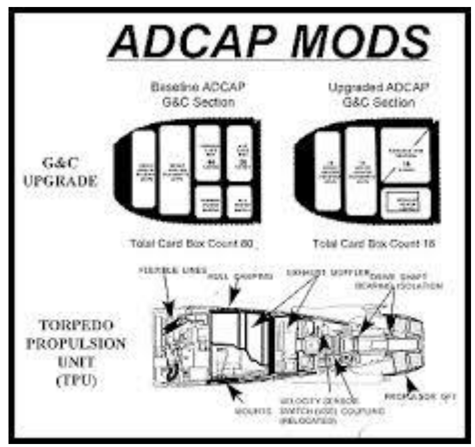



The Need for a New Paradigm In System Safety Engineering

Prof. Nancy G. Leveson

Aeronautics and Astronautics
MIT




ADCAP MODS
Baseline ADCAP
G&C Section
Upgraded ADCAP
G&C Section
G&C
UPGRADE
Total Card Box Count 80
Total Card Box Count 16
TORPEDO
PROPULSION
UNIT
(TPU)
MOUNTS

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)

Some Painful Truths

- Traditional safety and security techniques don’t work on today’s systems
  - Tomorrow will be worse
- They cannot be extended to make them work
- A paradigm change is needed to leap the hurdles we face

Why do losses occur today?

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."

Traditional Approach to Safety Engineering

- Assume accidents caused by chains of failure events
- Forms the basis for most safety engineering and reliability engineering analysis:

  FTA, PRA, FMEA/FMECA, Event Trees, FHA, etc.

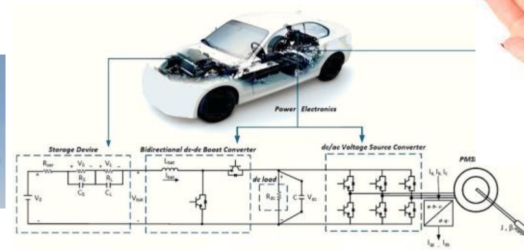
- Evaluate reliability of components separately and later combine analysis results into a system reliability value
  - Assumes losses caused by component failure,
  - Assumes independence of failures
  - Assumes randomness—do software and humans behave this way?

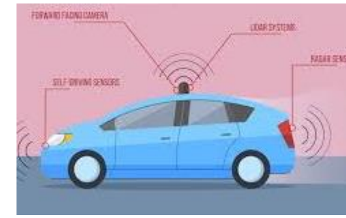
Traditional Approach to Safety Engineering (1)

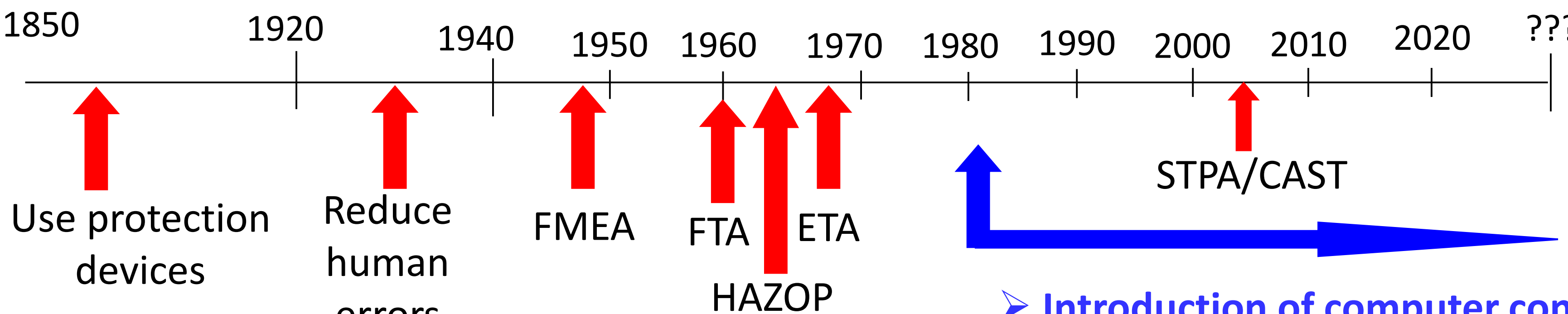
- Design (concentrate on dealing with component failure):
  - Redundancy and barriers (to prevent failure propagation),
  - High component integrity and overdesign,
  - Fail-safe design,
  - (humans) Operational procedures, checklists, training, ….

- Operations
  - Focus on compliance
  - Accident Analysis (mostly blamed on human operators)

History of System Safety Engineering







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

Assume accidents caused by lack of design protection

Assume accidents caused by human errors

Assume accidents caused by component failures: Problem is component reliability

?????

What Failed Here?

- Navy aircraft were ferrying missiles from one location to another.
- One pilot executed a planned test by aiming at aircraft in front and firing a dummy missile.
- Nobody involved knew that the software was designed to substitute a different missile if the one that was commanded 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.

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.

Lesson Learned

- Accidents today do not just result from component failures.
- Need to consider design errors

Software has Revolutionized Engineering (1)

1. Software does not “fail”

General Purpose Machine + Software = Special Purpose Machine

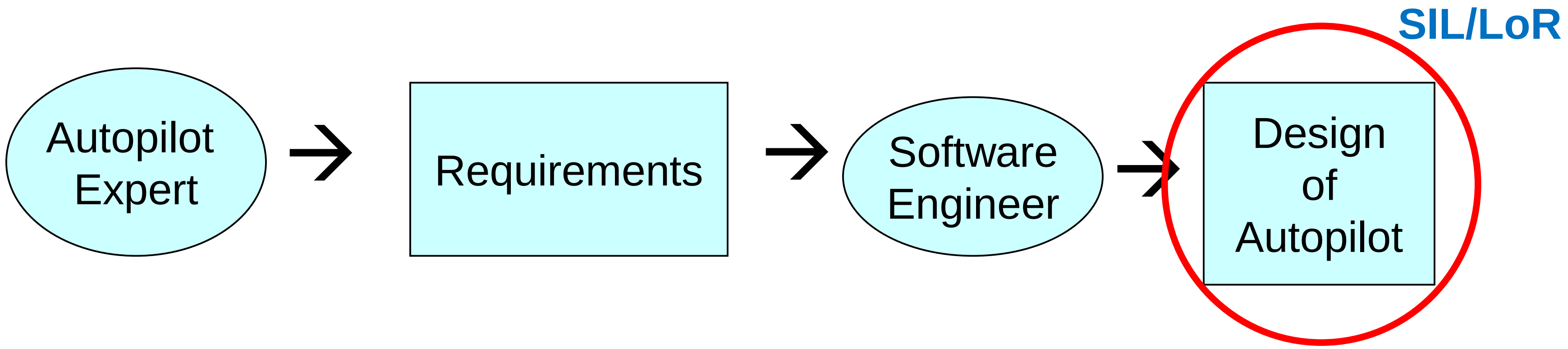
Software is simply the design of a machine abstracted from its physical realization

Software is pure design and designs do not “fail”

2. Software allows almost unlimited complexity (coupling)

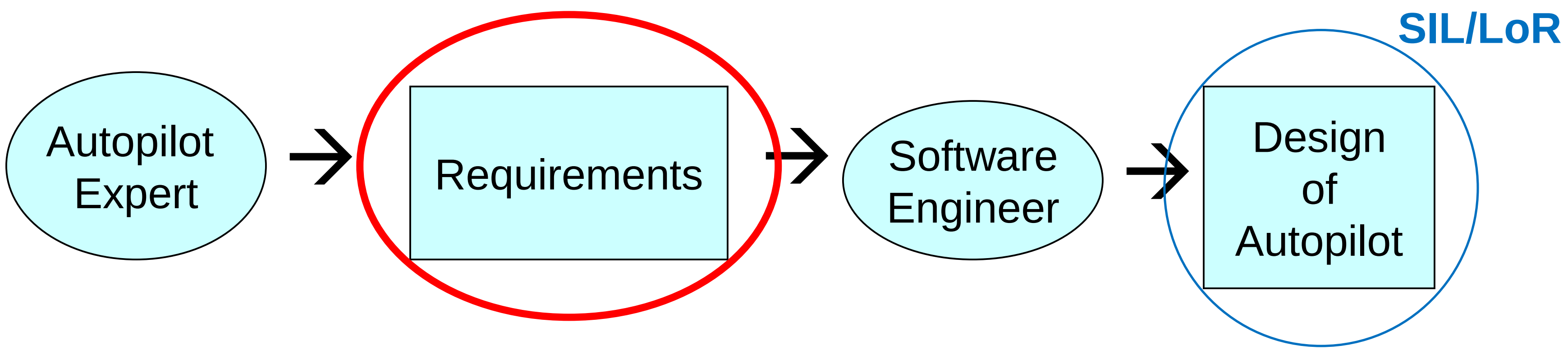
Software engineering focuses on implementing the requirements and validating it

- Ensure rigor placed on design and test



The role of software in accidents almost always involves flawed requirements

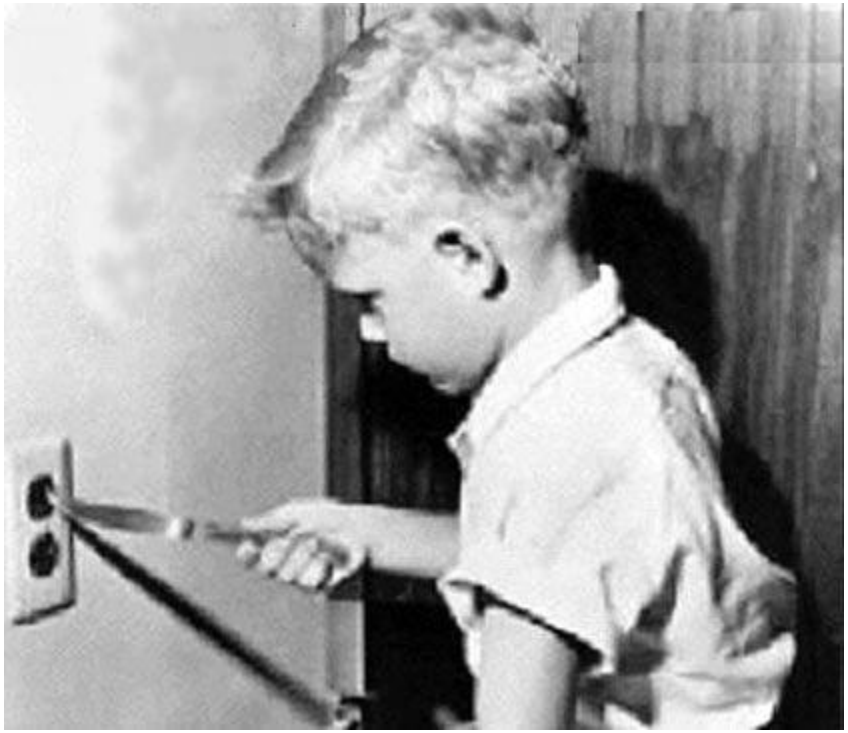
- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions



- 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**

Is this knife safe?

Safety Depends on Context



Example: Safety Depends on Context

Ariane 4 IRS (Inertial Reference Software)

Ariane 5 IRS (reused same software)

Lesson Learned

- Software
  - Contributes differently to accidents than hardware
    - Does not "fail" but can contribute to unsafe system behavior (including unsafe human behavior)
  - Adds almost unlimited complexity but
    - Cannot exhaustively test
    - Is not by itself safe or unsafe

Software changes the role of humans in systems

Typical assumption is that operator error is cause of most incidents and accidents

- So do something about operator involved (admonish, fire, retrain them)
- Or do something about operators in general
  - Marginalize them by putting in more automation
  - Rigidify their work by creating more rules and procedures

"Cause" from the American Airlines B-757 accident report (in Cali, Columbia):

> "Failure of the flight crew to revert to basic radio navigation at the time when the FMS-assisted navigation became confusing and demanded an excessive workload in a critical phase of flight."

Another Accident Involving Thrust Reversers

- Tu-204, Moscow, 2012
- Red Wings Airlines Flight 9268
- The soft 1.12g touchdown made runway contact a little later than usual.
- With the crosswind, this meant weight-on-wheels switches did not activate and the thrust-reverse system would not deploy.

Another Accident Involving Thrust Reversers

- Pilots believe the thrust reversers are deploying like they always do. With the limited runway space, they quickly engage high engine power to stop quicker. Instead this accelerated the Tu-204 forwards, eventually colliding with a highway embankment.

Another Accident Involving Thrust Reversers

- Pilots believe the thrust reversers are deploying like they always do. With the limited runway space, they quickly engage high engine power to stop quicker. Instead this accelerates the Tu-204 forwards, eventually colliding with a highway embankment.

In complex systems, human and technical considerations cannot be isolated

Human factors concentrates on the “screen out”

Hardware/Software engineering concentrates on the “screen in”

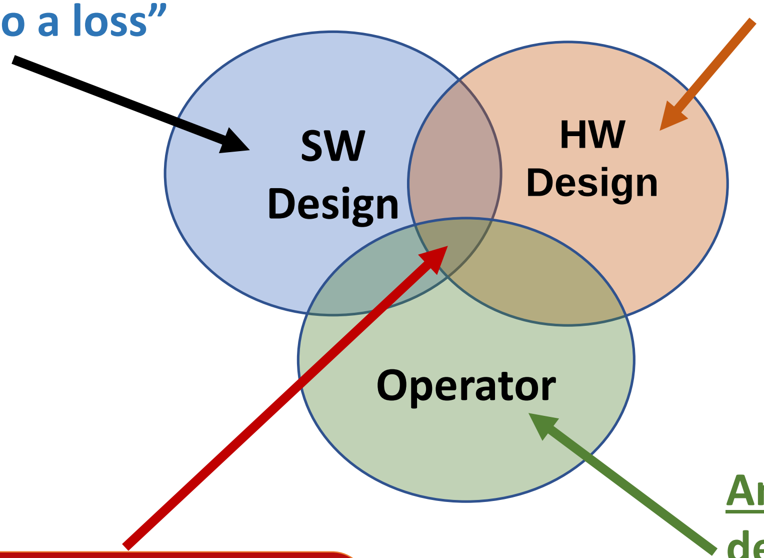
Not enough attention on integrated system as a whole

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

Easy to overlook the system problems when break up system analysis problem

<u>Analysis</u>: "How can SW contribute to a loss"

<u>Analysis</u>: "How can HW fail?"



