Engineering a Safer and More Secure World

Nancy Leveson

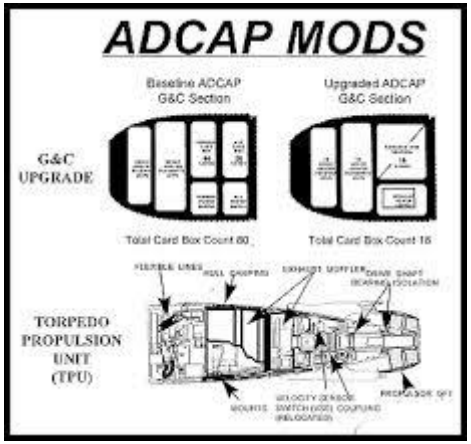
MIT

- You've carefully thought out all the angles
- You've done it a thousand times
- It comes naturally to you
- You know what you're doing, it's what you've been trained to do your whole life.
- Nothing could possibly go wrong, right?

© Copyright Nancy Leveson, June 2011

Think Again

© Copyright Nancy Leveson, June 2011



© Copyright Nancy Leveson, June 2011

Goal: Answer the Following Questions:

- Why do we need something new?
- What is STAMP and how does it differ from what people do now?
- What kinds of tools are available?
- How is it being used?
- Does it work?

© Copyright Nancy Leveson, June 2011

Why do we need something new?

© Copyright Nancy Leveson, June 2011

Our current tools are all 40-65 years old but our technology is very different today

