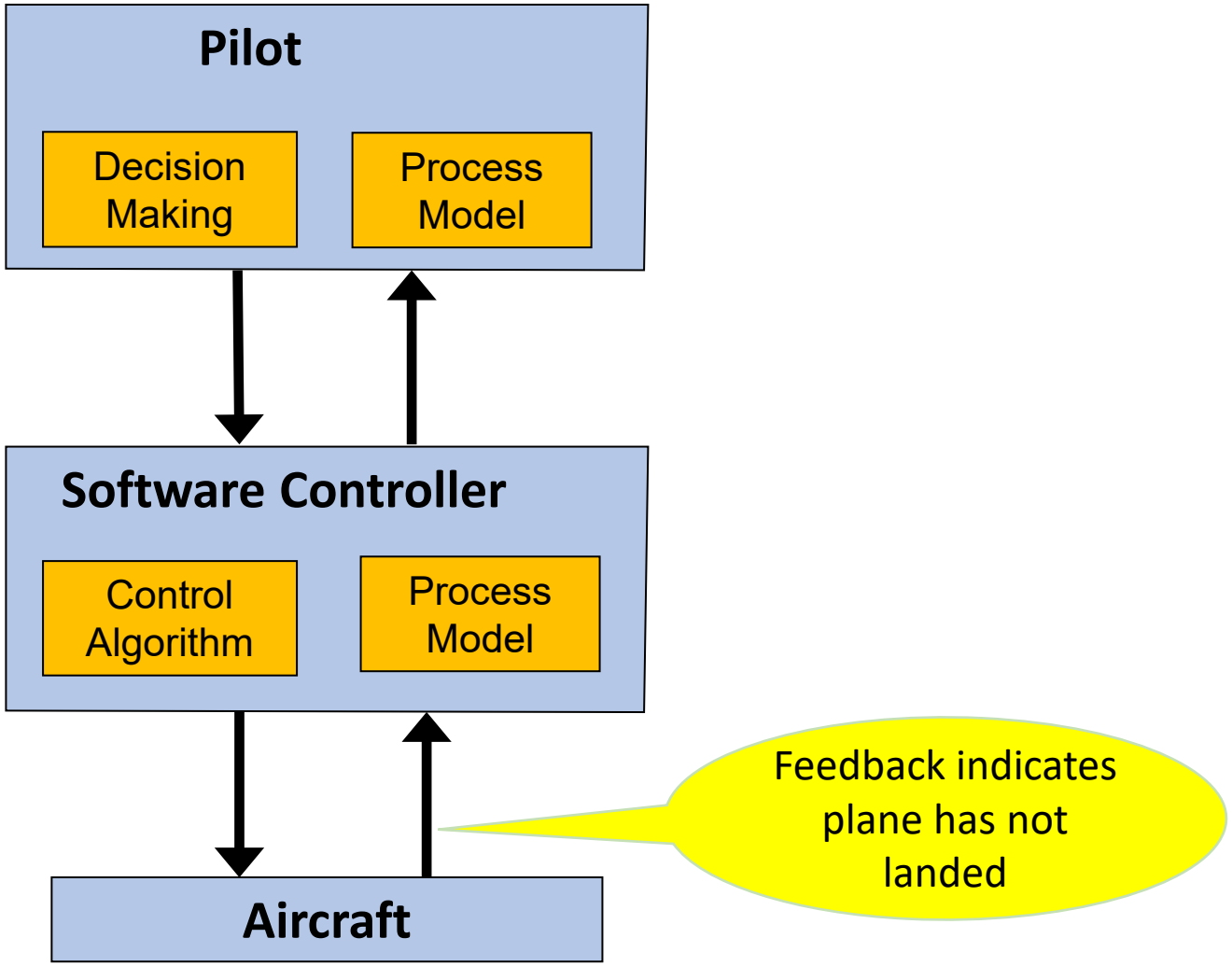
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



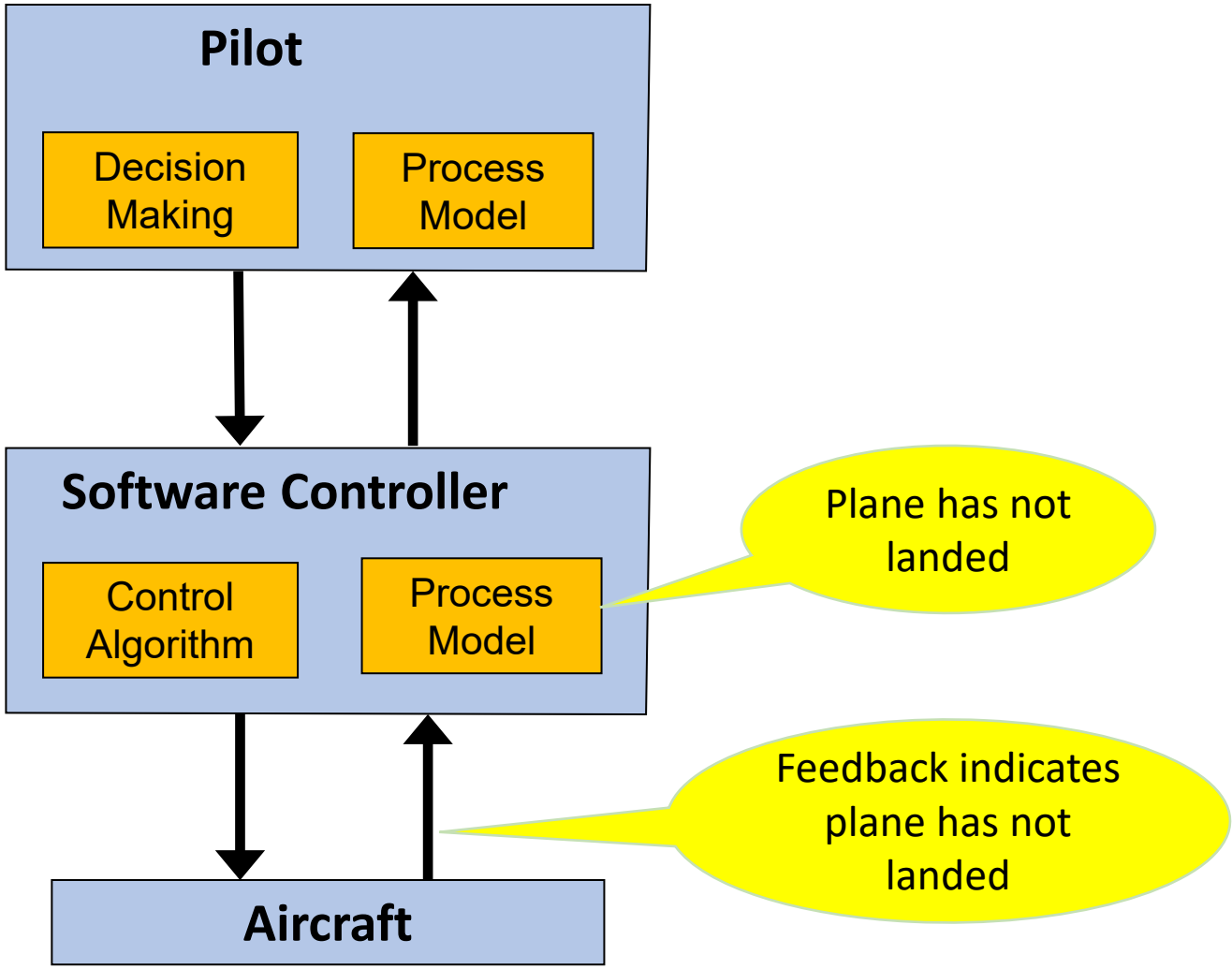
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