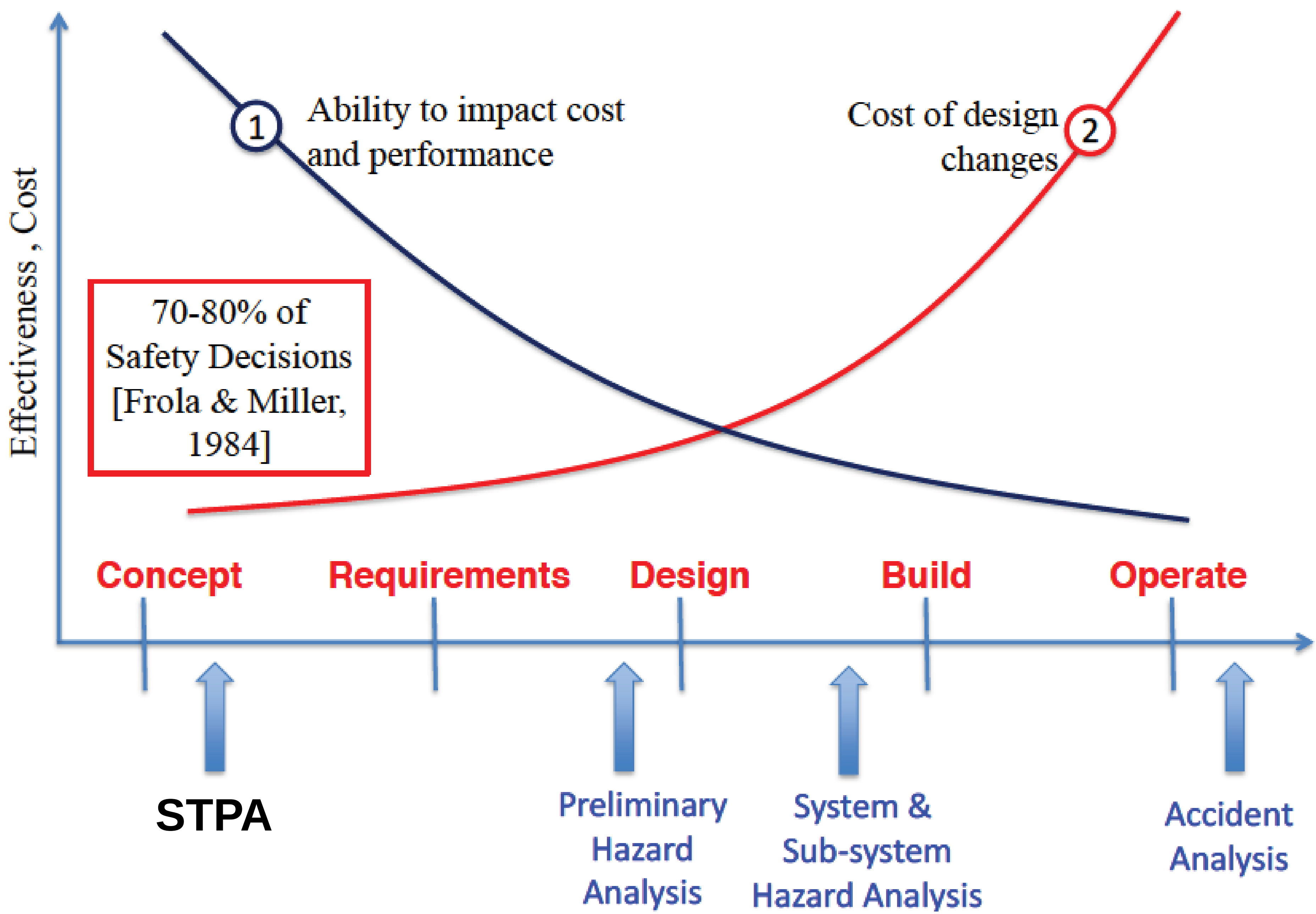
<u>Analysis</u>: "How can the operator deviate from intended/specified procedures?"

New, unplanned interactions in integrated system

Need to look at integrated system as a whole

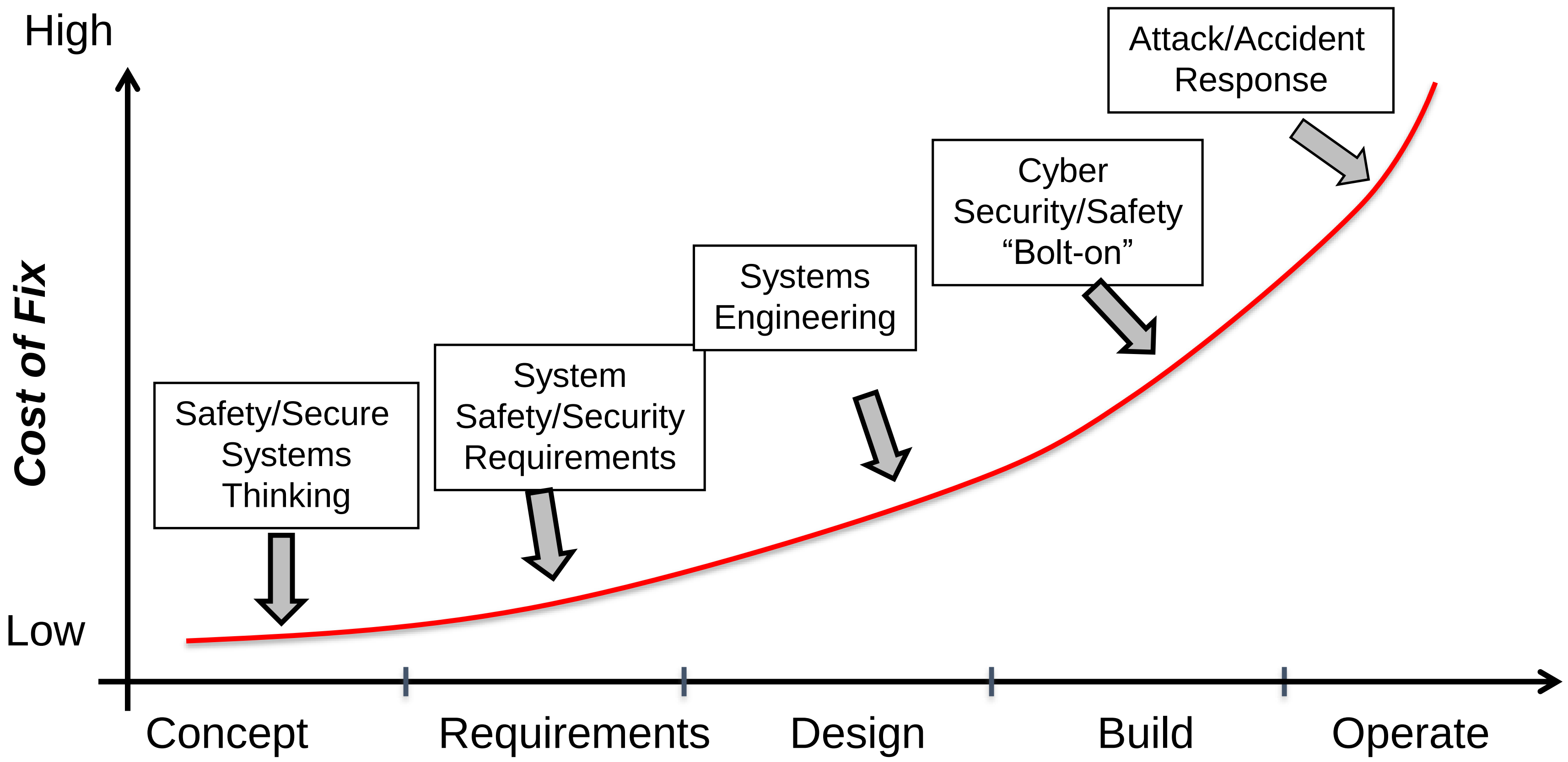
Lesson Learned

- Cannot effectively reduce accidents without integrating human/software/hardware engineering



Effectiveness , Cost
1
Ability to impact cost
and performance
2
Cost of design
changes
70-80% of
Safety Decisions
[Frola & Miller,
1984]
Concept
Requirements
Design
Build
Operate
STPA
Preliminary
Hazard
Analysis
System &
Sub-system
Hazard Analysis
Accident
Analysis

Build safety and security into system from beginning



Lesson Learned

- Can no longer wait until design completed to analyze its safety.
- Need to build safety into systems from the beginning

Focus on Identifying a Root or Probable Cause

- The cause of all accidents is not the events but <u>why</u> the events occurred

- B737 MAX

  Quote from Muilenberg (CEO of Boeing):

  - "Accidents always involve a chain of events"
  - "Pilots were in chain of events as was MCAS"
  - "MCAS added to workload of pilots"
  - "We can break chain of events that led to both crashes by developing a software fix that would limit the potency of that stabilization system"

- Is that really the "root" cause of the B737 MAX accidents?

- Are we missing deeper issues—<u>why</u> the events occurred—that then are never eliminated?

Focus on Identifying a Root or Probable Cause

- While software needs to be fixed, are there not deeper causes that also were involved?
  - Impact of competitive pressures with Airbus A320neo on Boeing management decision making?
  - Was lack of redundancy in AOA sensor simply a random mistake of a design engineer?
  - What was the impact of certification procedures?
  - Inadequate resources of FAA?
  - Changes in regulatory policies and procedures that changed over time to give Boeing more autonomy?
  - Role of system engineering processes and procedures?

- Need to fix the deeper causes

Systemic Factors in Laboratory Data Errors

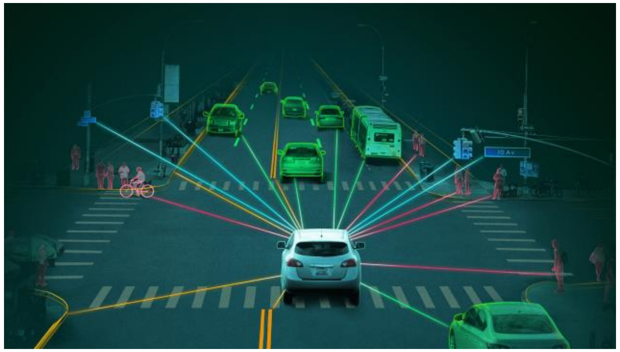
- Decentralized and missing oversight
- Inadequacies and gaps in standards
- Inaccurate perception of risks in use of laboratory data and use of health IT
- Lack of systems view leading to unintended consequences
- Inadequate regulatory emphasis on safety of health IT
- Flawed communication and coordination (missing formal communication channels, missing feedback and error reporting, misidentification of patients, missing information)

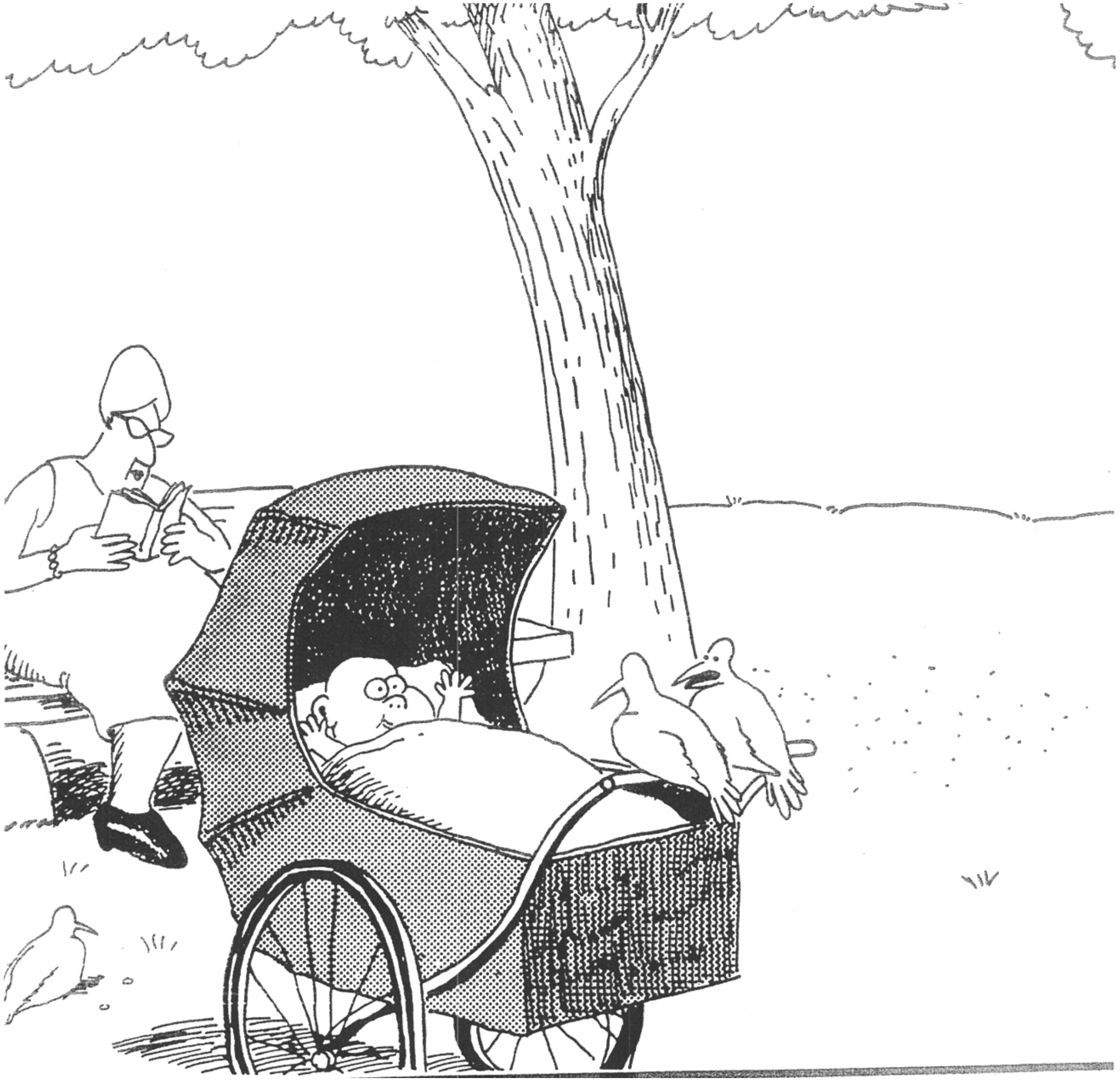
Lesson Learned

- Need to look beyond events to prevent accidents
  - *Why* did events occur?
  - To learn, we need to look at:
    - Conditions that lead to the events
    - Systemic factors that influence almost everything but not necessarily directly related (cannot just draw an arrow or assume a "failure")
- Cannot concentrate only on physical system
  - Need to look at role of social/managerial factors in losses