1940 1950 1960 1970 1980 1990 2000 2010 2020

FMEA

FTA HAZOP

ETA

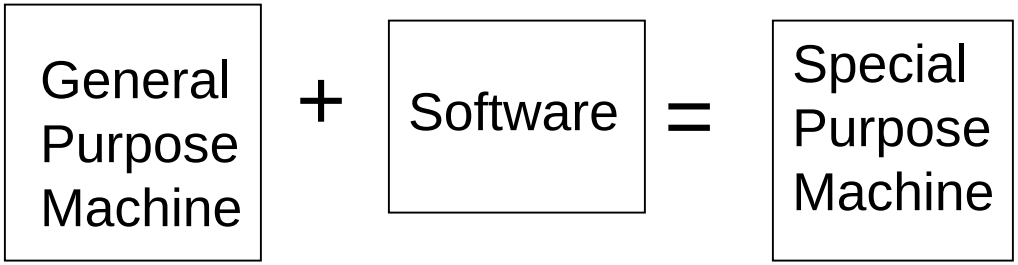
Bow Tie (CCA) FTA + ETA

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

Assumes accidents caused by component failures

© Copyright Nancy Leveson, June 2011

1. Software does not “fail”

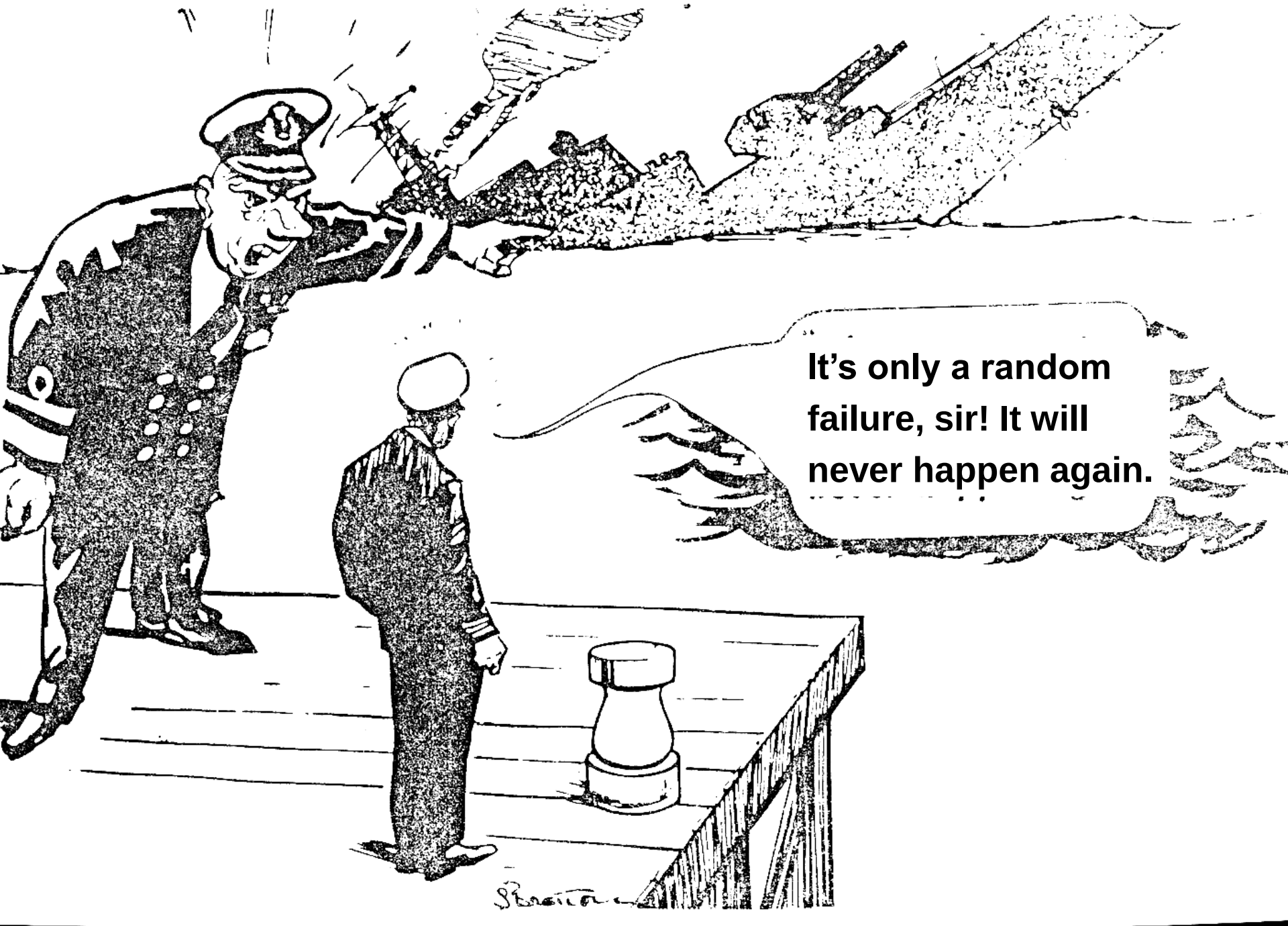


Software is simply the design of a machine abstracted from its physical realization

Advantages

- Machines that were physically impossible or impractical to build become feasible
- Design can be changed without retooling or manufacturing
- Can concentrate on steps to be achieved without worrying about how steps will be realized physically

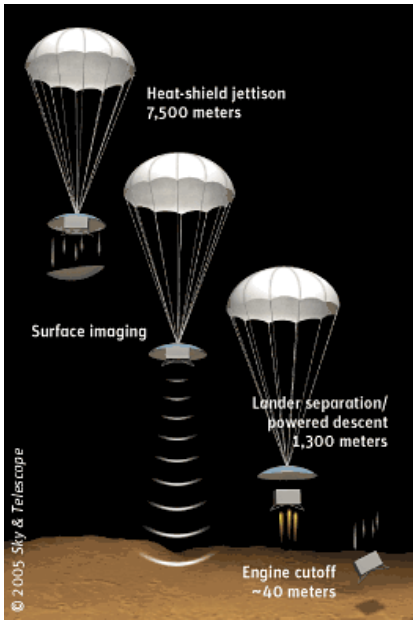
Software is pure design and designs do not “fail”


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

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.

Accident with No Component Failures



- Mars Polar Lander
  - Have to slow down spacecraft to land safely
  - Use Martian atmosphere, parachute, descent engines (controlled by software)
  - Software knows landed because of sensitive sensors on landing legs. Cut off engines when determine have landed.
  - But “noise” (false signals) by sensors generated when parachute opens. Not in software requirements.
  - Software not supposed to be operating at that time but software engineers decided to start early to even out the load on processor
  - Software thought spacecraft had landed and shut down descent engines while still 40 meters above surface

Two Types of Accidents

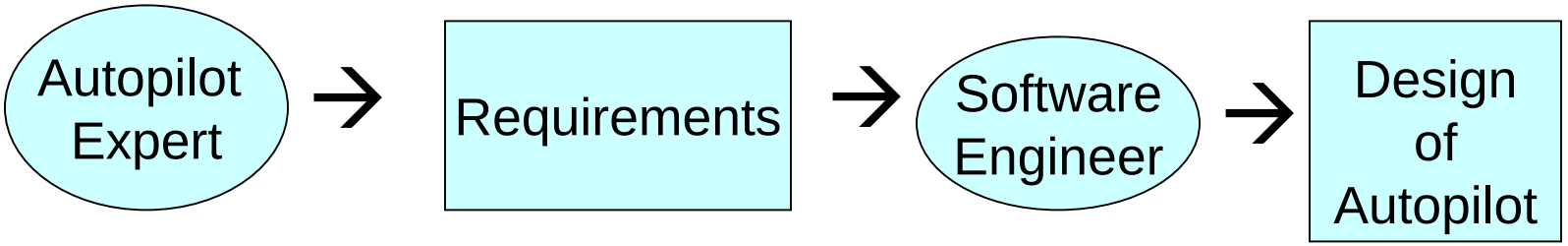
- **Component Failure Accidents**
  - Single or multiple component failures
  - Usually assume random failure
- **Component Interaction Accidents**
  - Arise in interactions among components
  - Related to complexity (coupling) in our system designs, which leads to system design and system engineering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

Warsaw A320 Accident

- Software protected 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 runway into a small hill.