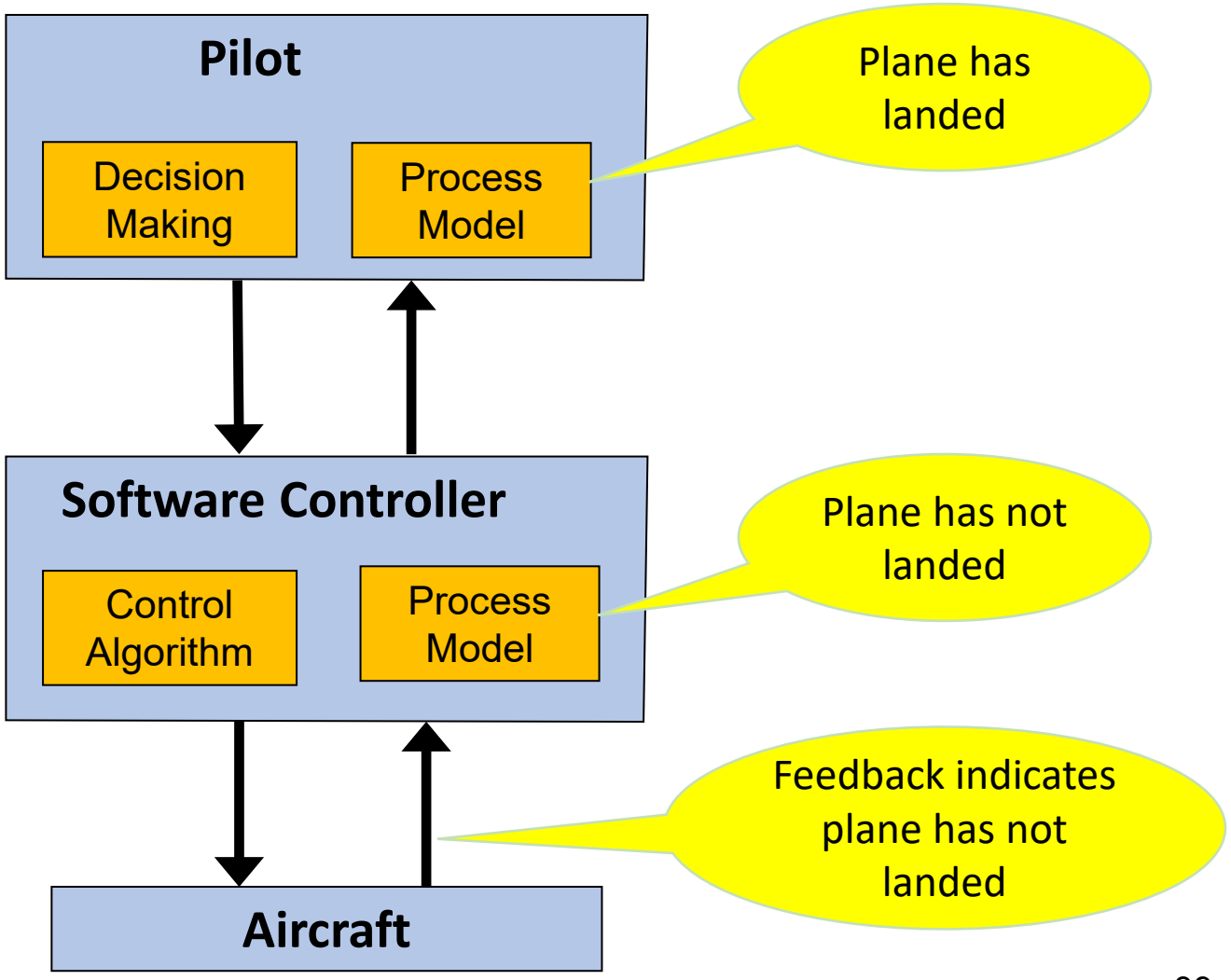
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



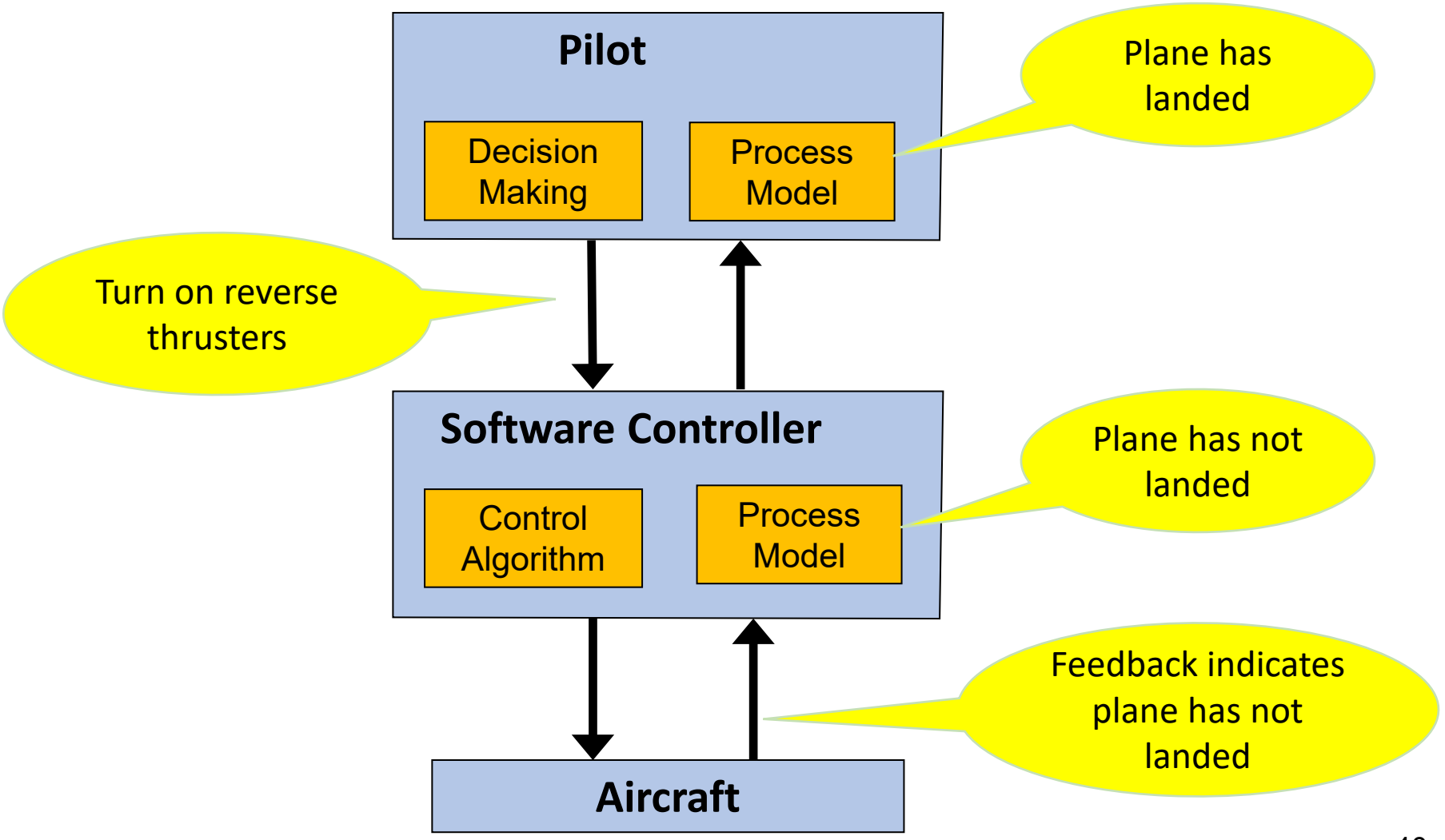
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