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





It’s still hungry … and I’ve been stuffing worms into it all day.

Two Approaches Being Taken Now

Pretend there is no problem

Shoehorn new technology and new levels of complexity into old methods

We need something new!

The Problem is Complexity

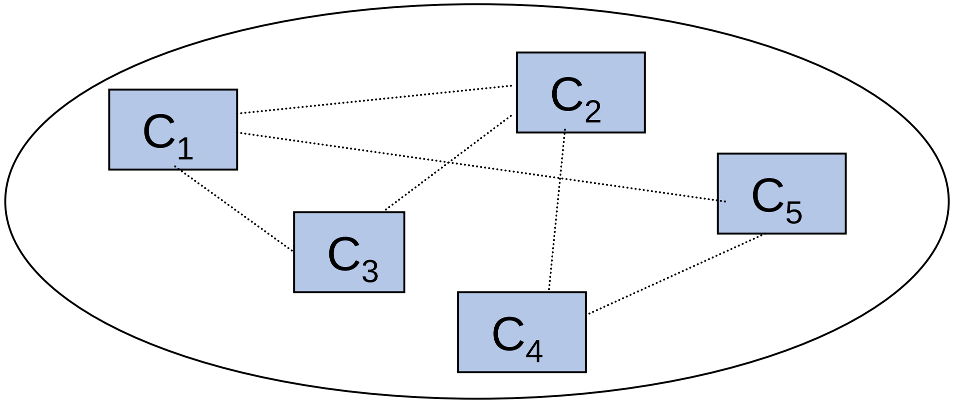
Ways to Cope with Complexity

- Analytic Decomposition
- Statistics
- Systems Theory

Analytic Decomposition ("Divide and Conquer")

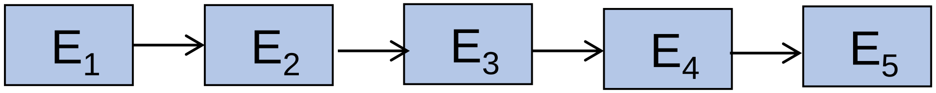
1. Divide system into separate parts

Physical/Functional: Separate into distinct components



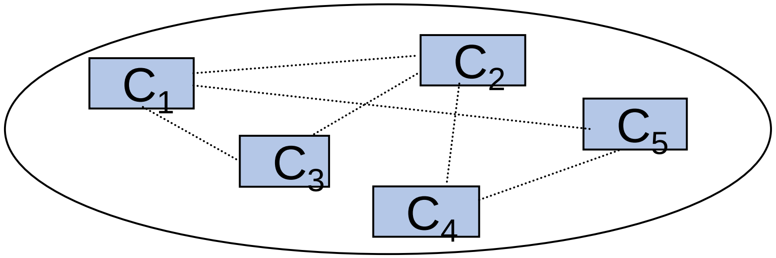
Components interact
In direct ways

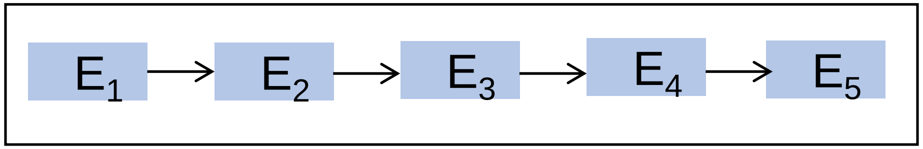
Behavior: Separate into events over time



Each event is the direct
result of the preceding event

Analytic Decomposition (2)

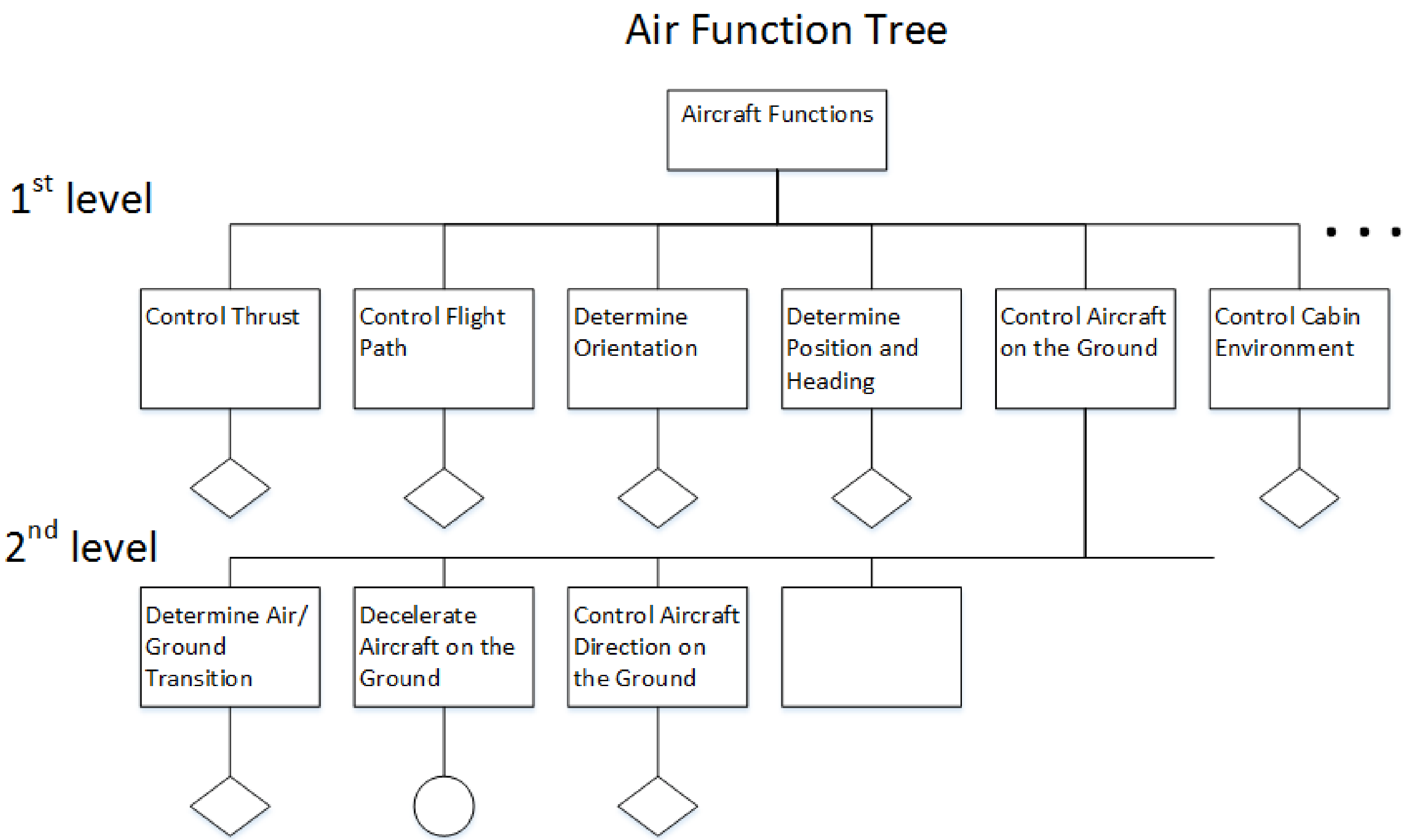




2. Analyze/examine pieces separately and combine results

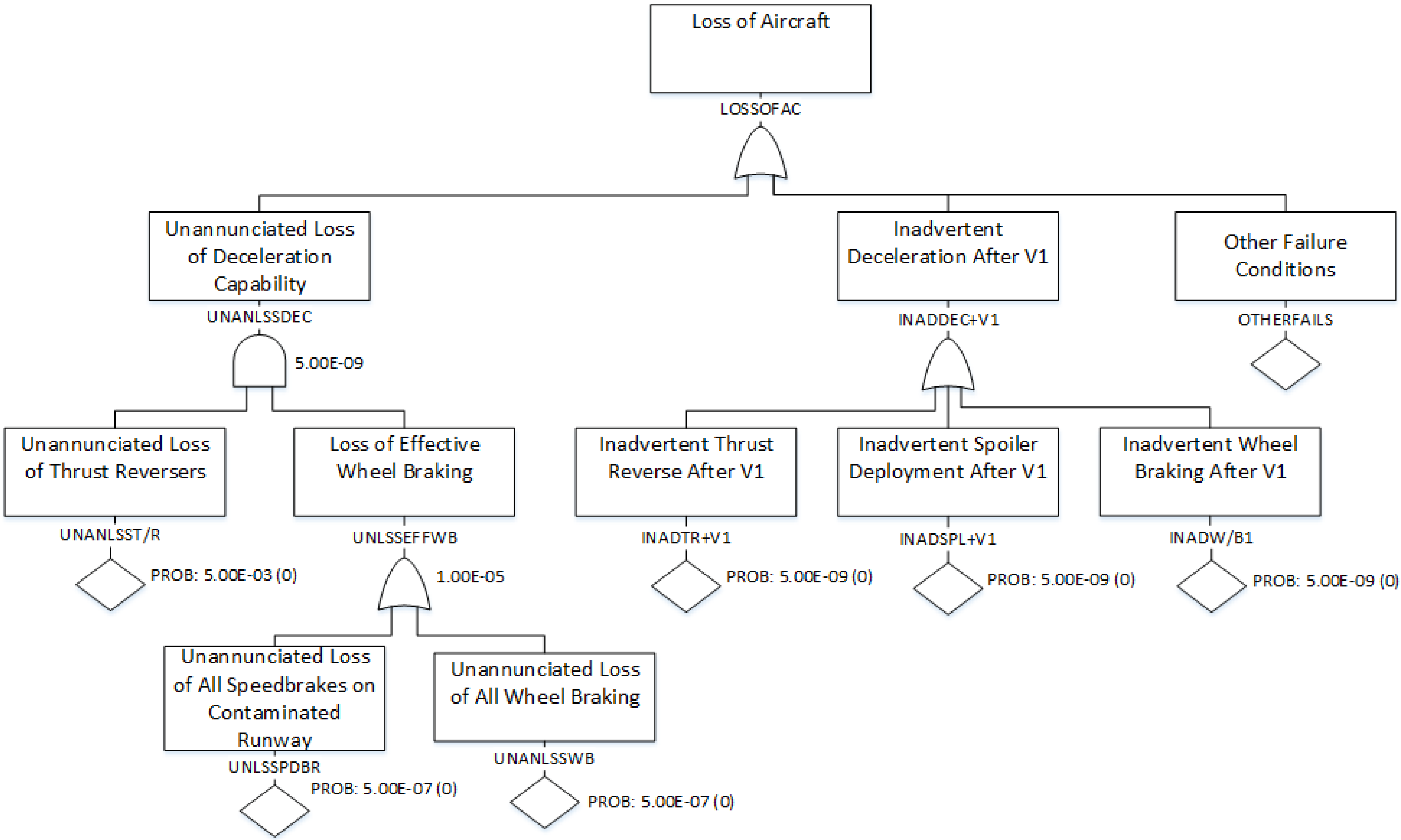
- Assumes such separation does not distort phenomenon
  - ✓ Each component or subsystem operates independently
  - ✓ Components act the same when examined singly as when playing their part in the whole
  - ✓ Components/events not subject to feedback loops and non-linear interactions
  - ✓ Interactions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

Typical Decomposition Approach (SAE ARP 4761)



First, decompose top-down into components

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



Limitations of Probabilistic Risk Assessment

- Failures of components must be independent
- Doesn’t work for non-failure accidents (caused by system design errors and not component failures)
- Doesn’t work for software or new technology or new designs
- Doesn’t work for human errors in complex systems
- Unreliable results
  - Two scientific evaluations (1980s and 2002)
  - Both showed variance in results of 3-4 orders of magnitude
- Empirical results are terrible: All accidents I have seen had a PRA that showed they could not happen!