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

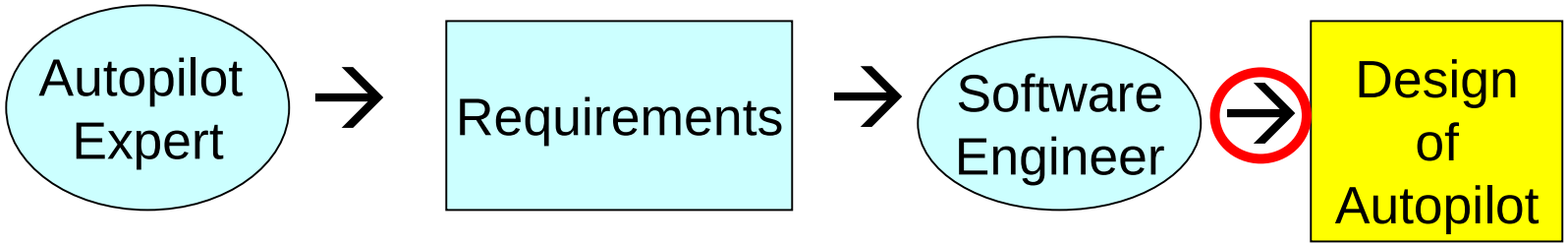
- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions



© Copyright Nancy Leveson, June 2011

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

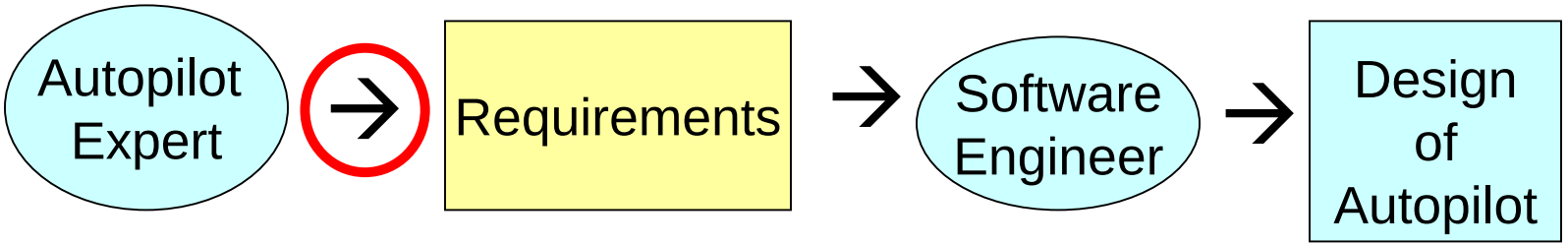
- Incomplete or wrong assumptions about operation of controlled system or required operation of computer
- Unhandled controlled-system states and environmental conditions



© Copyright Nancy Leveson, June 2011

2. 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

Boeing 787 Lithium Battery Fires

Models predicted 787 battery thermal problems would occur once in 10 million flight hours (10^7 flight hours using 4761 certification paradigm).

But two batteries overheated in just two weeks (52,000 flight hours or 2.6×10^4 flight hours) [NTSB 2013]

© Copyright John Thomas 2016

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

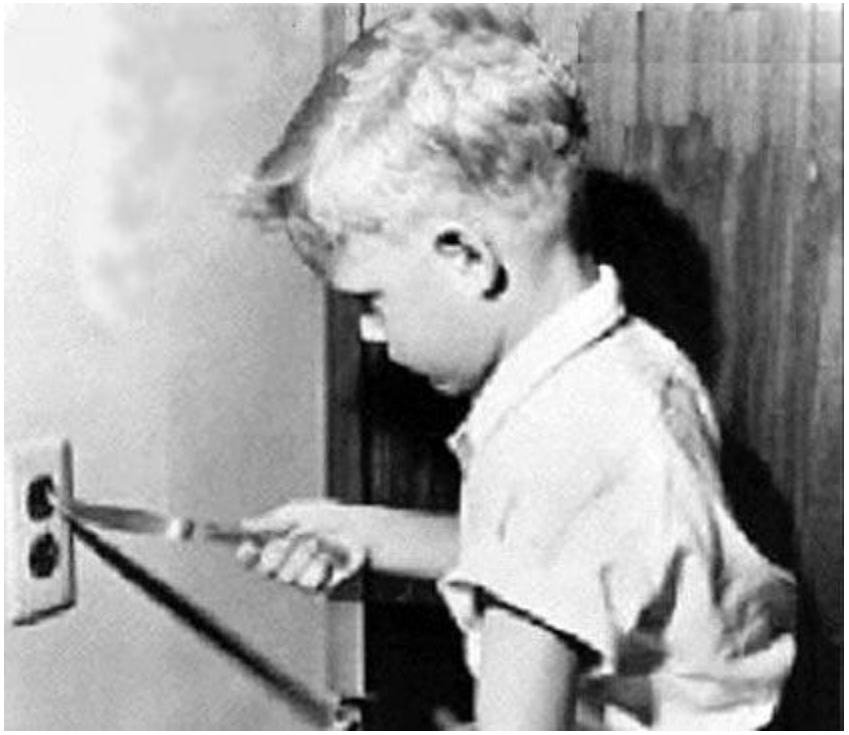
© Copyright John Thomas 2016

3. Software Allows Unlimited System Complexity

- Complexity (coupling) means can no longer
  - Plan, understand, anticipate, and guard against all undesired system behavior
  - Exhaustively test to get out all design errors
- **Context** determines whether software is safe
  - Ariane 4 software was safe but when reused in Ariane 5, the spacecraft exploded
  - DAL, Rigor of Development, SIL will not ensure software is safe
  - Not possible to look at software alone and determine its "safety"