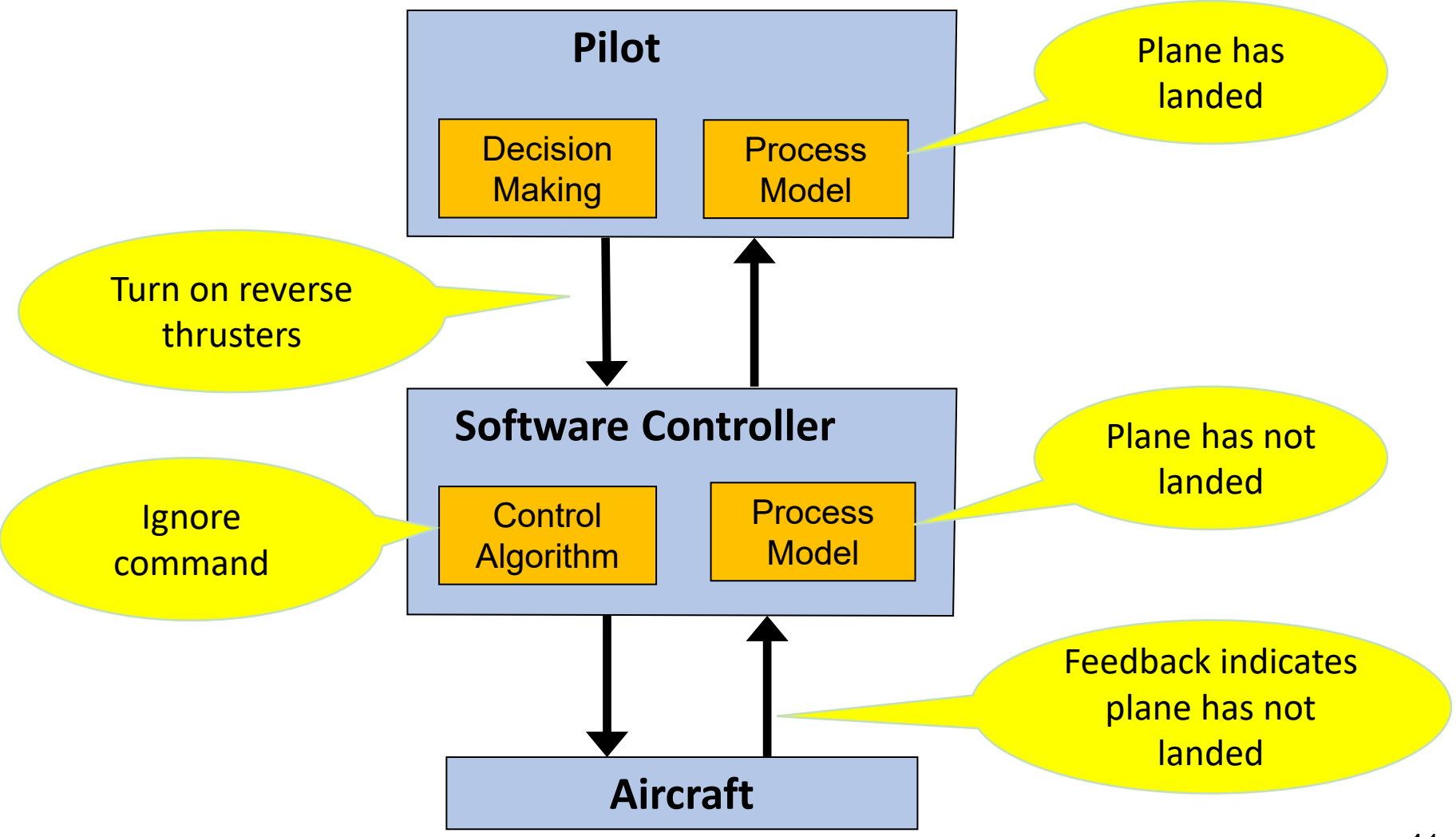
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing