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

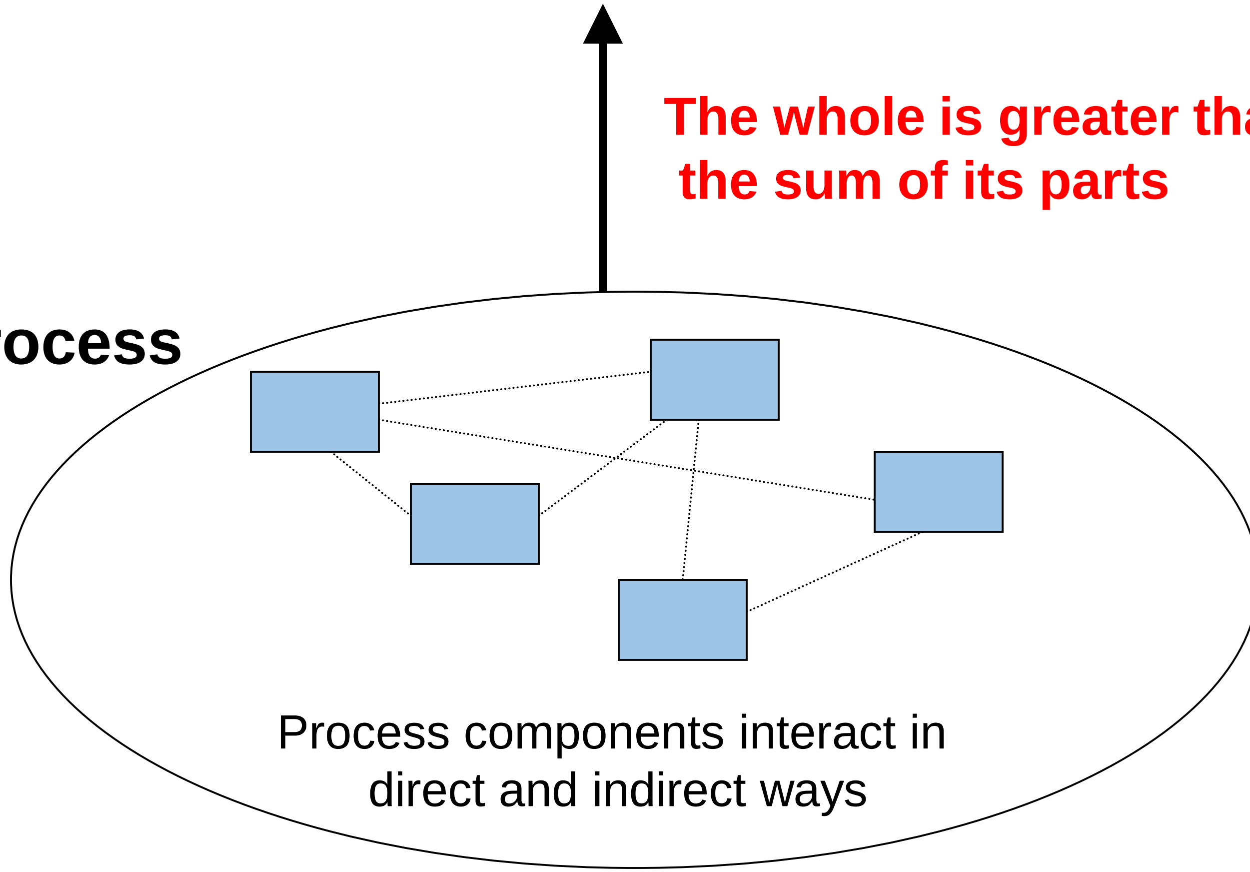
System Theory

Emergent properties

(arise from complex interactions)

The whole is greater than the sum of its parts

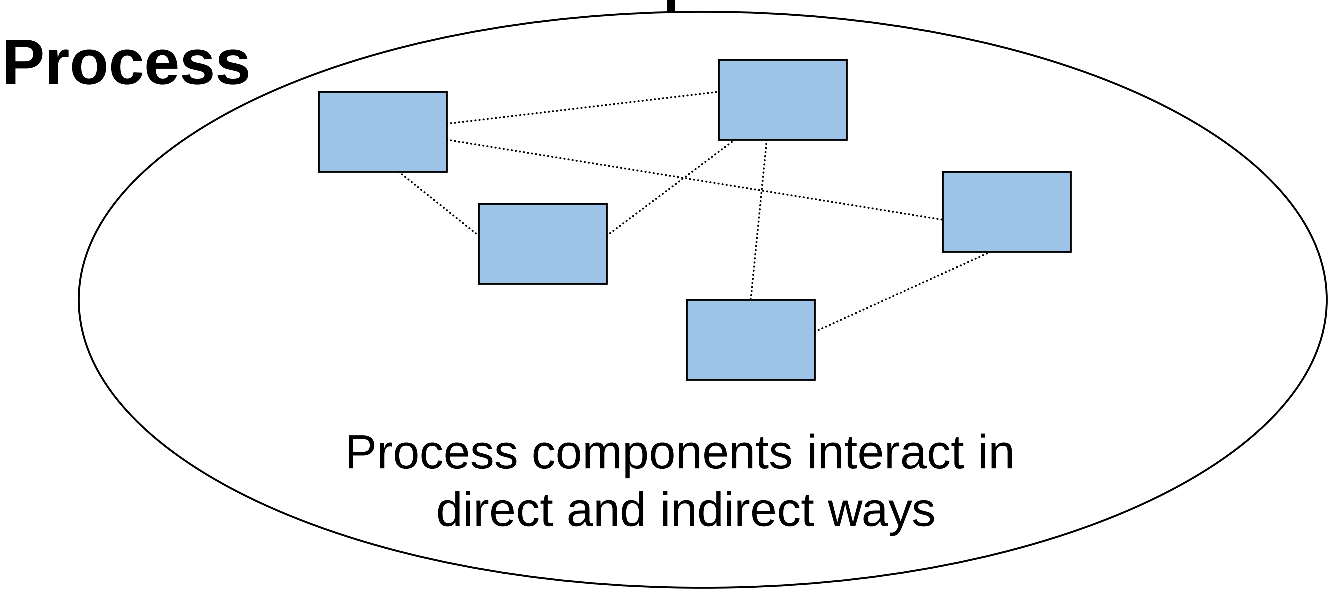
Process



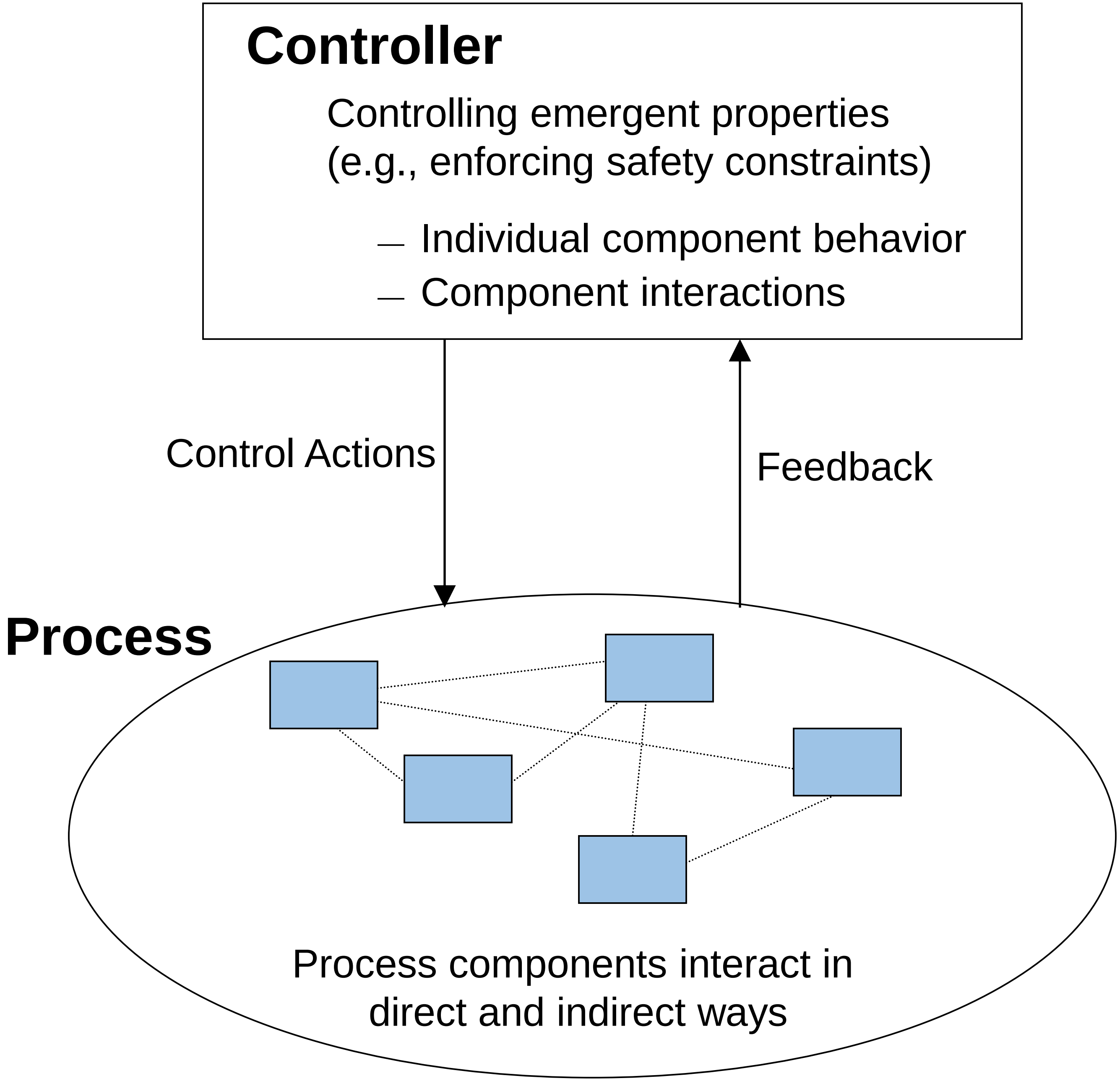
Emergent properties

(arise from complex interactions)

The whole is greater than the sum of its parts



Safety and security are emergent properties


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

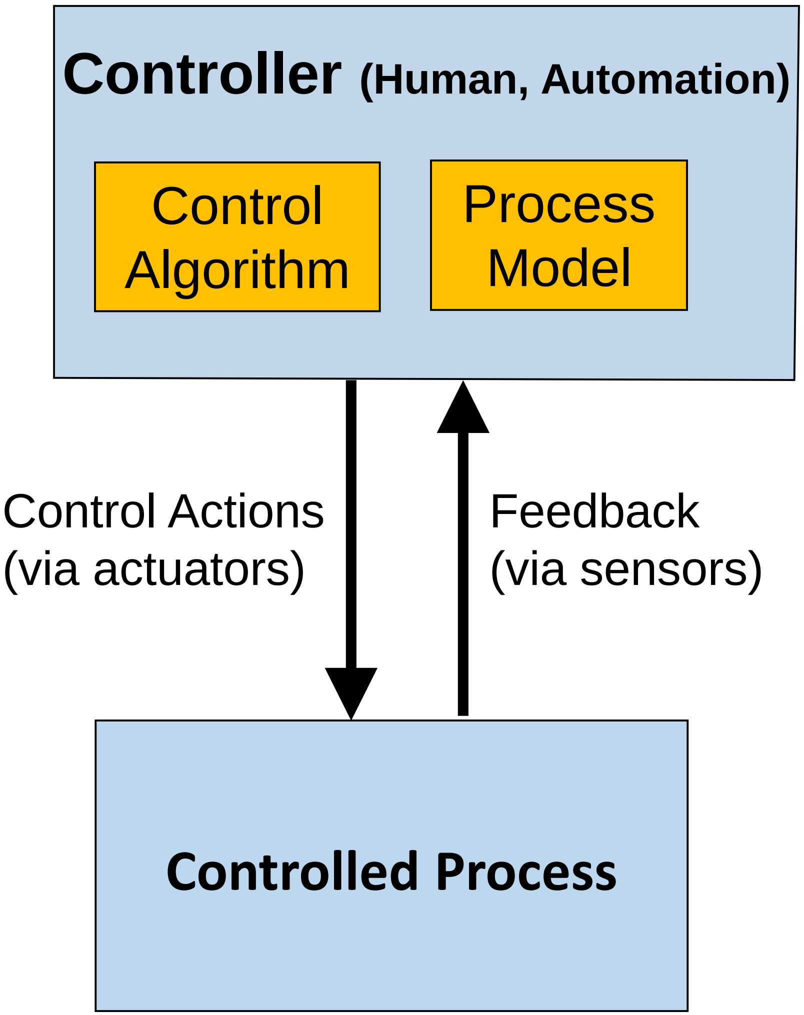
- Aircraft must maintain sufficient lift to remain airborne
- Vehicles must maintain minimum separation
- Public health system must prevent exposure of public to contaminated water, food products, and viruses
- Pressure in a offshore well must be controlled
- Integrity of hull must be maintained on a 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

STAMP (System-Theoretic Accident Model and Processes)

- A new, more powerful accident/loss causality model
- Based on systems theory, not reliability theory
- Treats accidents/losses as a dynamic control problem (vs. a failure problem)
- Applies to <u>very</u> complex systems
- Includes
  - Scenarios from traditional hazard analysis methods (failure events)
  - Component interaction accidents
  - Software and system design errors
  - Human errors
  - Entire socio-technical system (not just technical part)