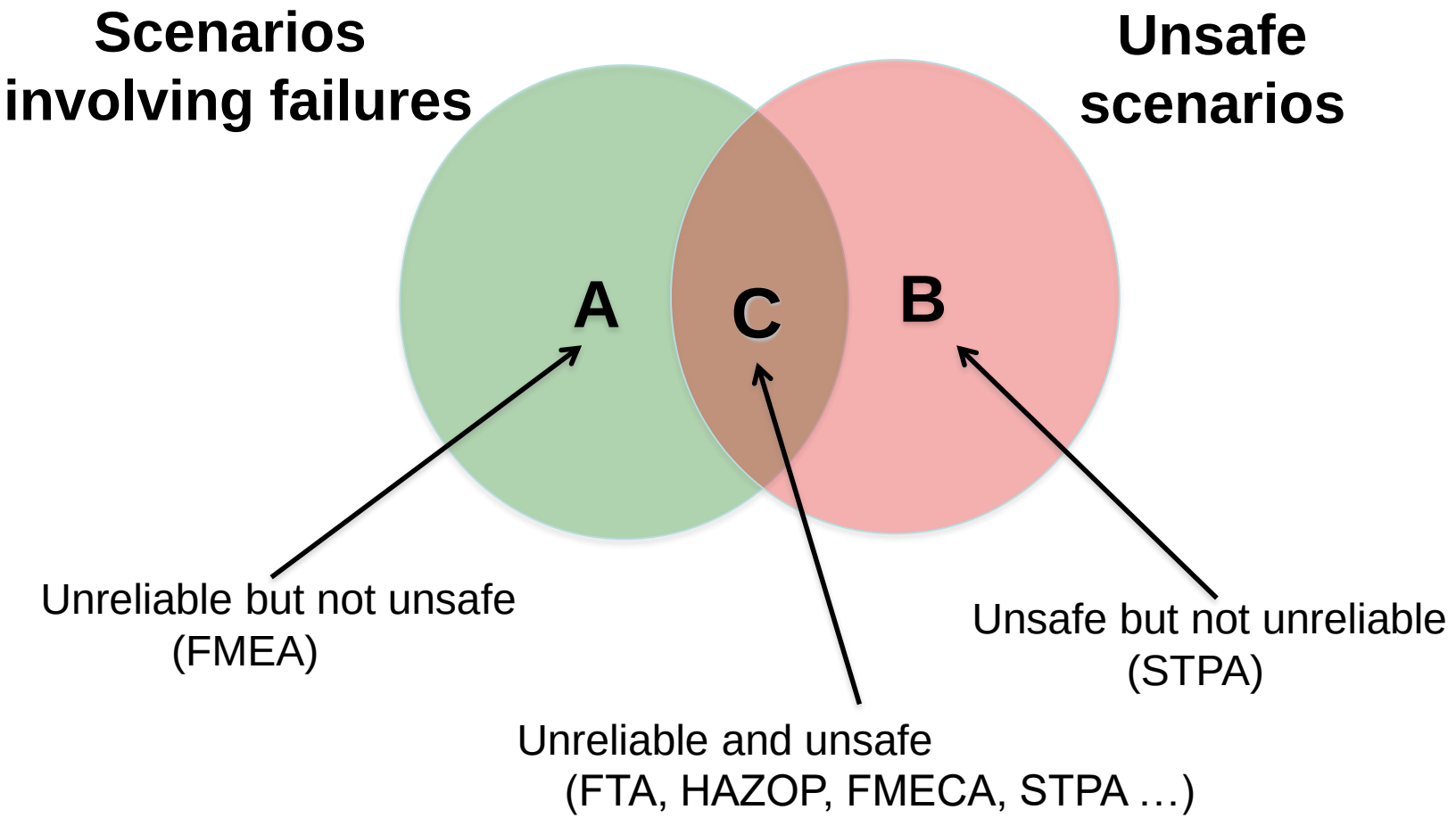
Safe or Unsafe?

Safety Depends on Context



Reliability is NOT equal to safety in complex, software-intensive systems

Confusing Safety and Reliability



Preventing Component or Functional Failures is Not Enough

© Copyright Nancy Leveson, June 2011

4. 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

© Copyright Nancy Leveson, June 2011

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.

© Copyright John Thomas 2016

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.