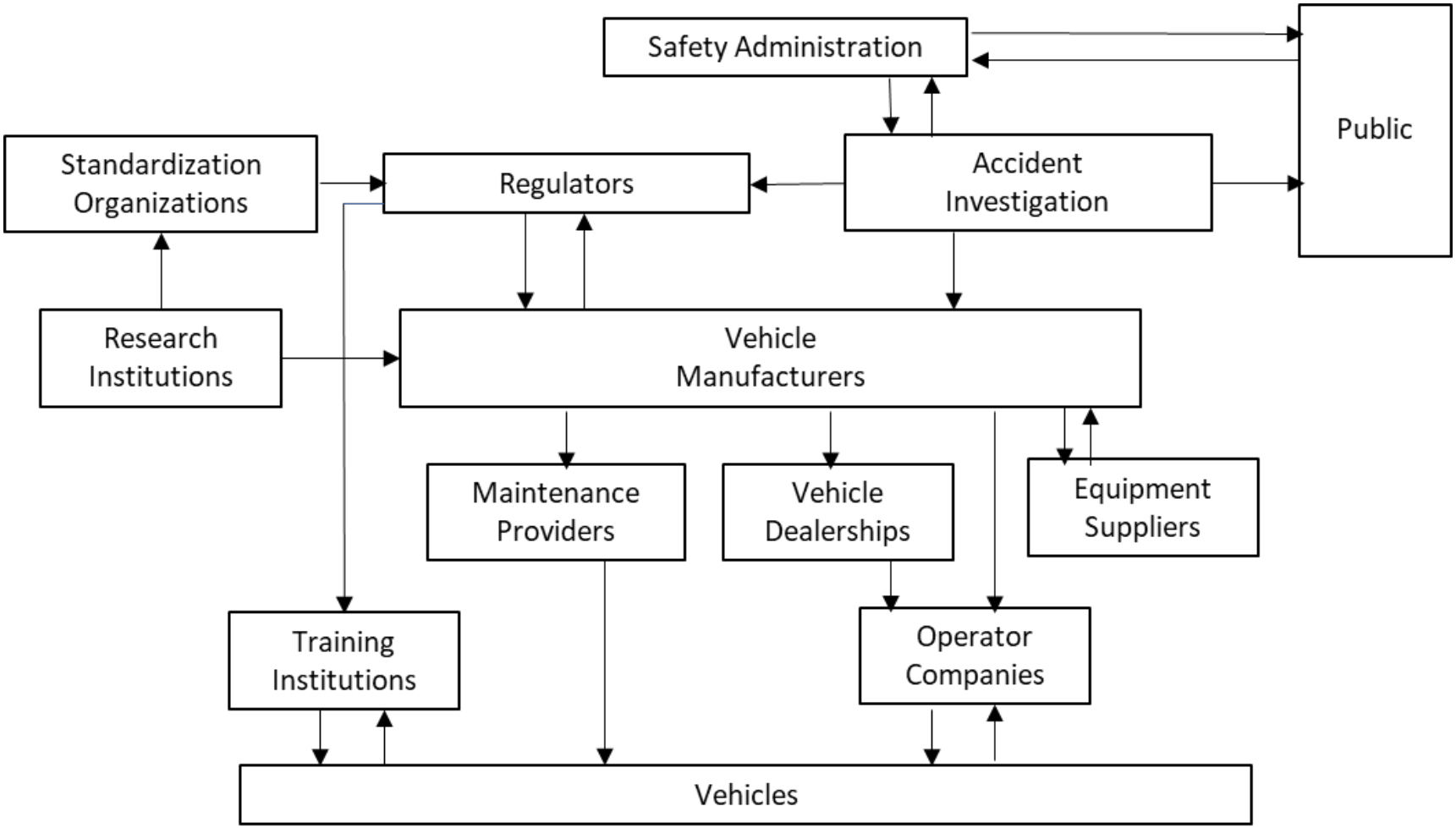


Warsaw (Reverse Thrusters)

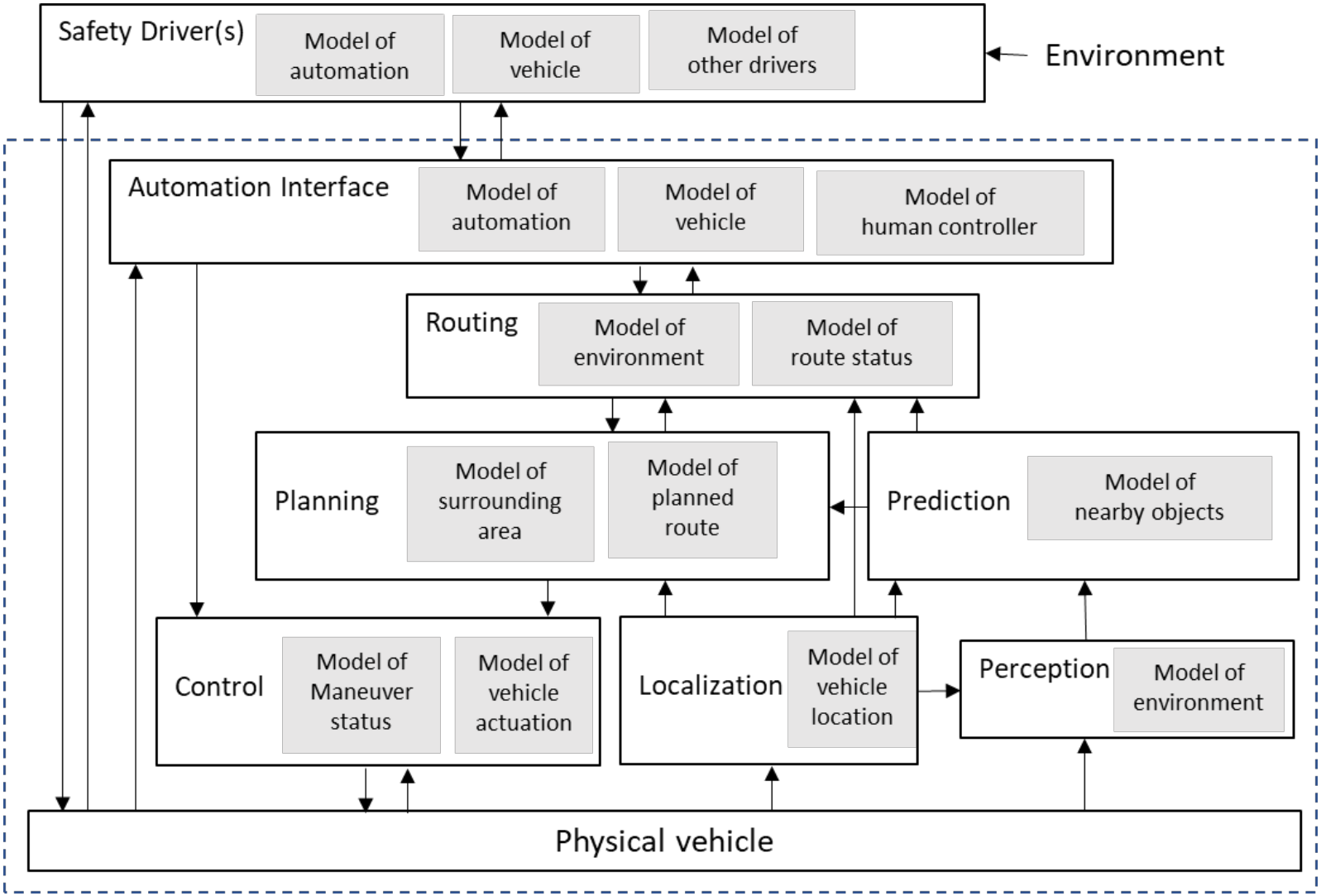
Hazard: Inadequate aircraft deceleration after landing