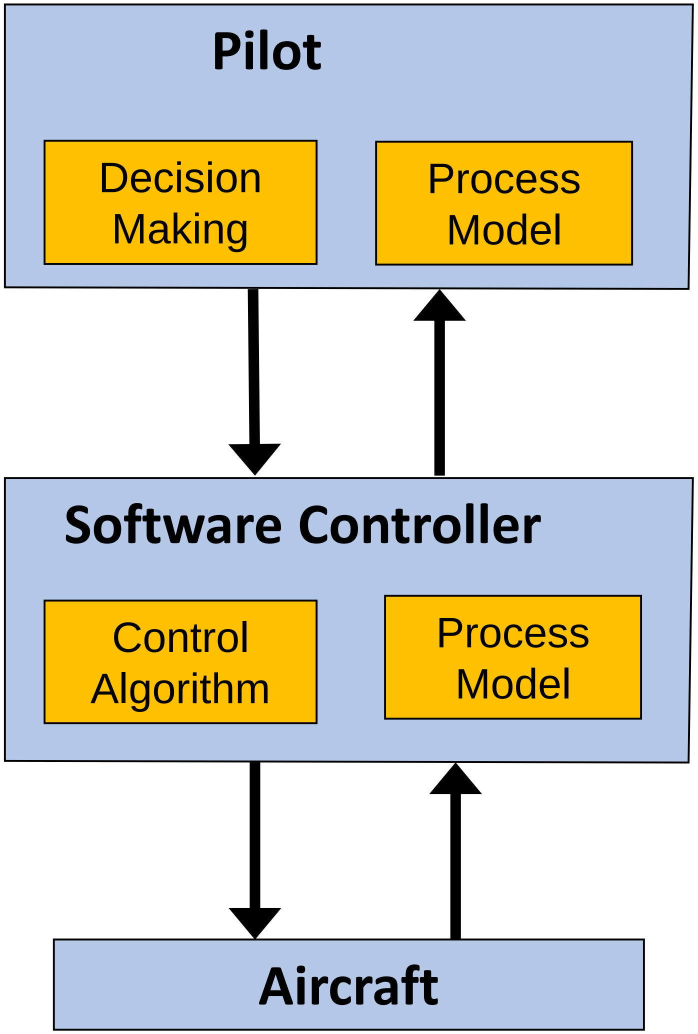
Treating Safety as a Control Problem



- Controllers use a **<u>process model</u>** 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 …

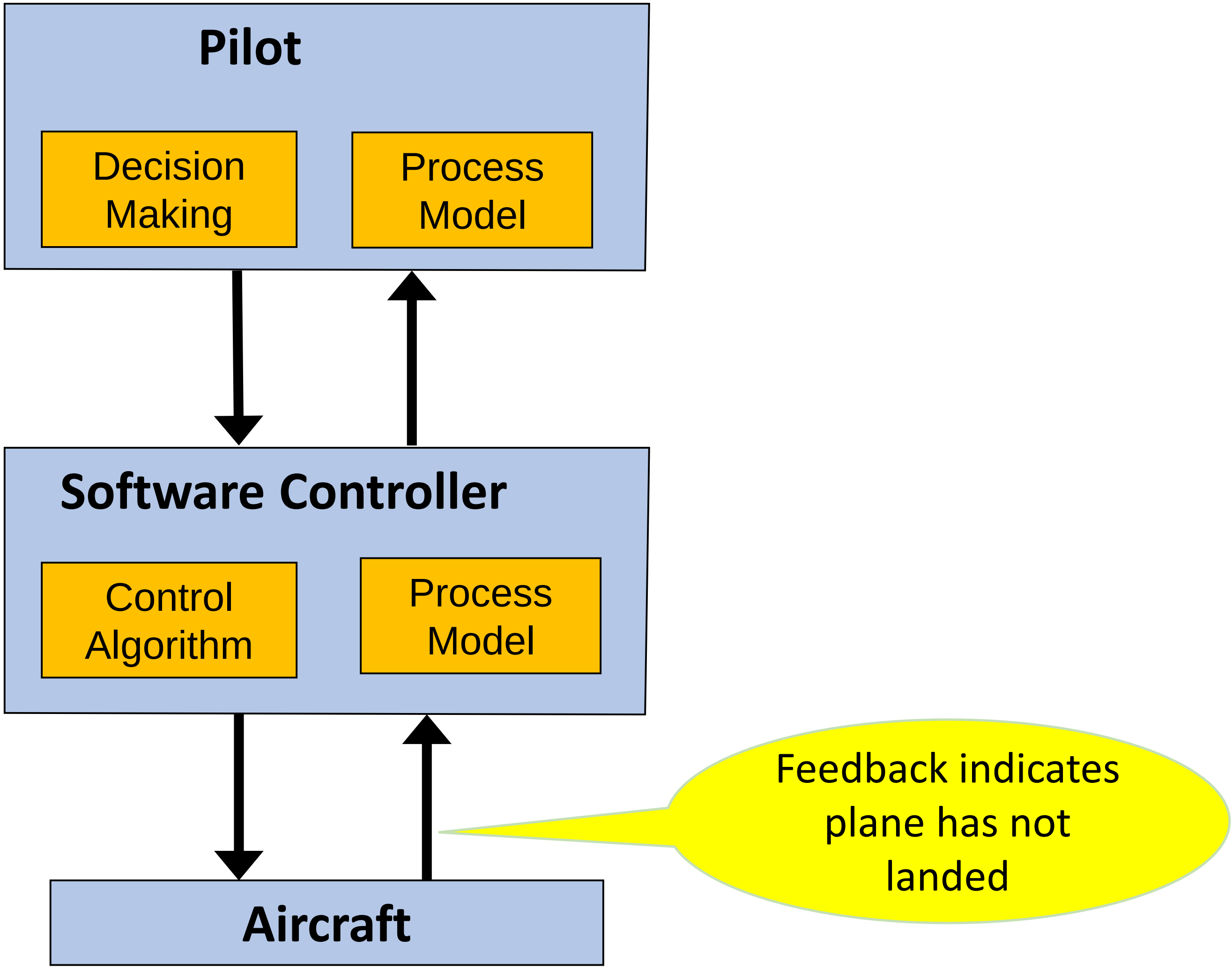
Warsaw (Reverse Thrusters)

<u>Hazard</u>: Inadequate aircraft deceleration after landing



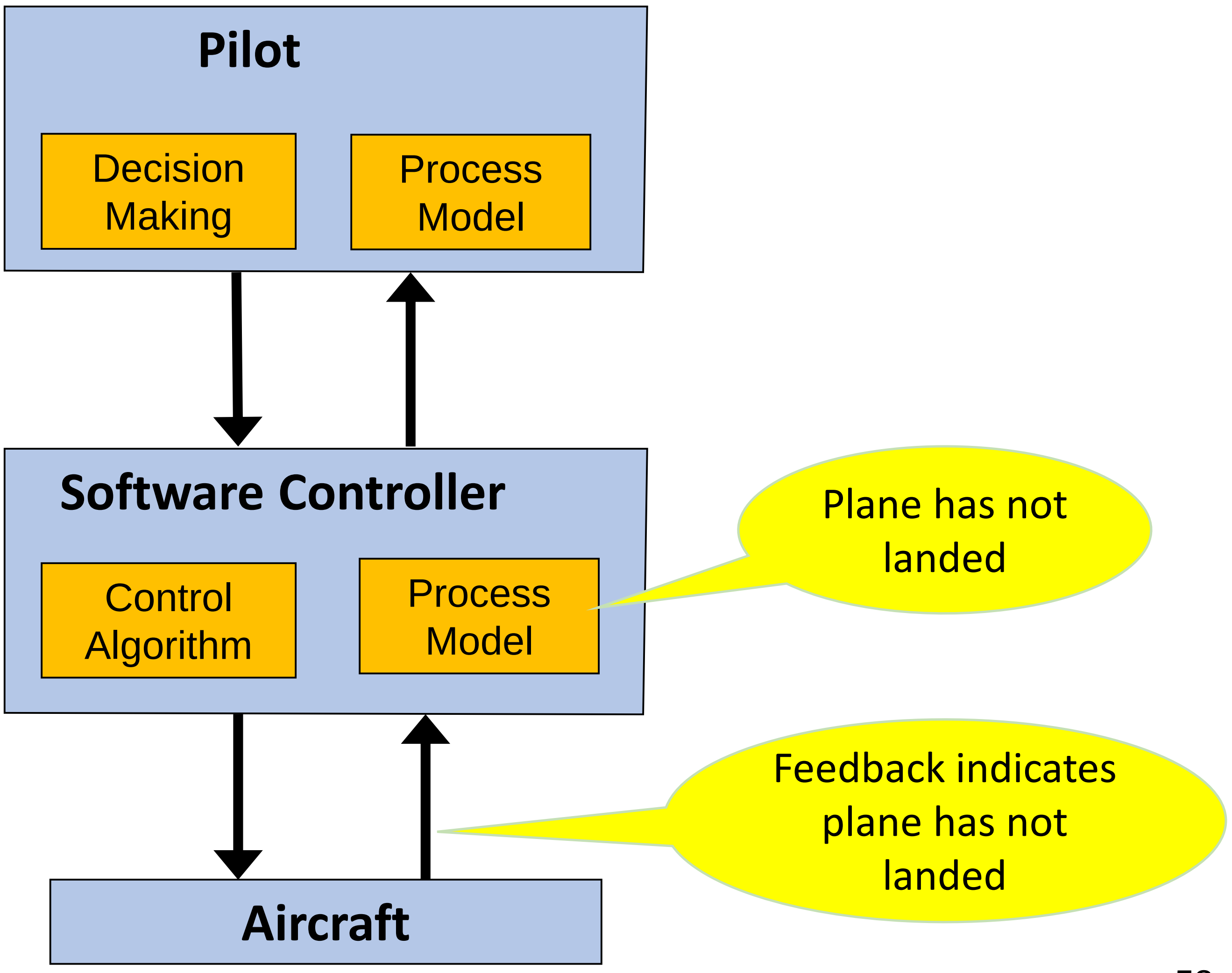
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



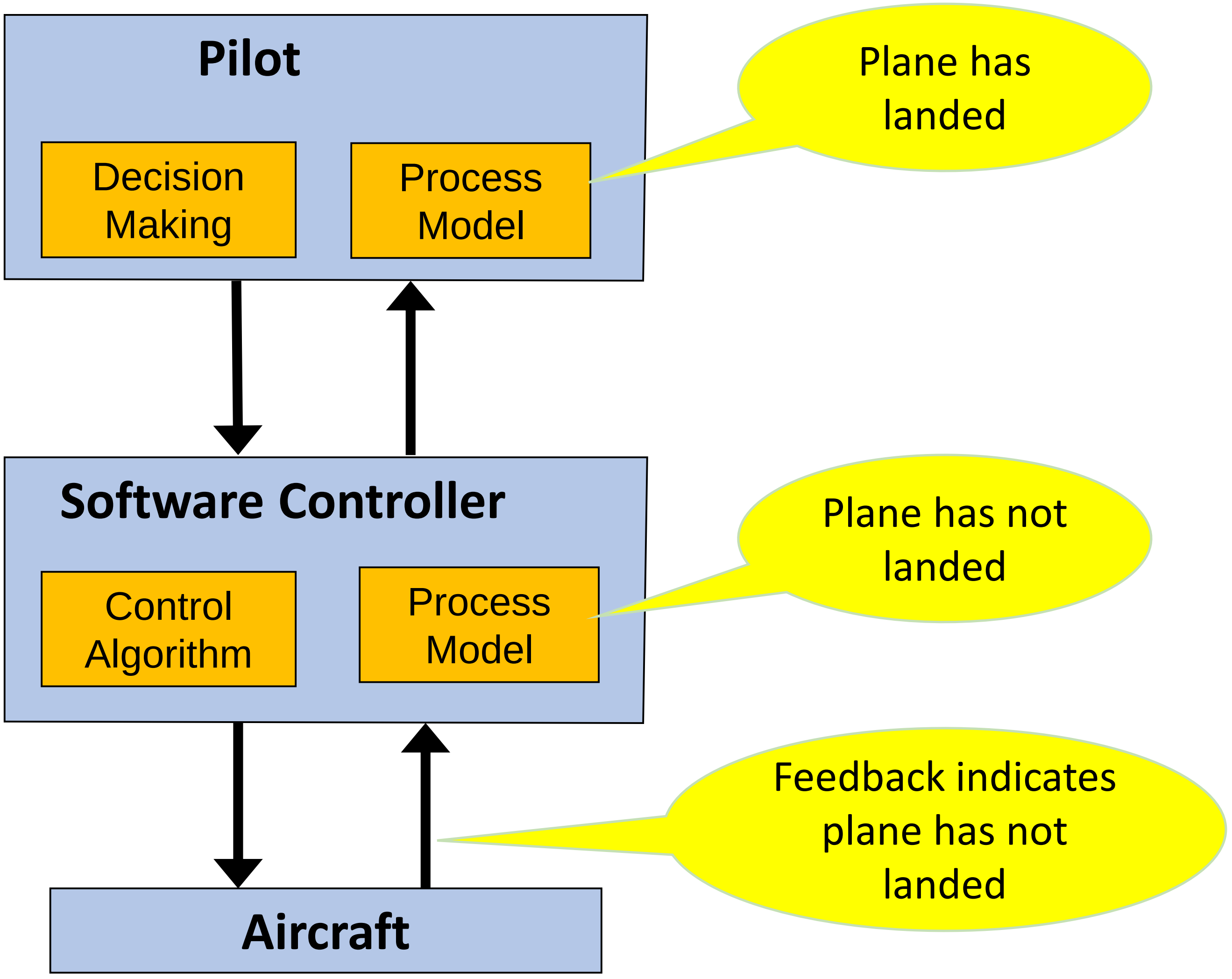
Warsaw (Reverse Thrusters)