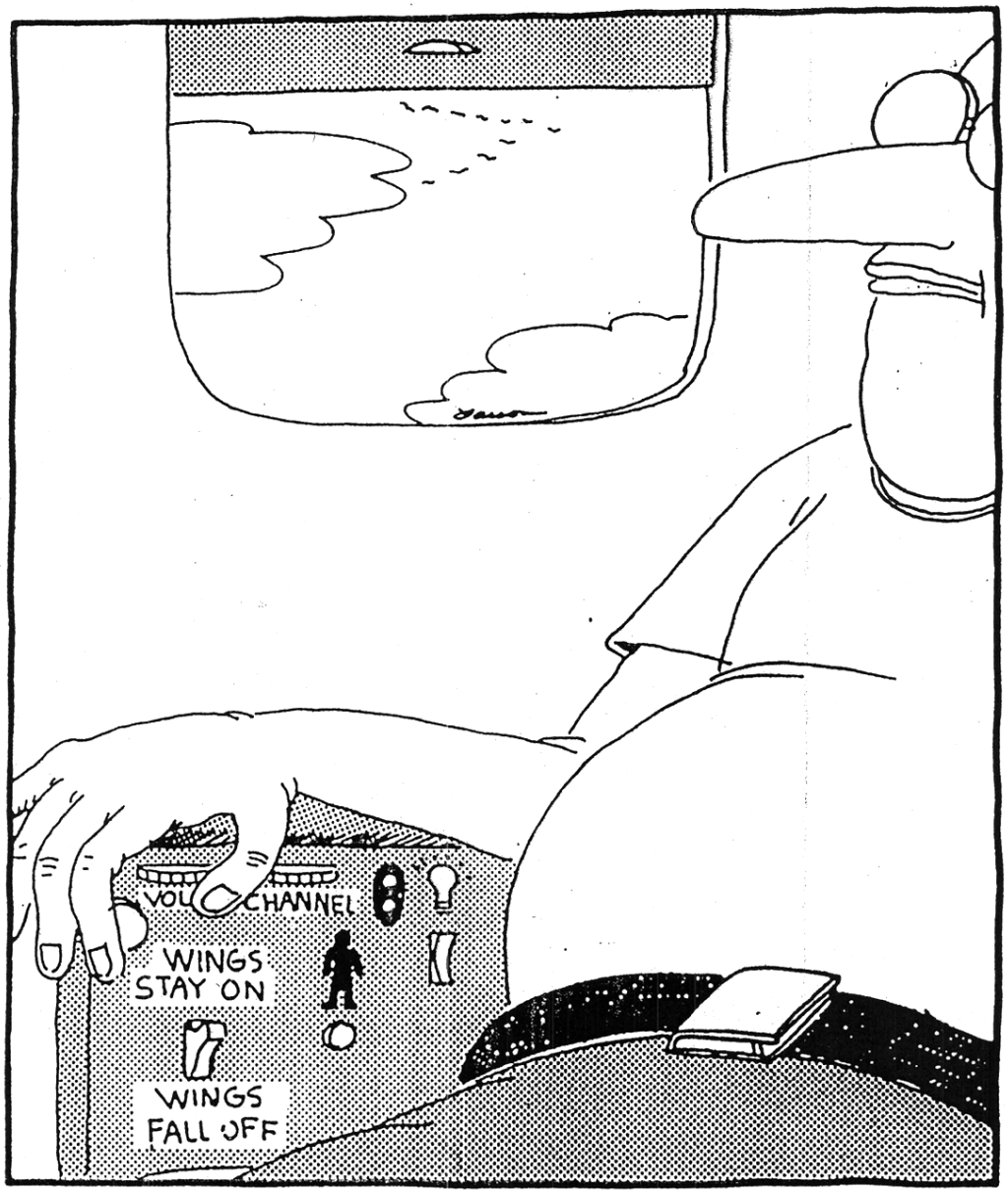
© Copyright John Thomas 2016

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

© Copyright John Thomas 2016



Fumbling for his recline button Ted unwittingly instigates a disaster

© Copyright Nancy Leveson, June 2011

A Systems View of Operator Error

- Operator error is a symptom, not a cause
- All behavior affected by context (system) in which occurs
  - Role of operators is changing in software-intensive systems as is the errors they make
  - Designing systems in which operator error inevitable and then blame accidents on operators rather than designers
- To do something about operator error, must look at system in which people work:
  - Design of equipment
  - Usefulness of procedures
  - Existence of goal conflicts and production pressures
- **Human error is a symptom of a system that needs to be redesigned**

© Copyright Nancy Leveson, June 2011

Human factors concentrates on the "screen out"

Hardware/Software engineering concentrates on the "screen in"

© Copyright Nancy Leveson, June 2011

Not enough attention on integrated system as a whole

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

© Copyright Nancy Leveson, June 2011

What is STAMP and how does it differ from what people do now?