Models include Social Factors

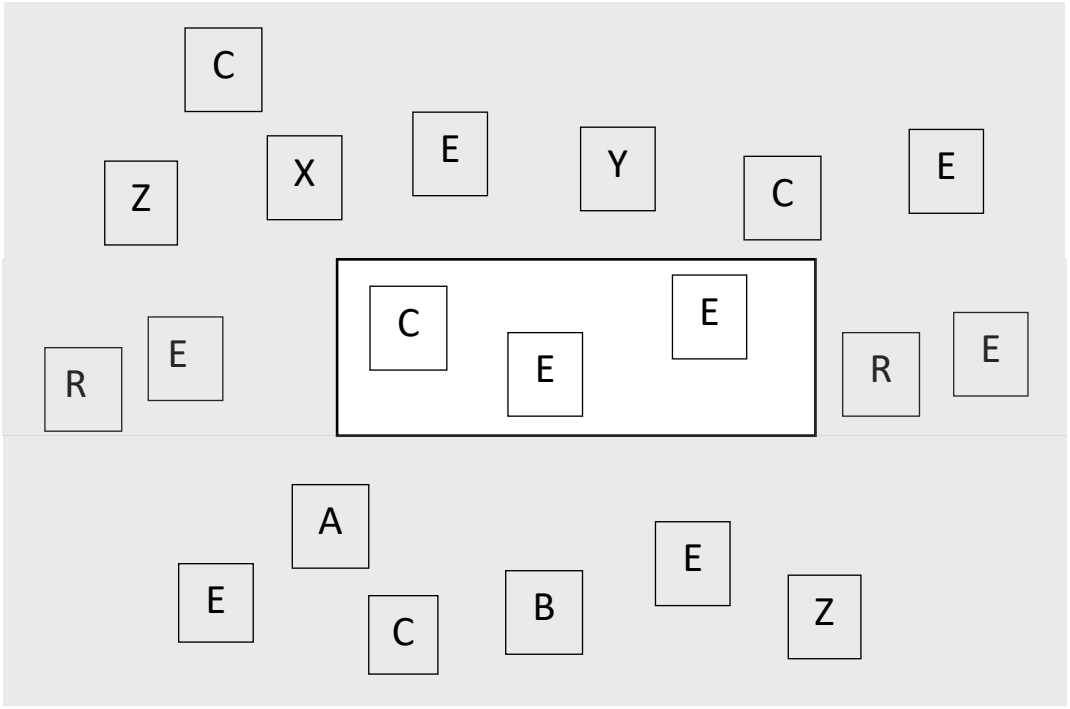


Model of Vehicle



2. New modeling languages and tools

Models Filter Out “Irrelevant” Information (for problem being solved)



Physical Structure of a Spacecraft

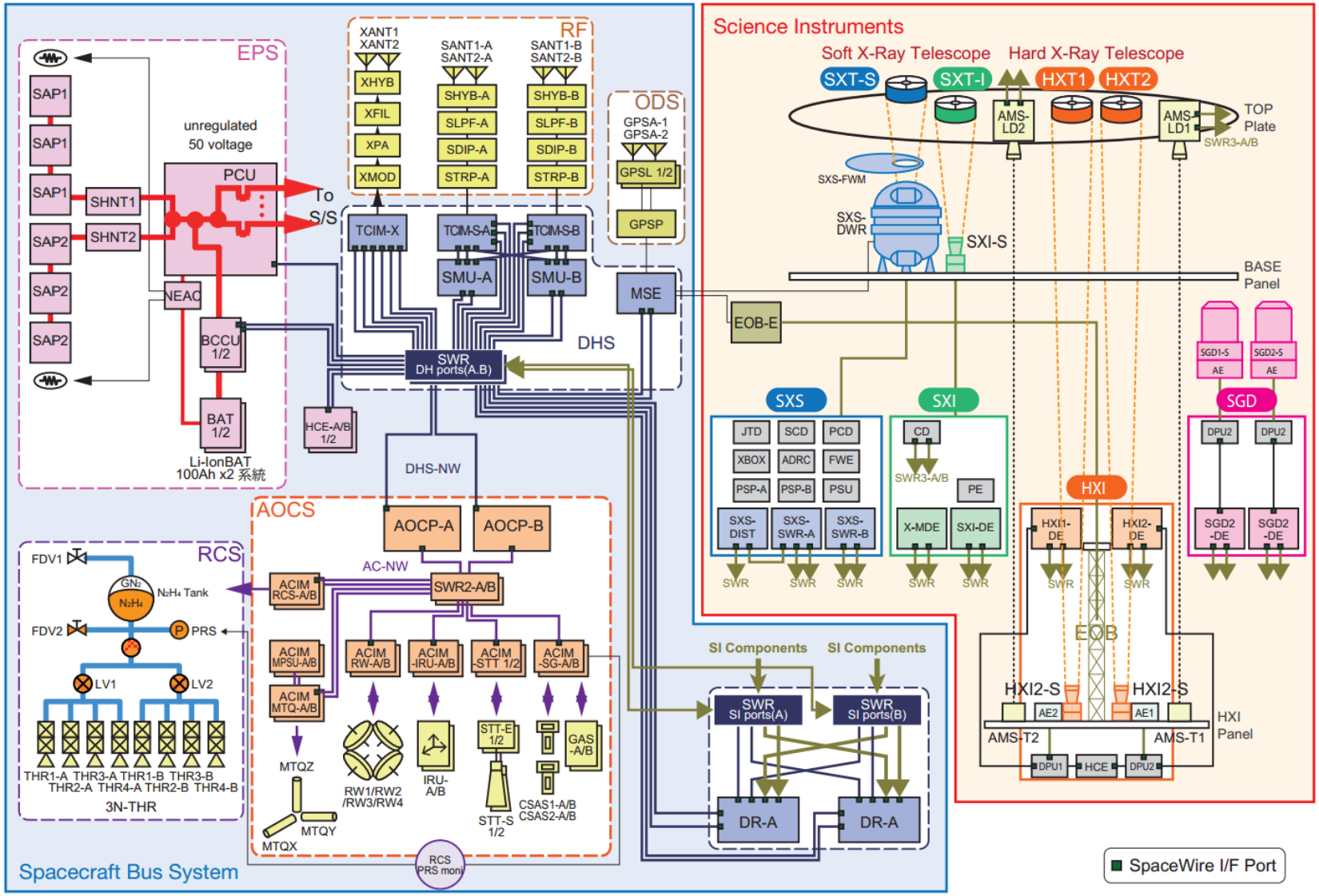
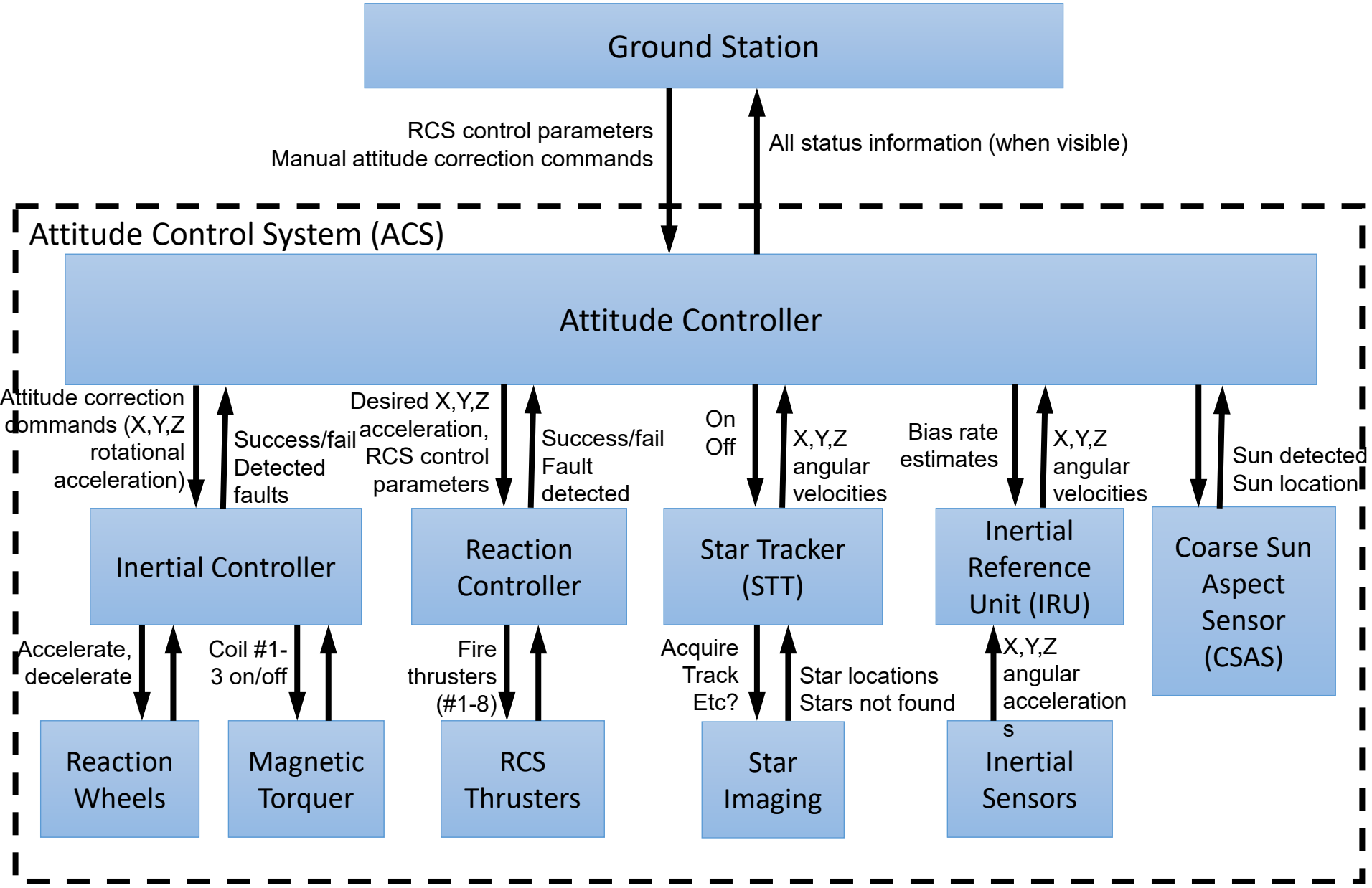