<u>Hazard</u>: Inadequate aircraft deceleration after landing



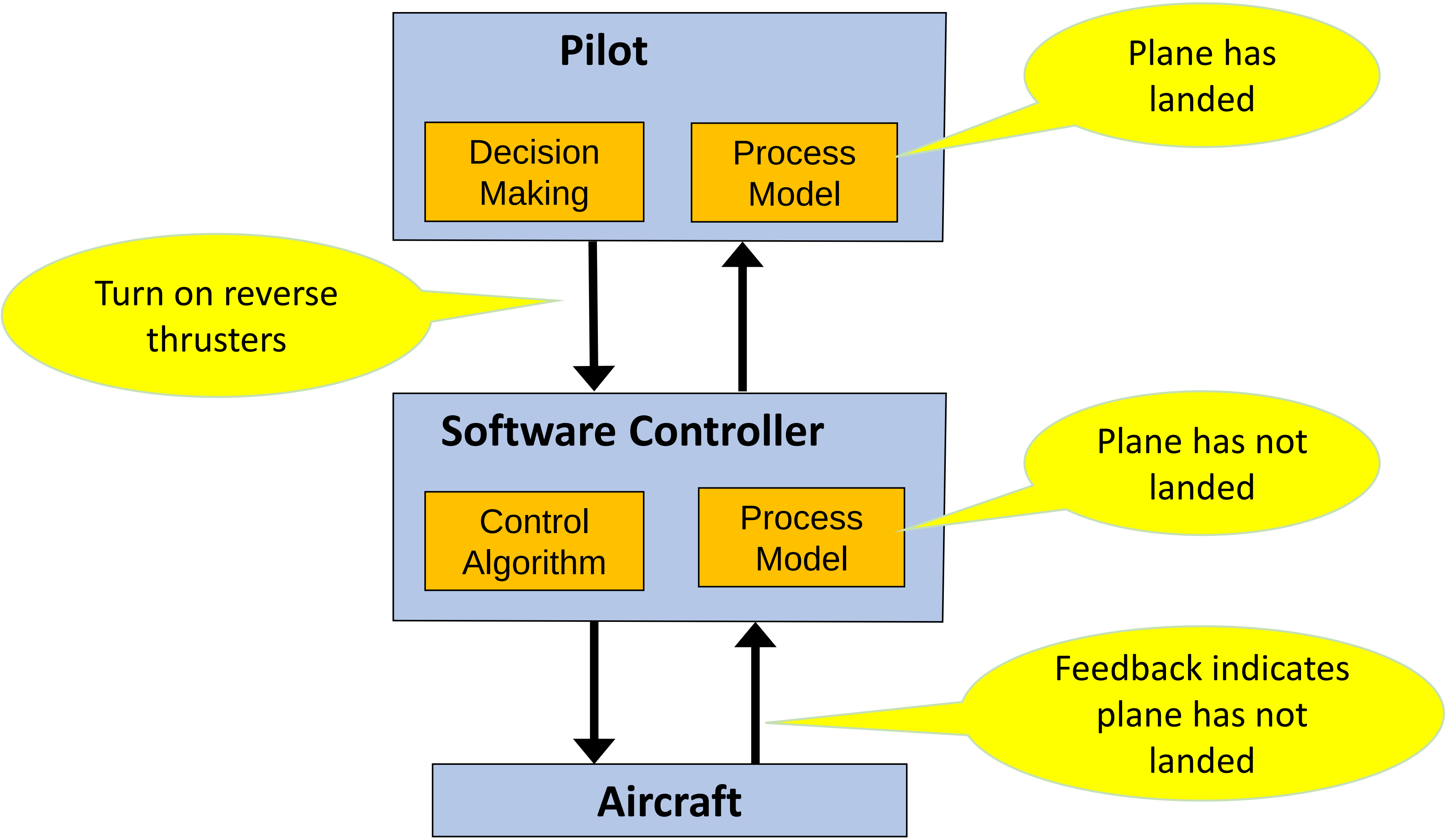
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



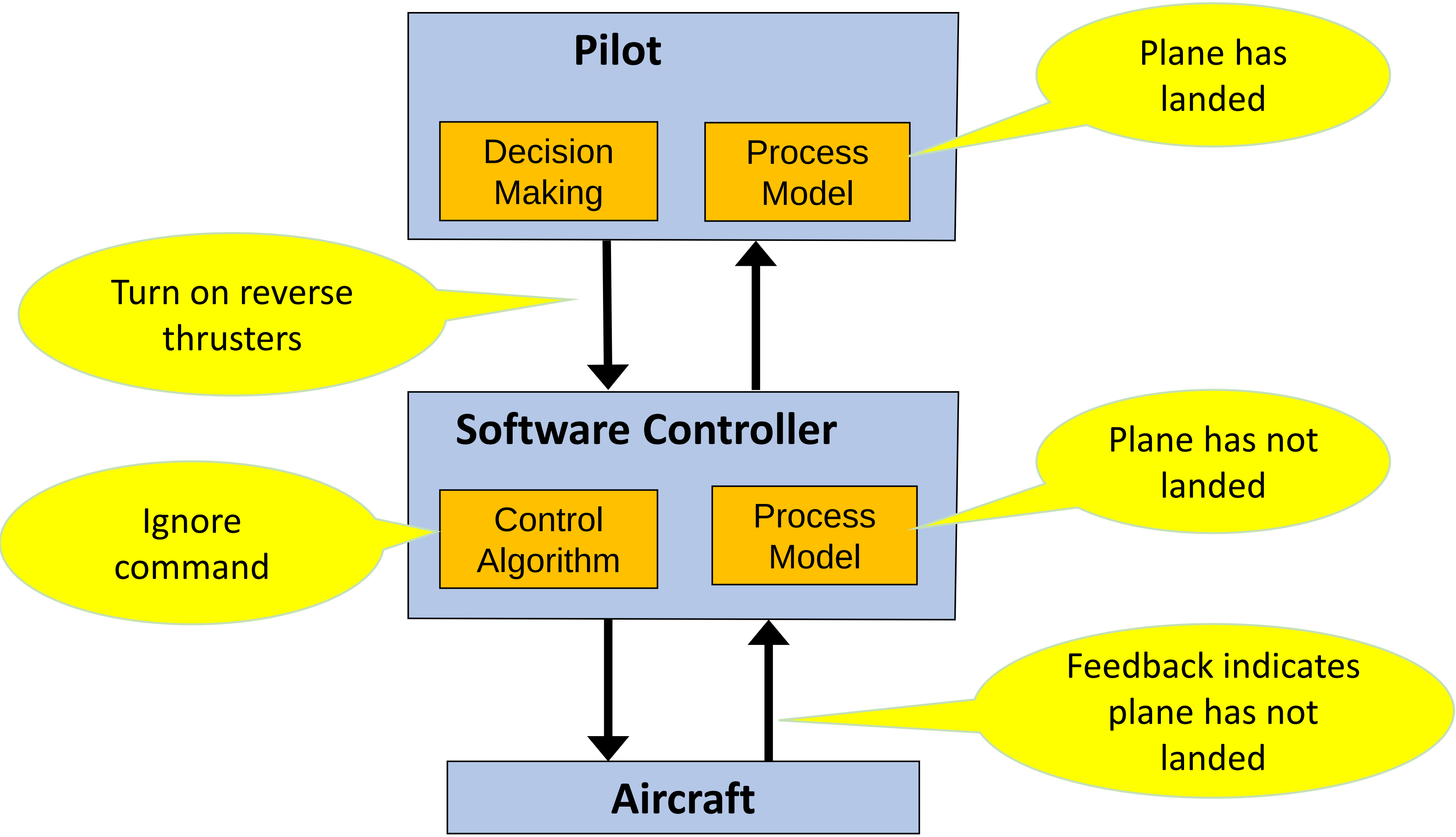
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



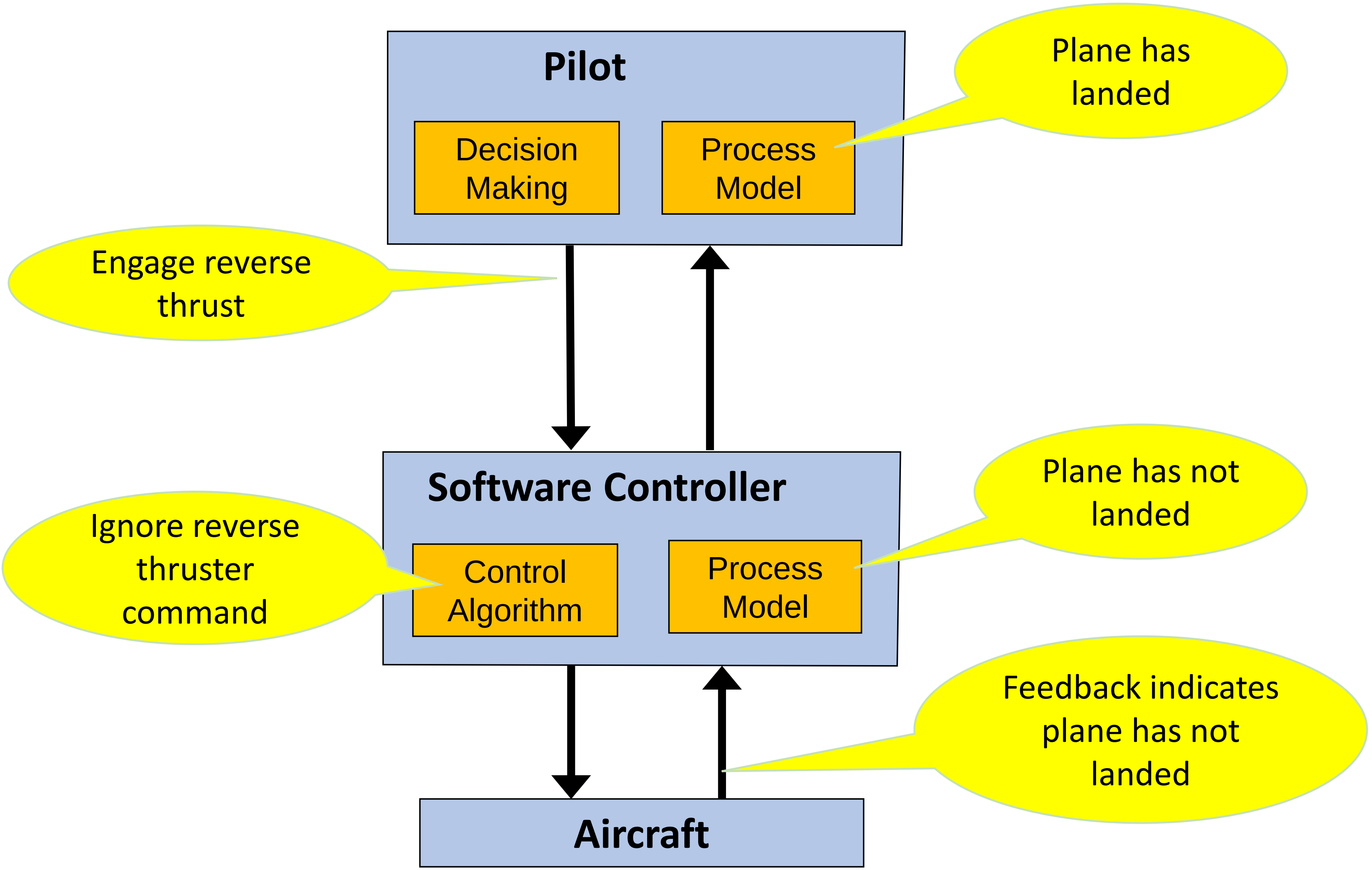
Warsaw (Reverse Thrusters)