© Copyright Nancy Leveson, June 2011

The Problem is Complexity

Ways to Cope with Complexity

- Analytic Reduction
- Statistics
- Systems Theory

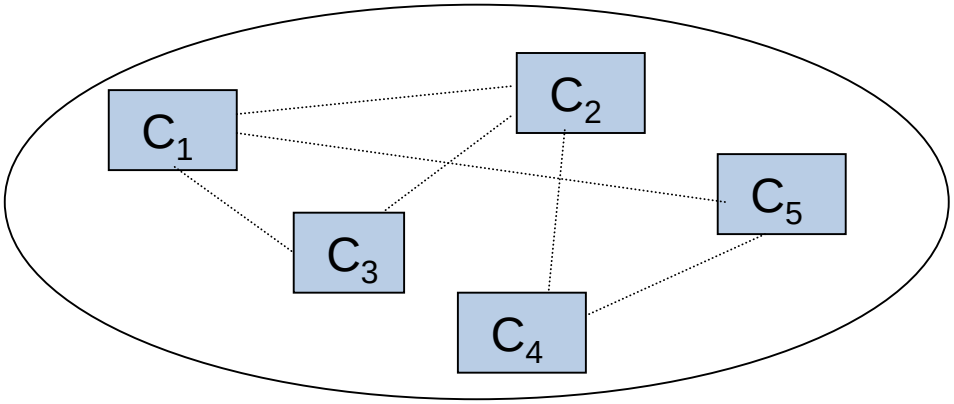
© Copyright Nancy Leveson, June 2011

Traditional Approach to Coping with Complexity

Analytic Reduction ("Divide and Conquer")

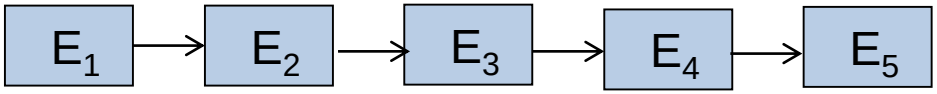
1. Divide system into separate parts

Physical/Functional: Separate into distinct components



Components interact
In direct ways

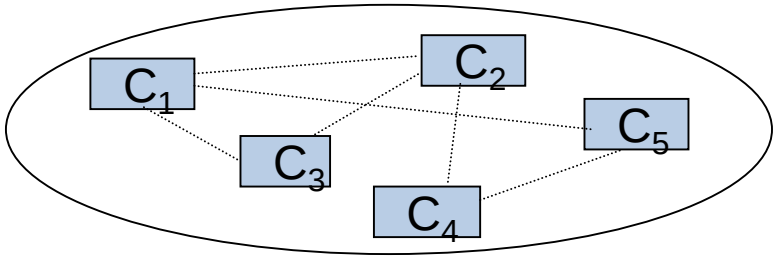
Behavior: Separate into events over time

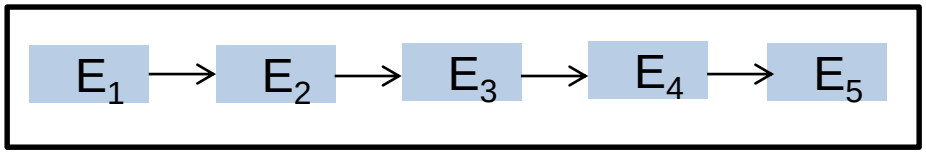


Each event is the direct
result of the preceding event

© Copyright Nancy Leveson, June 2011

Analytic Reduction (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

© Copyright Nancy Leveson, June 2011

Bottom Line

- These assumptions are no longer true in our
  - Tightly coupled
  - Software intensive
  - Highly automated
  - Connected

  engineered systems
- Need a new theoretical basis
  - *System theory* can provide it