Figure 3.9: System block diagram. A is the primary and B is the redundant system.


Ground Station
RCS control parameters
Manual attitude correction commands
All status information (when visible)
Attitude Control System (ACS)
Attitude Controller
Attitude correction commands (X,Y,Z rotational acceleration)
Success/fail
Detected faults
Desired X,Y,Z acceleration, RCS control parameters
Success/fail
Fault detected
On
Off
X,Y,Z angular velocities
Bias rate estimates
X,Y,Z angular velocities
Sun detected
Sun location
Inertial Controller
Reaction Controller
Star Tracker (STT)
Inertial Reference Unit (IRU)
Coarse Sun Aspect Sensor (CSAS)
Accelerate, decelerate
Coil #1-3 on/off
Fire thrusters (#1-8)
Acquire
Track
Etc?
Star locations
Stars not found
X,Y,Z angular acceleration
s
Reaction Wheels
Magnetic Torquer
RCS Thrusters
Star Imaging
Inertial Sensors

Functional Structure

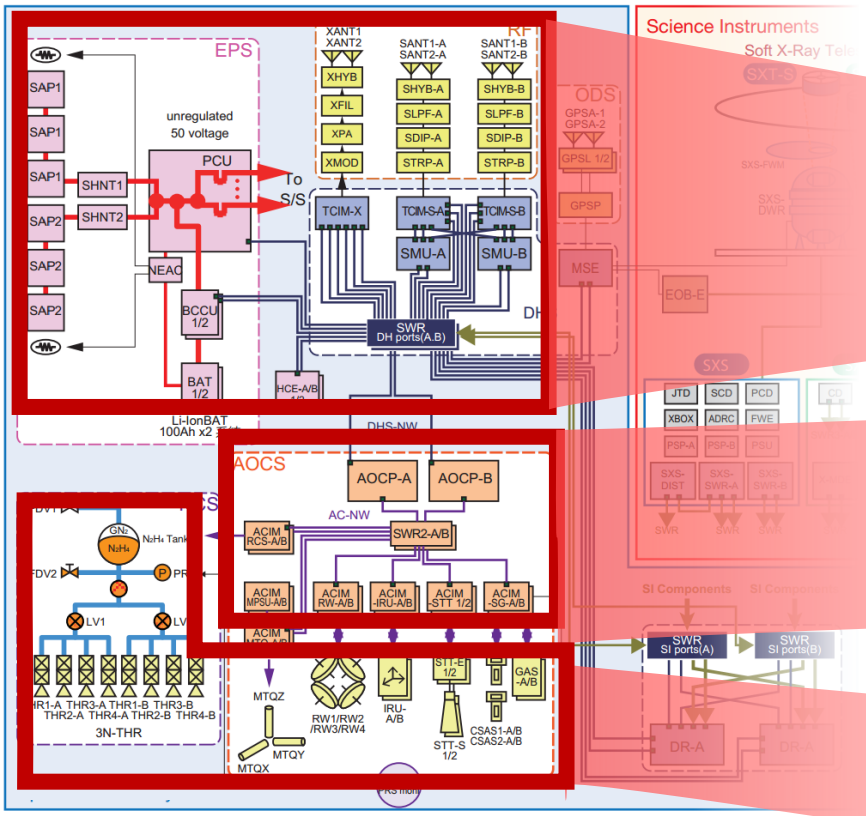
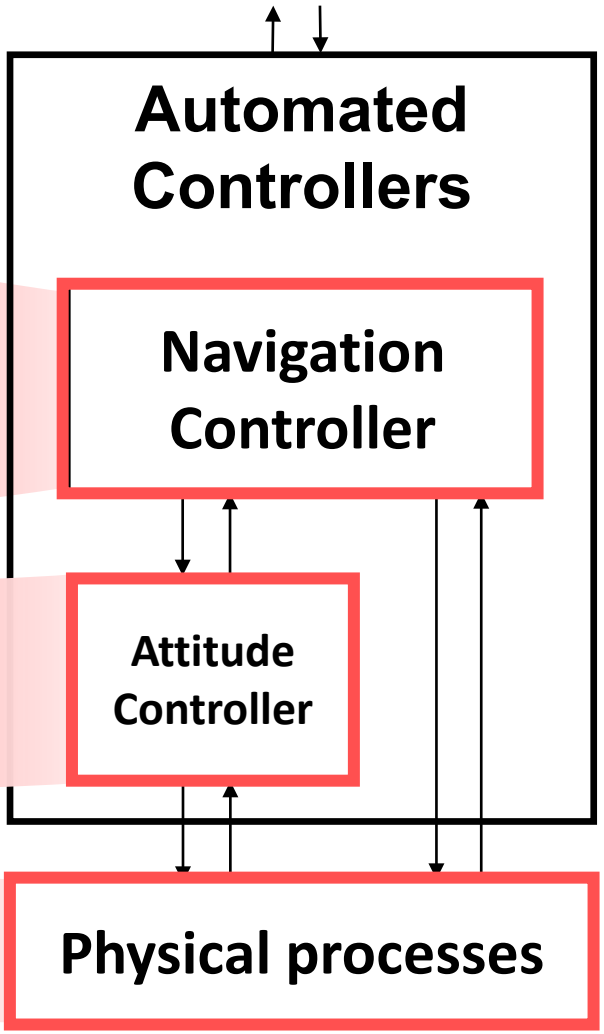
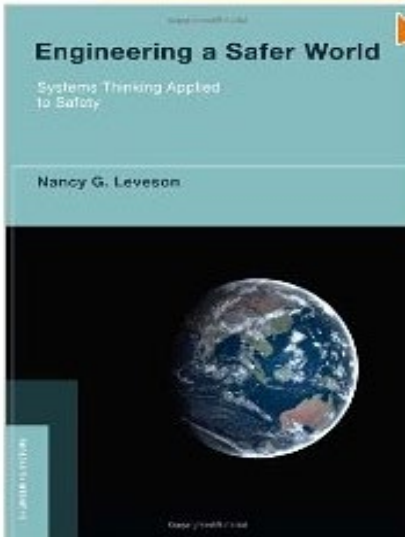