Hazard: Inadequate aircraft deceleration after landing



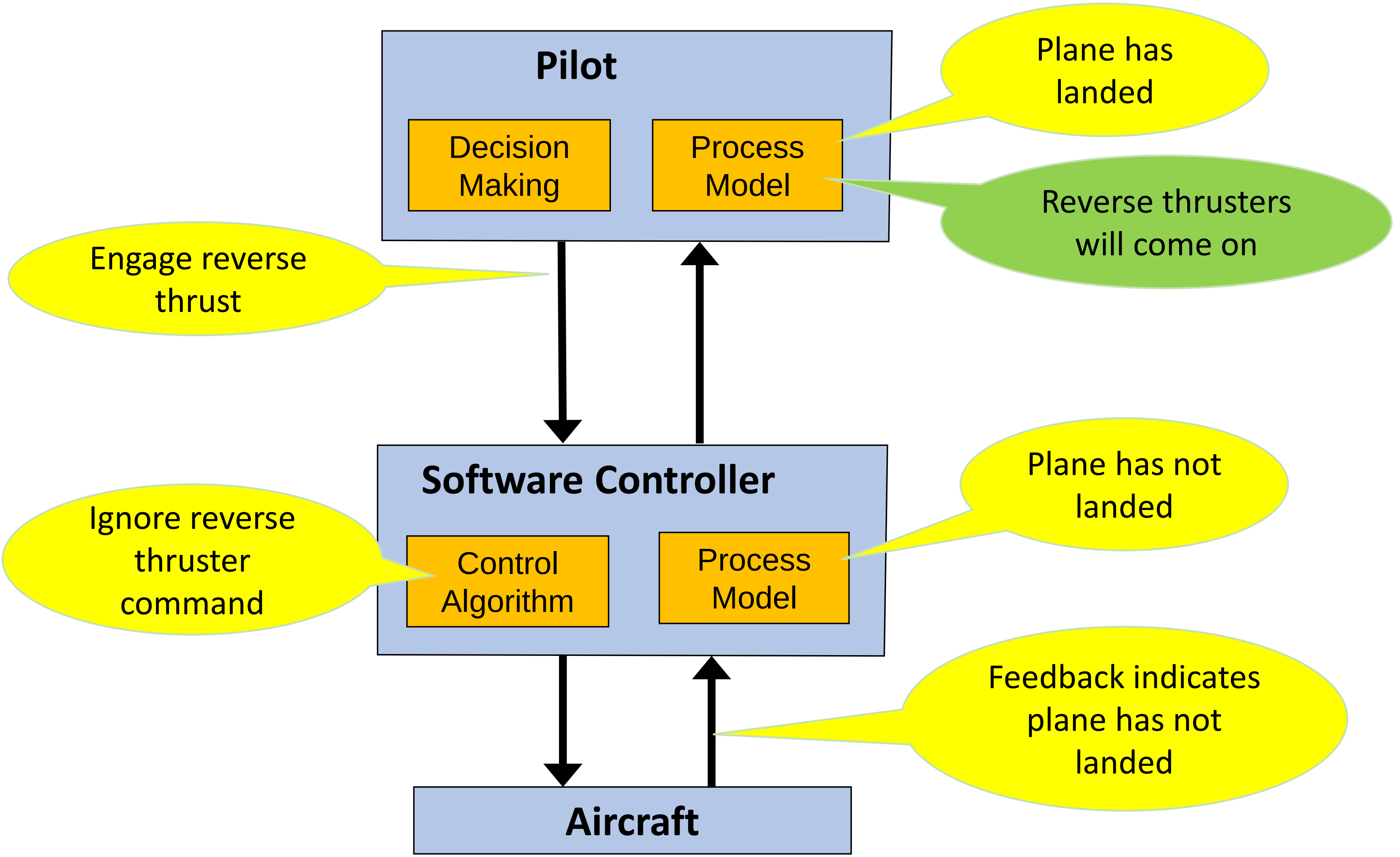
Moscow (Reverse Thrusters)

Hazard: Inadequate Deceleration after Landing



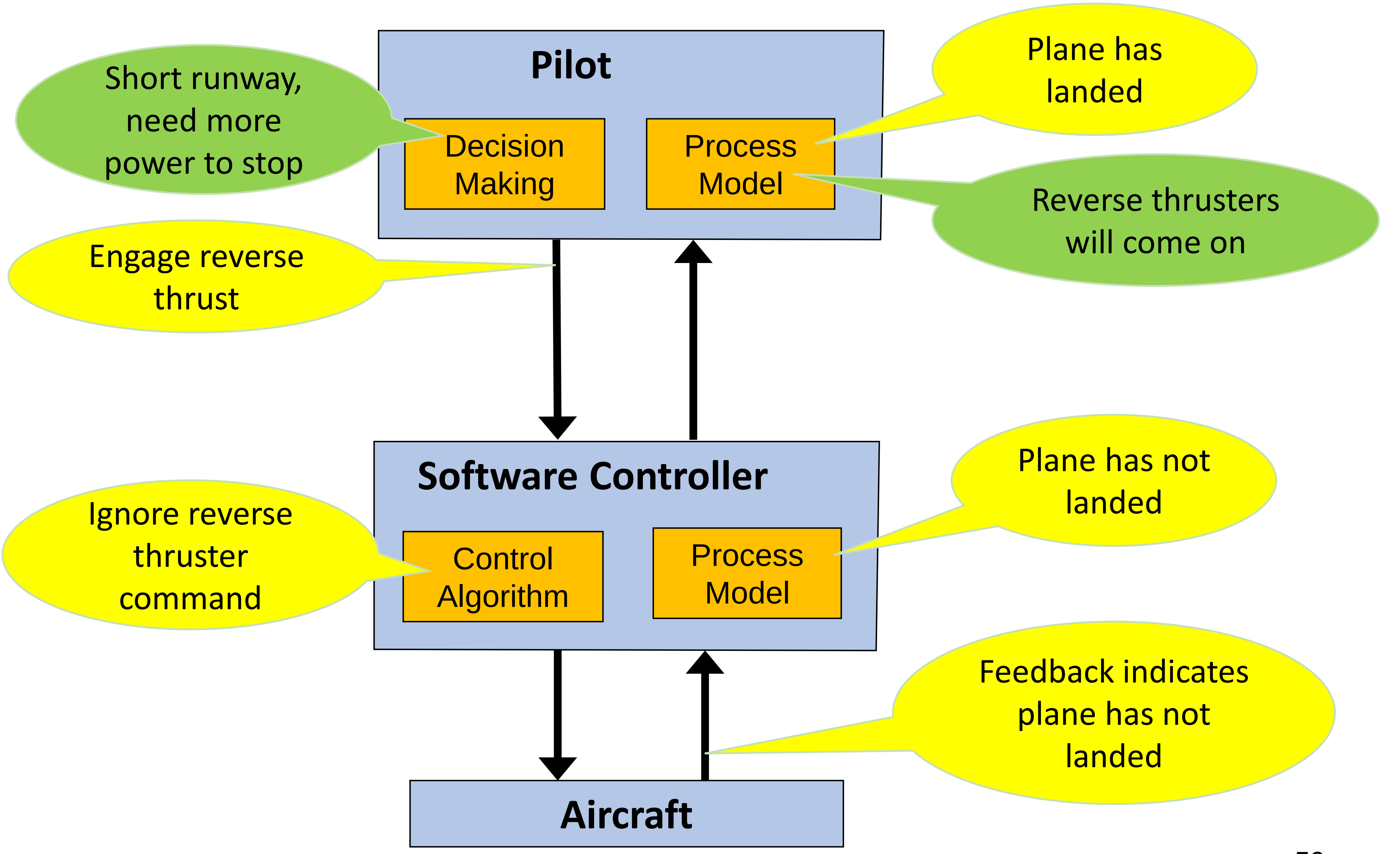
Moscow (Reverse Thrusters)

Hazard: Inadequate Deceleration after Landing



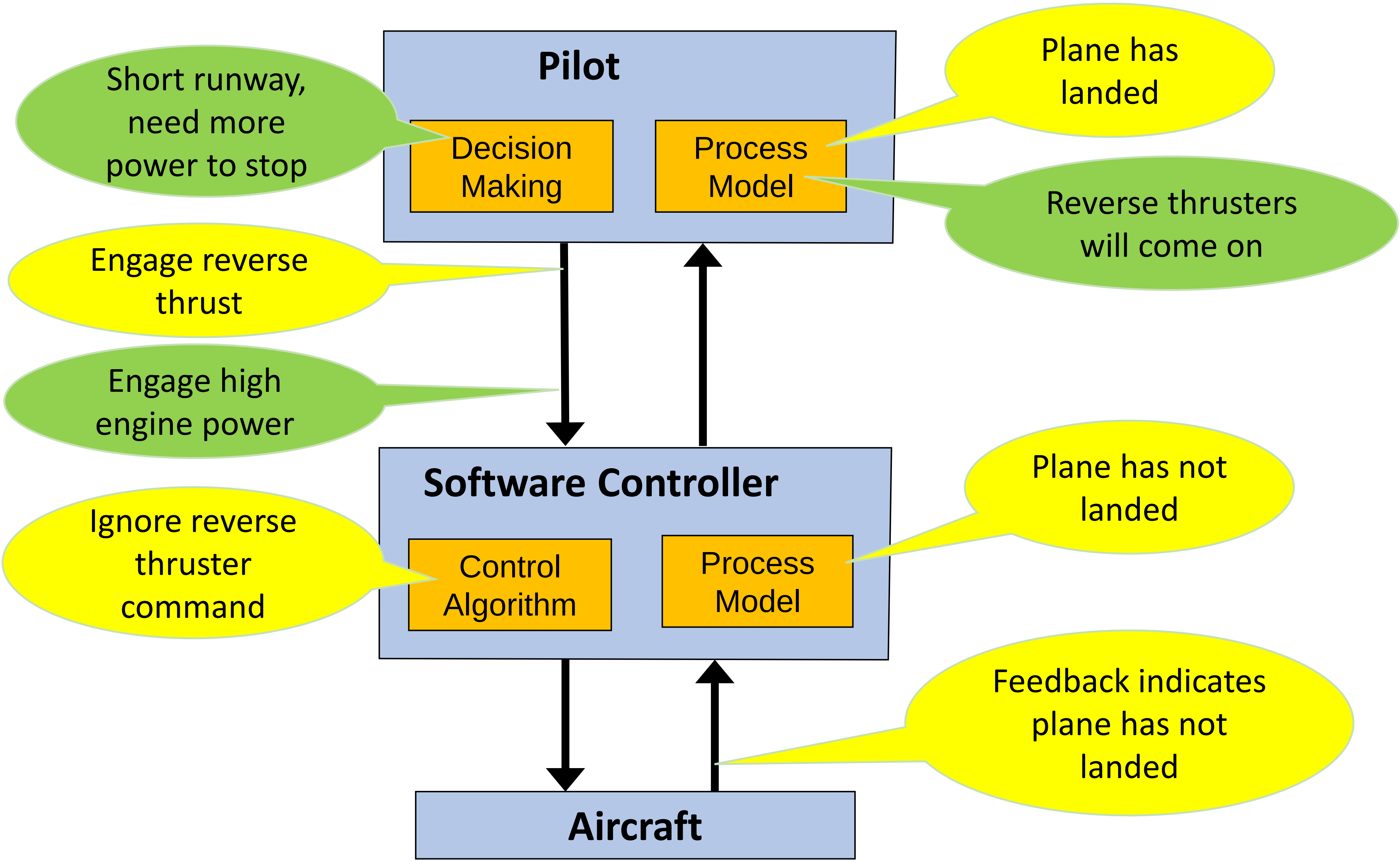
Moscow (Reverse Thrusters)

Hazard: Inadequate Deceleration after Landing



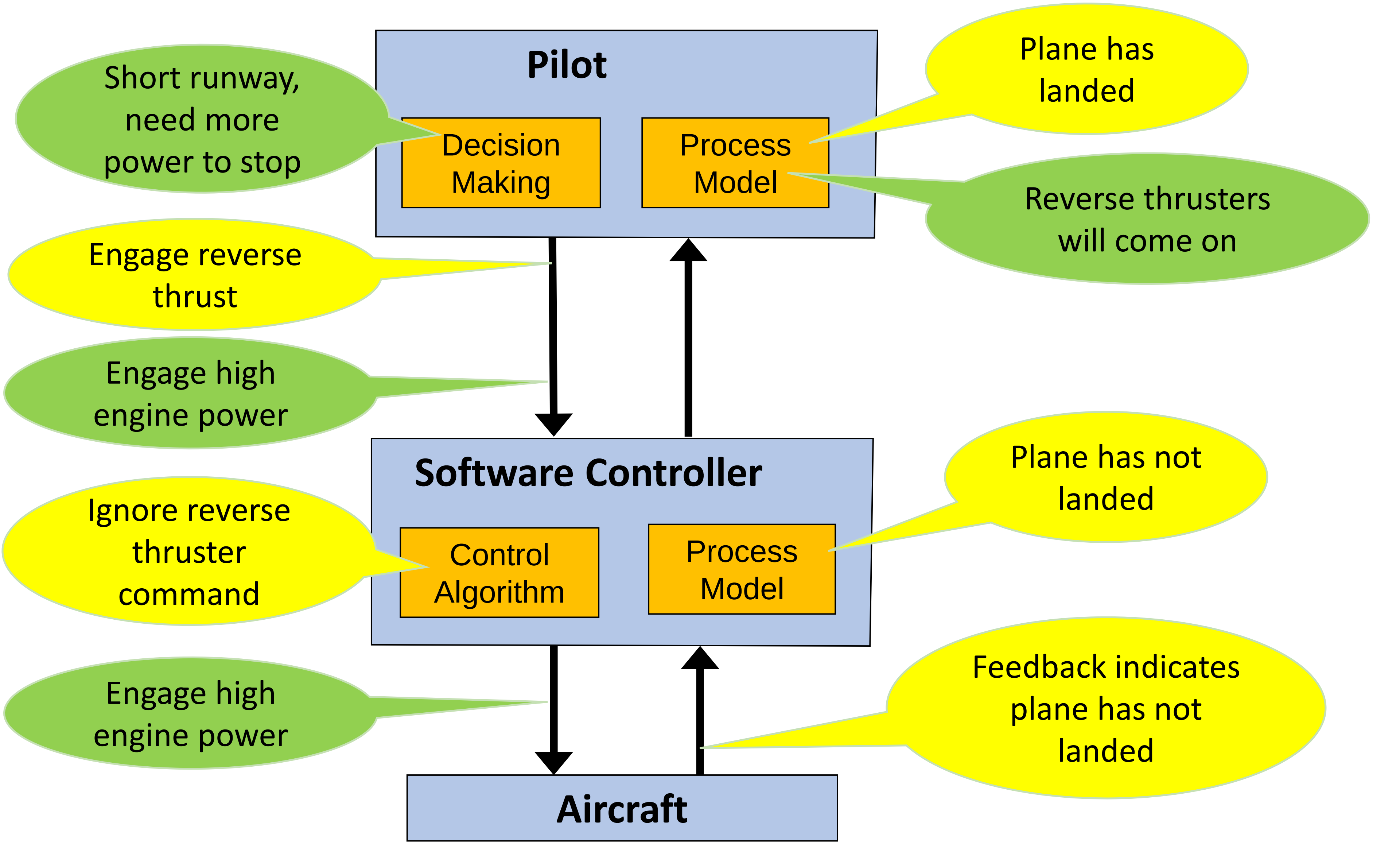
Moscow (Reverse Thrusters)

Hazard: Inadequate Deceleration after Landing



Moscow (Reverse Thrusters)

Hazard: Inadequate Deceleration after Landing



Is The Approach Practical?