Traditional Approach to Safety

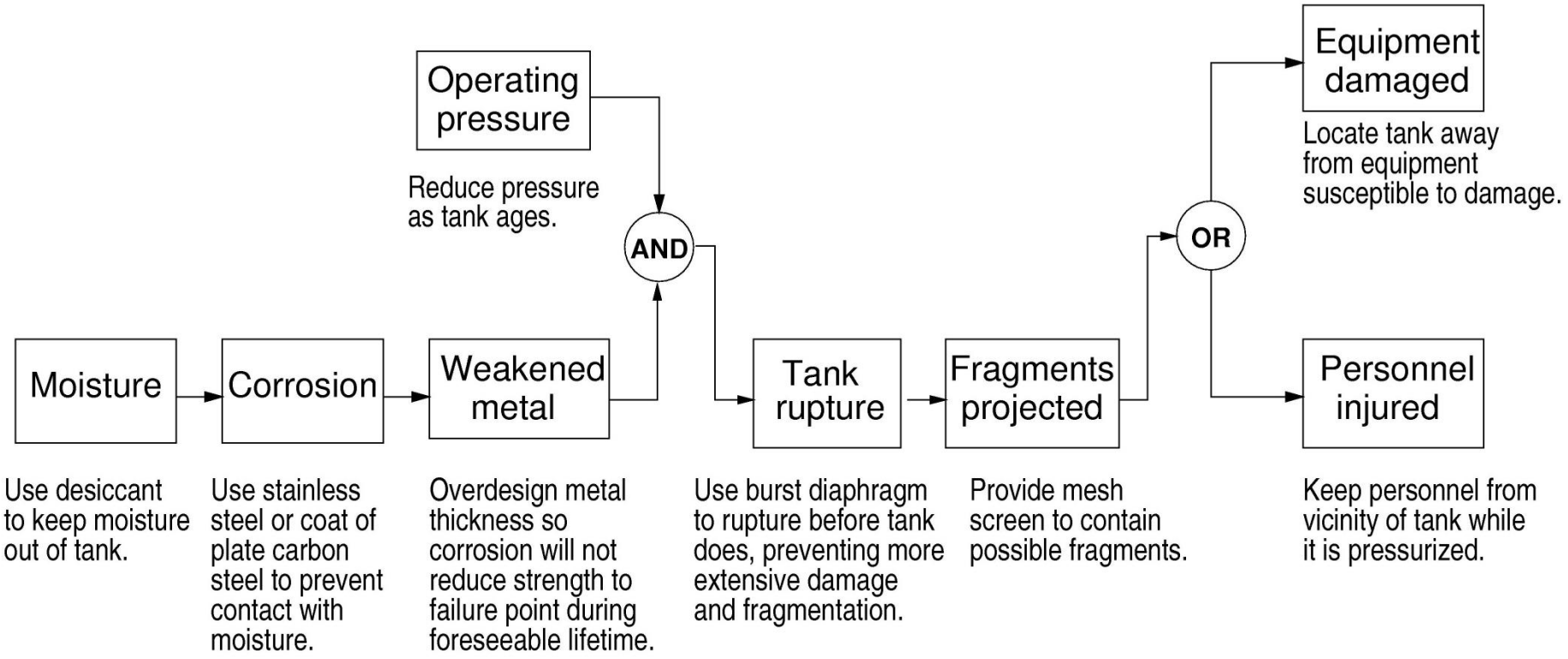
- Reductionist
  - Divide system into components
  - Assume accidents are caused by component failure
  - Identify chains of directly related physical or logical (functional) component failures that can lead to a loss
  - Evaluate reliability of components separately and later combine analysis results into a system reliability value

Note: Assume randomness in the failure events so can derive probabilities for a loss