Figure 3.9: System block diagram. A is the primary and B is the redundant syst



The model used will impact the apparent complexity of the system

Does it Work? Extensive Industrial Use of STAMP



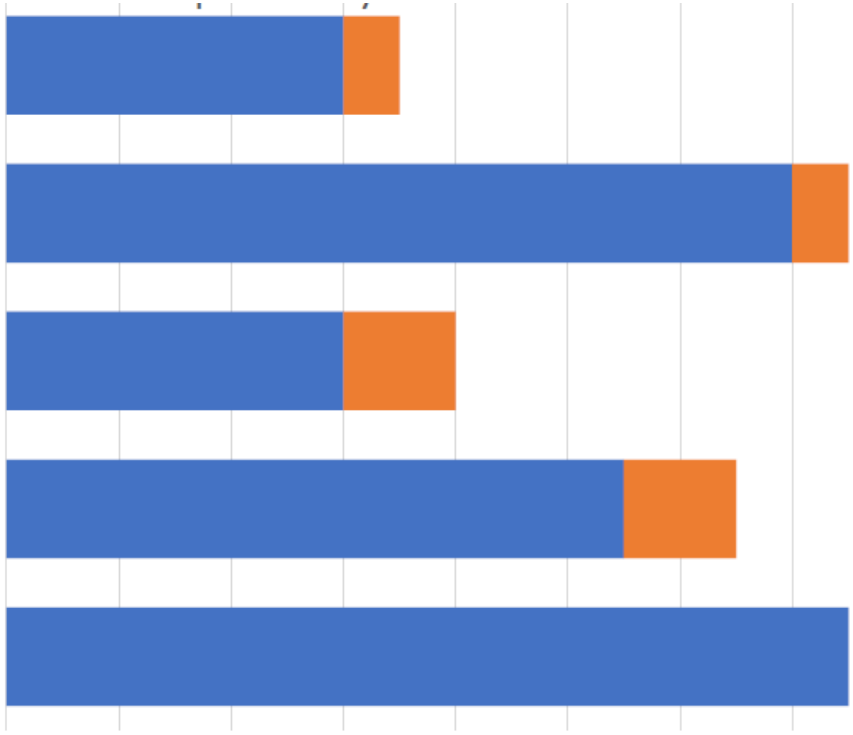
- Book came out about 10 years ago
- 2,000 - 3000 attendees (85 countries) at yearly workshop
- Standards (SAE, ISO, IEEE)
- STPA Handbook
  - 200,000+ downloads in last 3 years
  - Translated into Japanese, Korean, two forms of Chinese
- Hundreds of controlled and empirical (industry) experiments and studies
  - All show more powerful than old techniques
  - Some data to suggest orders of magnitude less expensive

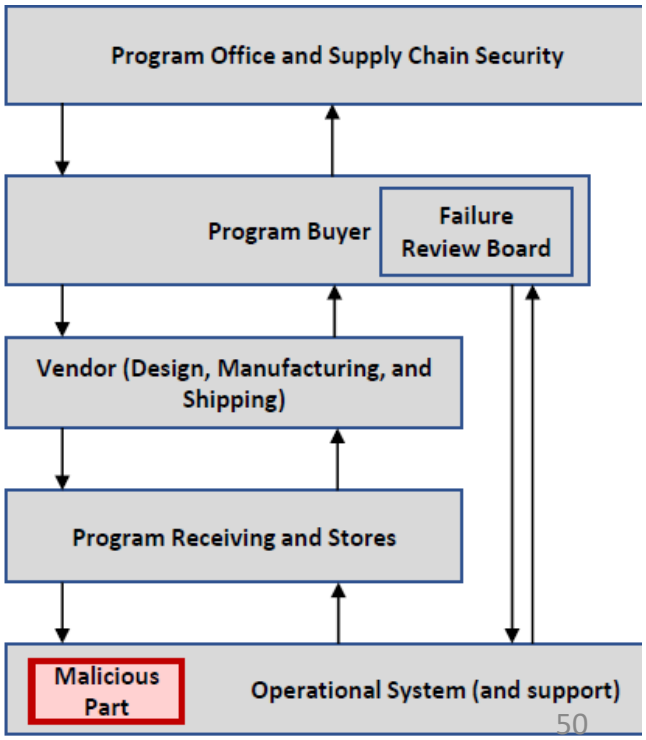
Security Too: STPA applied to one DoD program before SolarWinds attack

Michael Bear (BAE), John Thomas (MIT), Col. William Young (USAF)

- Program that used STPA was protected from SolarWinds
- Vulnerabilities found by STPA