- Has been or is being used in a large variety of industries
  - Automobiles (>80% use)
  - Aircraft and Spacecraft (extensive use and growing)
  - Defense systems (UAVs, AF GBSD, Army FVL, etc.)
  - Ships/Marine
  - Air Traffic Control
  - Medical Devices and Hospital Safety
  - Chemical plants
  - Oil and Gas
  - Nuclear and Electric Power
  - Robotic Manufacturing / Workplace Safety
- 2,316 registrants (87 countries) for STAMP Workshop this year
- New international standards (autos, aircraft, defense) created or in development.

Does it Work?

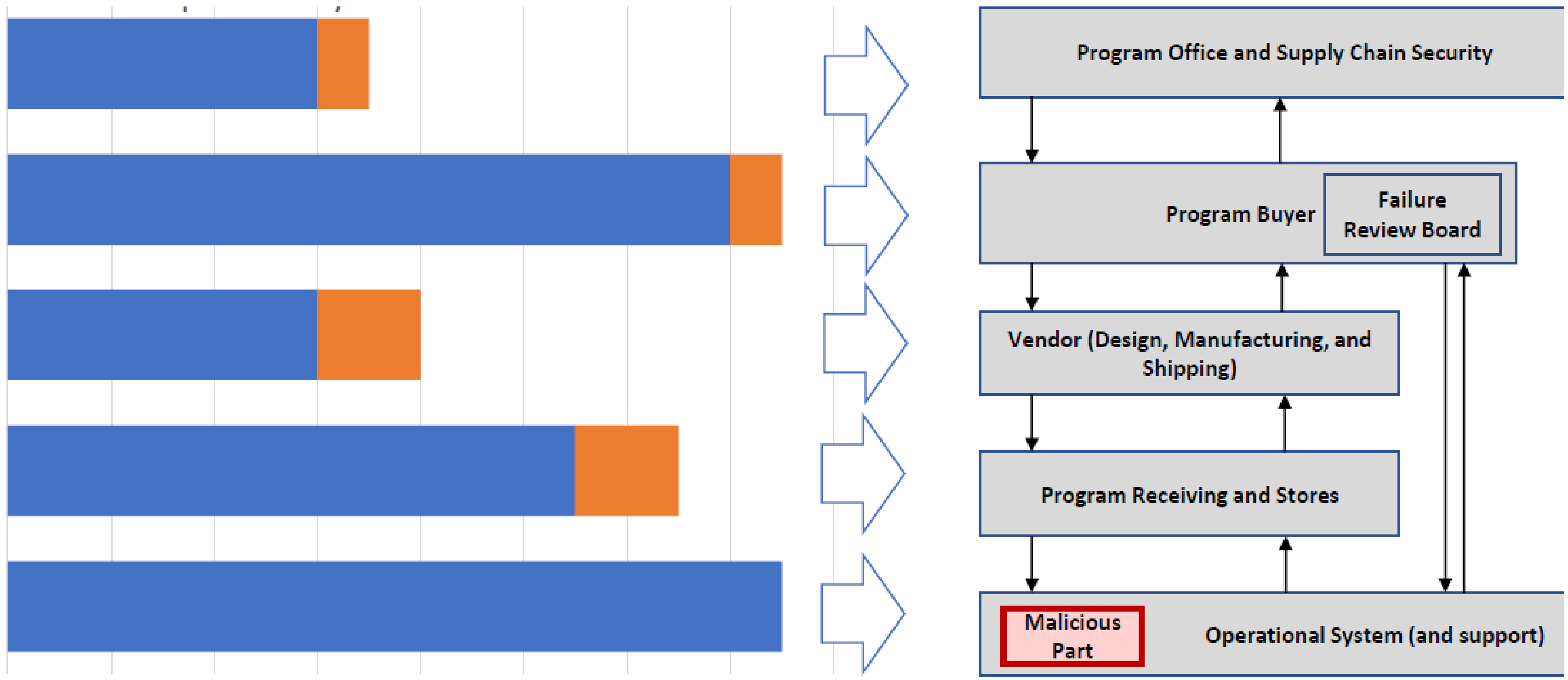
- Hundreds of evaluations and comparison with traditional approaches used now
  - Controlled scientific and empirical (in industry)
  - All show STPA is better (identifies more critical requirements or design flaws)
  - All (that measured) show STPA requires orders of magnitude fewer resources than traditional techniques

Example: STPA applied to one DoD program before SolarWinds attack

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

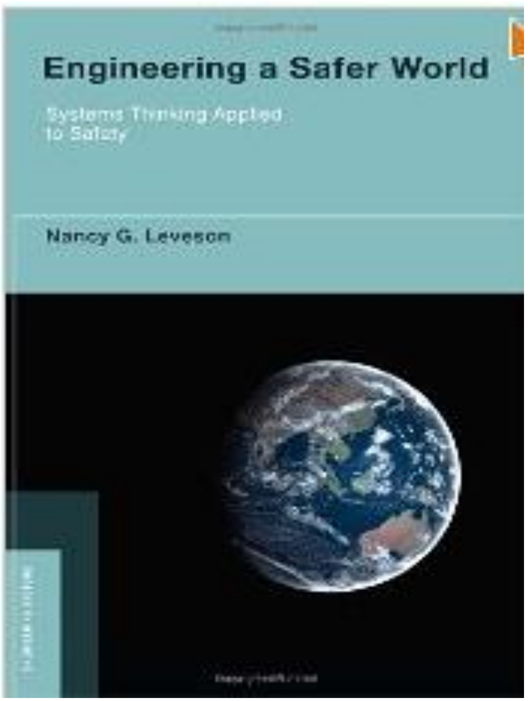
- Program that used STPA was protected from SolarWinds
- Vulnerabilities found by STPA

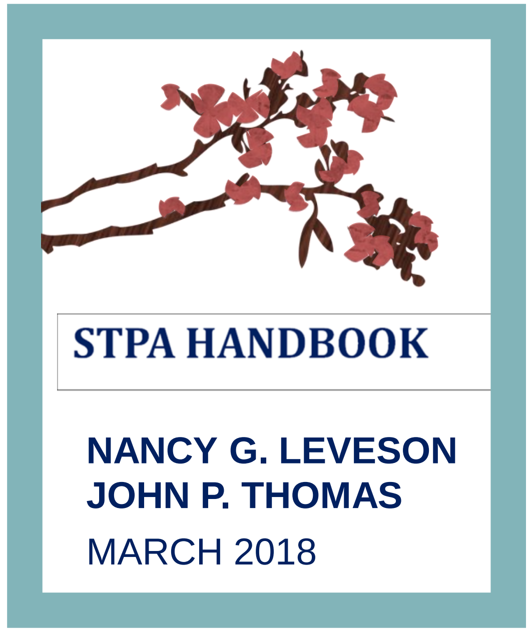
Later exploited by SolarWinds attackers
Not exploited by SolarWinds attackers



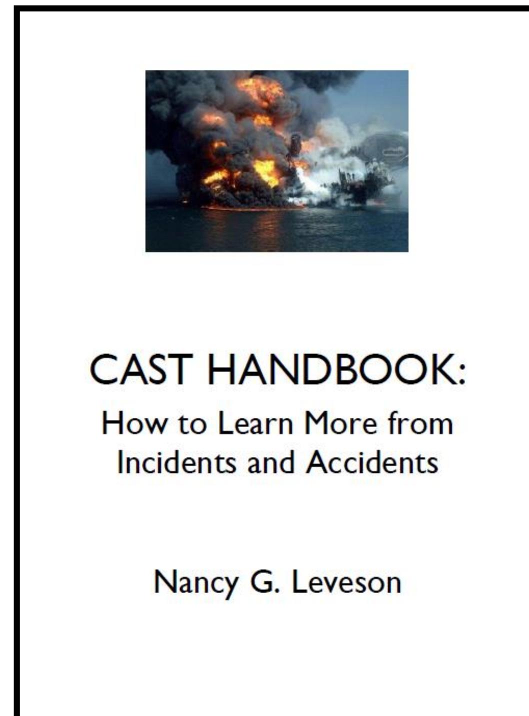
More Information

- http://psas.scripts.mit.edu (papers, presentations from conferences, tutorial slides, examples, etc.)





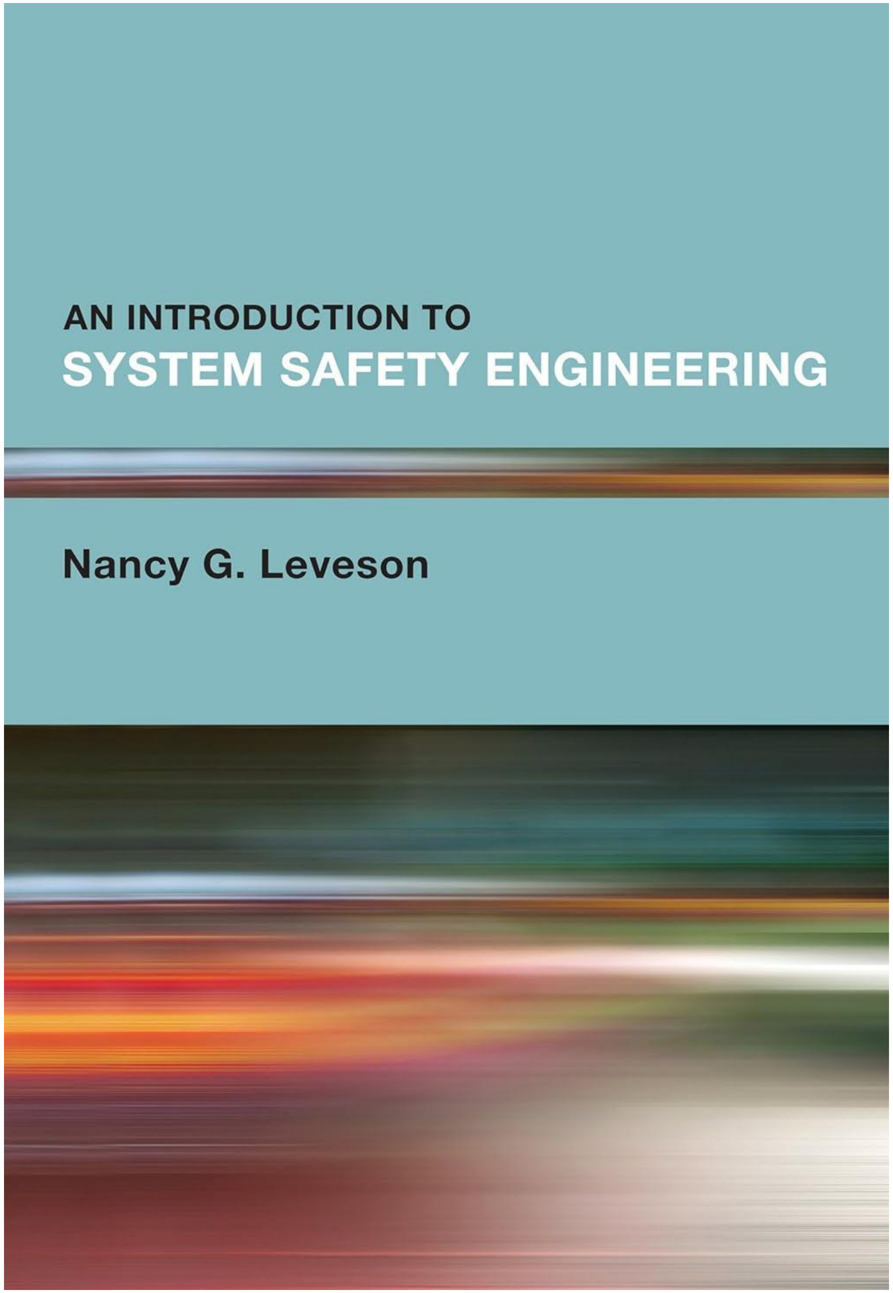
Free download:
http://psas.scripts.mit.edu
(**300,000+** downloads since 2018.
Japanese, Chinese, and
Korean versions)



Free download:
http://mitpress.mit.edu/books/engineering-safer-world
In Japanese available 2024

Free download:
http://sunnyday.mit.edu/CAST-Handbook.pdf
(Korean, Japanese versions)

New Textbook



Conclusions (1)

- Complexity is reaching a new level (tipping point)
  - Old safety approaches becoming less effective
  - New causes of losses appearing (especially related to use of software and autonomy)

- Traditional analysis approaches do not provide the information necessary to prevent losses in these systems

- Need a paradigm change to a "systems approach"

  Change focus

  ~~Increase component reliability (prevent failures)~~

  ↓

  Enforce safe system behavior (constraints on system behavior)

Conclusions (2)

- Allows creating new analysis and engineering approaches
  - More powerful and inclusive
  - Orders of magnitude less expensive
  - Work on extremely complex systems (top-down system engineering)
  - Help to design safety, security, and other properties in from the beginning

- New paradigm works much better than old techniques:
  - Empirical evaluations and controlled studies show it finds more causal scenarios (the “unknown unknowns”)
  - Can be used before a detailed design exists to design safe and secure systems from the beginning