Later exploited by SolarWinds attackers
Not exploited by SolarWinds attackers





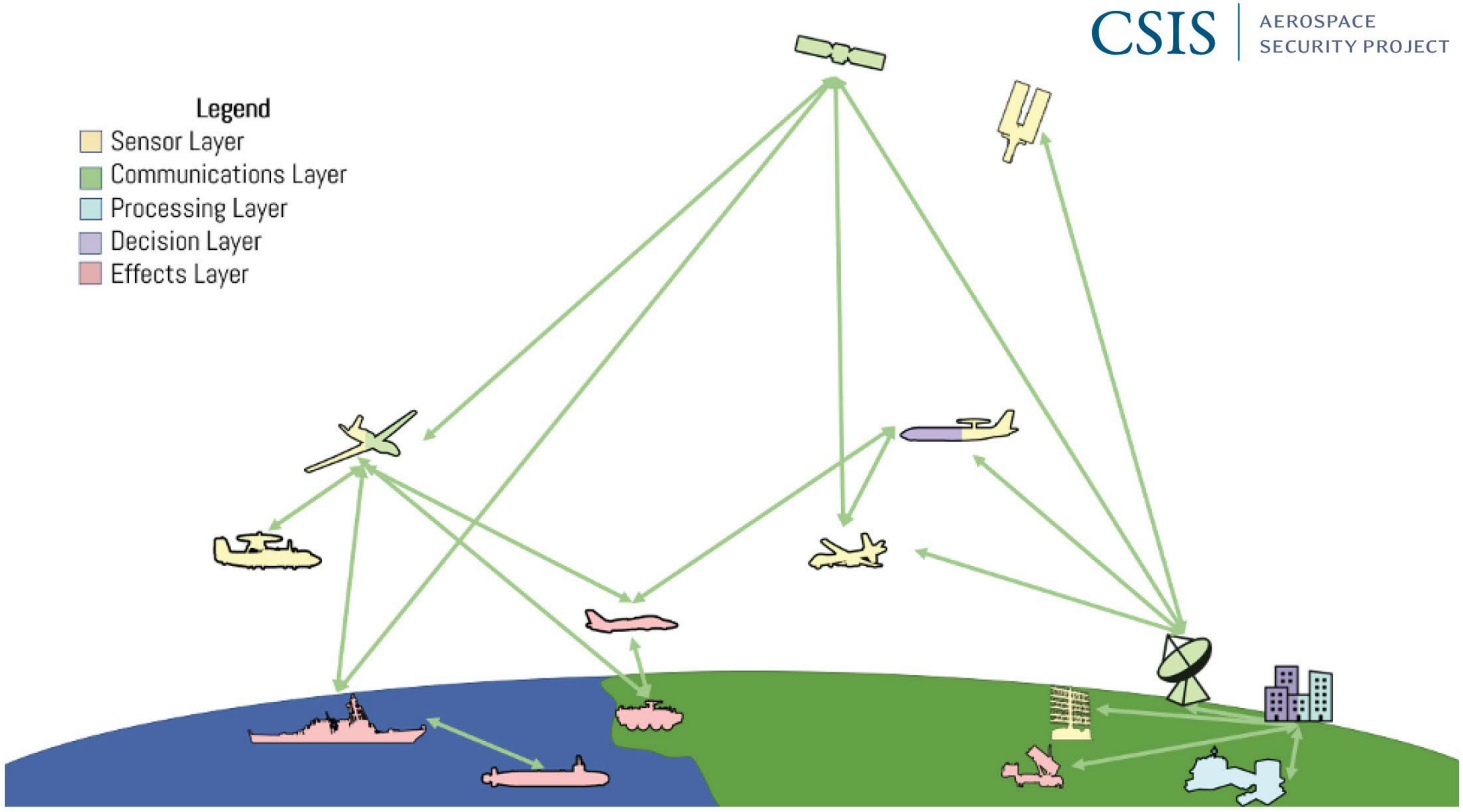
What this paradigm change can produce:

- Top-down design from concept development: create requirements, architecture, and design using STPA analysis to guide tradeoffs
  - Build safety (other qualities) in vs. trying to assure it after the fact
- Leading indicators of increasing risk
- Sophisticated teaming between humans and automation
- Any emergent property: mission assurance, security, serviceability, quality
- Etc.

(Kopeikin)

Envisioned Systems of the Future

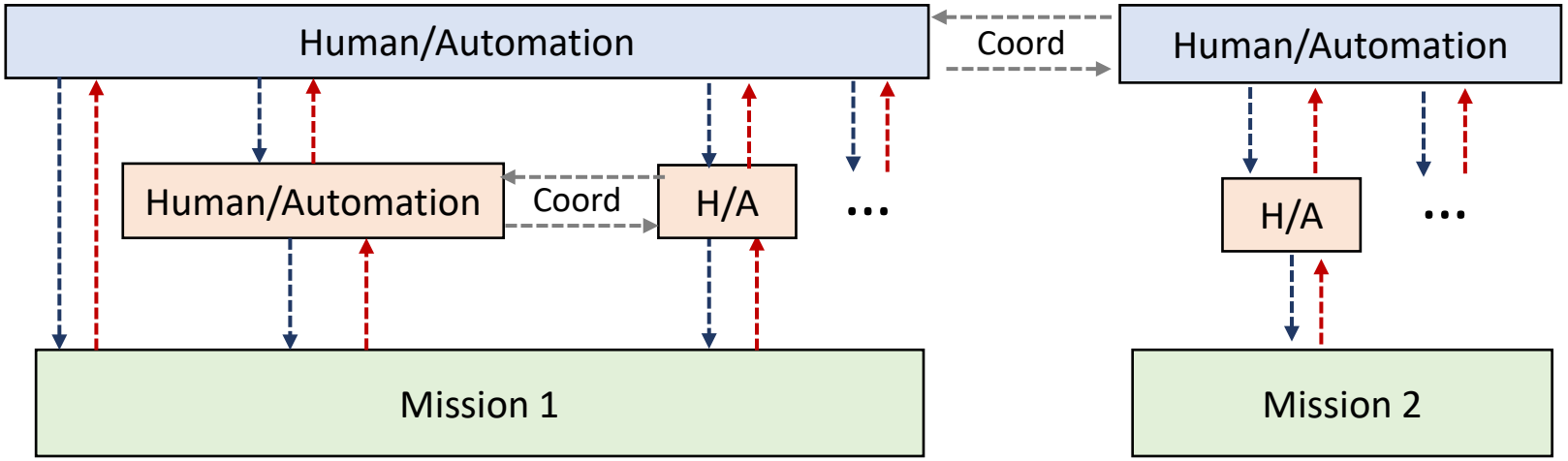
- Complex interactions among humans and automation beyond current modeling, analysis, design, and assurance methods



Source: Based on author's own creation.

Complex Human–Machine Teaming
Lt. Col. Andrew Kopeikin, USAF

MUM-T (Manned-Unmanned Teaming)



- Lateral coordination
- Dynamic connectivity
- Transfer of authority (dynamic authority)
- Shared authority

- Dynamic hierarchy
- Dynamic membership
- Mutually closing loops
- Cognitive alignment
- …

Bottom-Line Up Front (BLUF)

- Our methods and tools are being swamped by the complexity in the systems we are trying to build.
- New causes of losses appearing (not just component failure).
- To make progress, we need a paradigm change in
  - Engineering for safety
  - MBSE models and tools
  - General system engineering approaches (FUSE)