- **Software and humans do not satisfy this assumption**

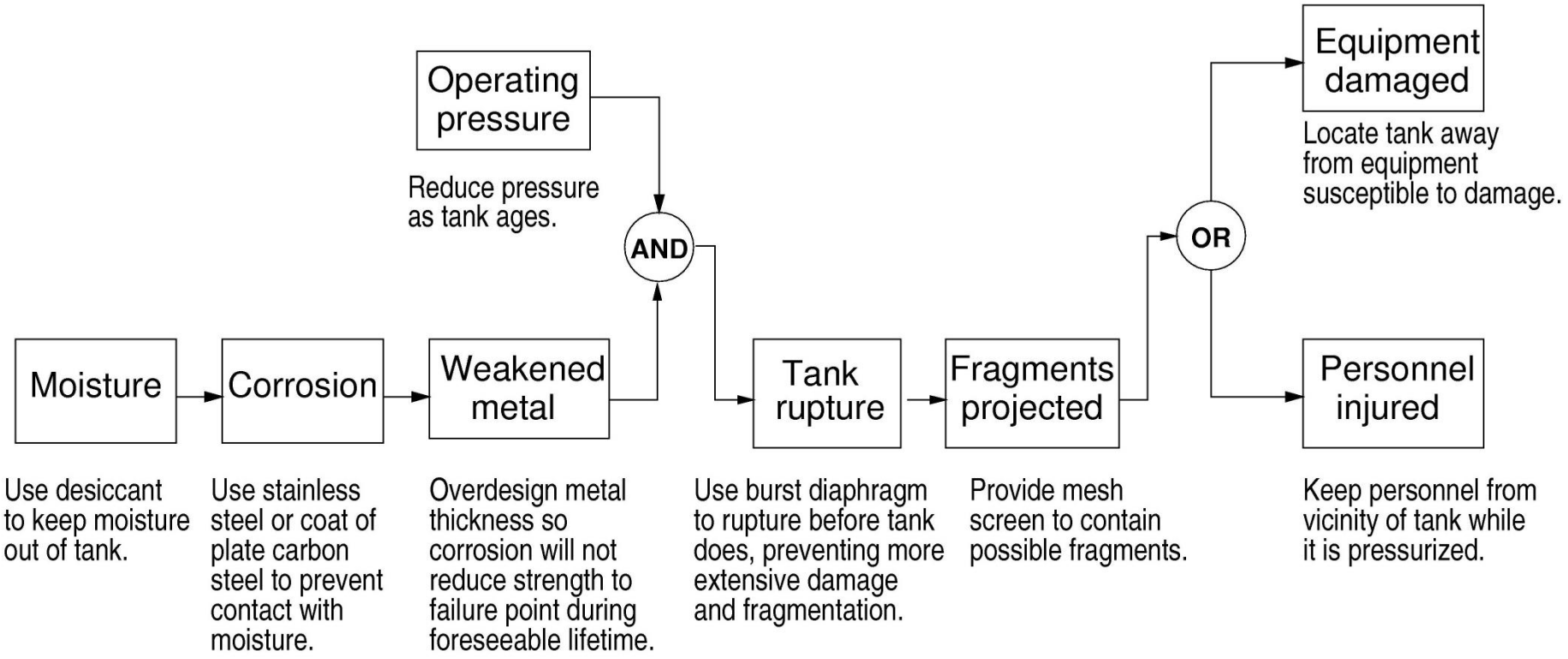
© Copyright Nancy Leveson, June 2011

Chain-of-events example



© Copyright Nancy Leveson, June 2011

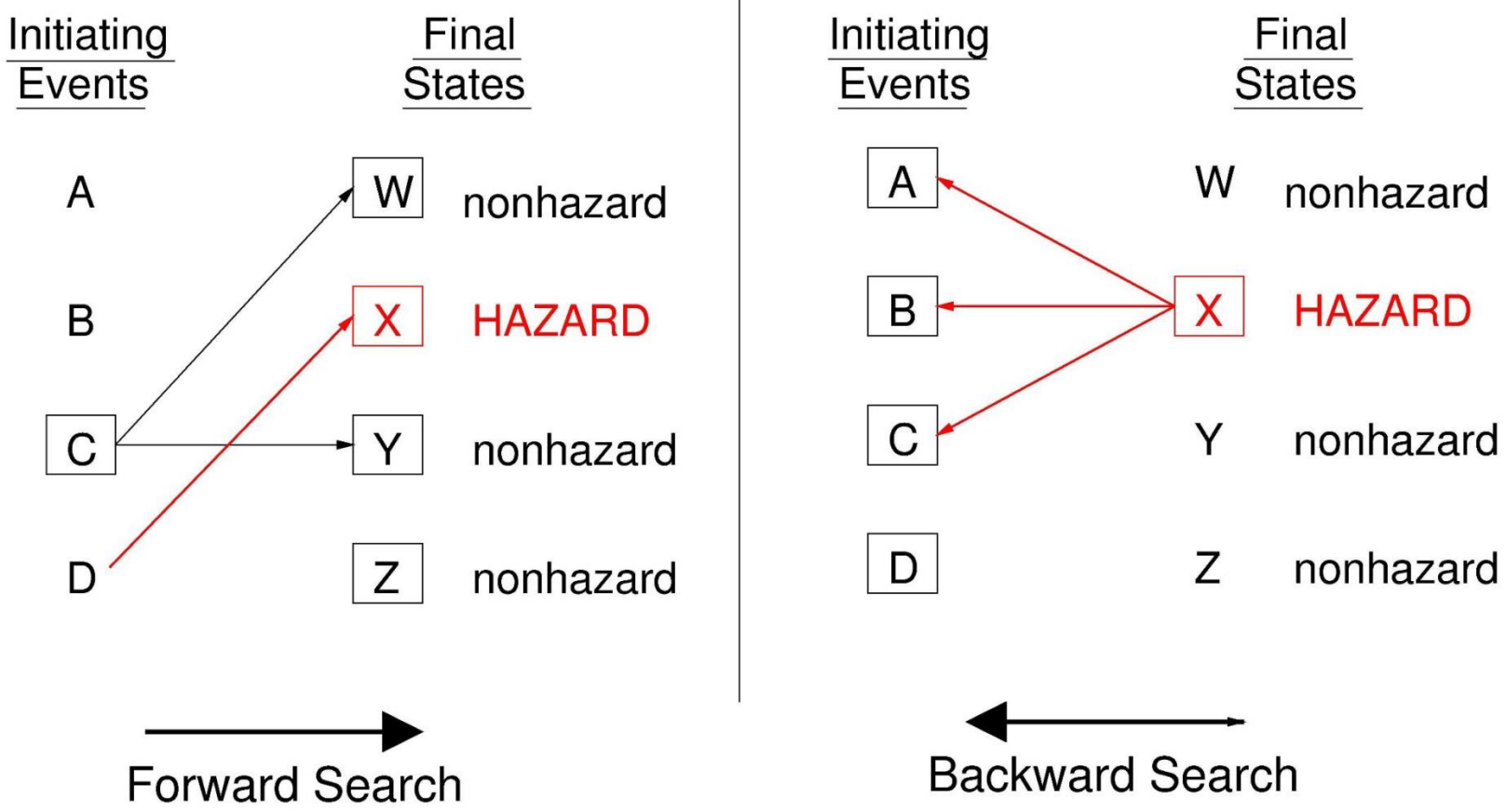
Chain-of-events example



How are the event chains identified?

© Copyright Nancy Leveson, June 2011

Forward vs. Backward Search



Accidents as Chains of Failure Events

- Forms the basis for most safety engineering and reliability engineering analysis:

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

and design (concentrate on dealing with component failure):

Redundancy and barriers (to prevent failure propagation)

High component integrity and overdesign

Fail-safe design

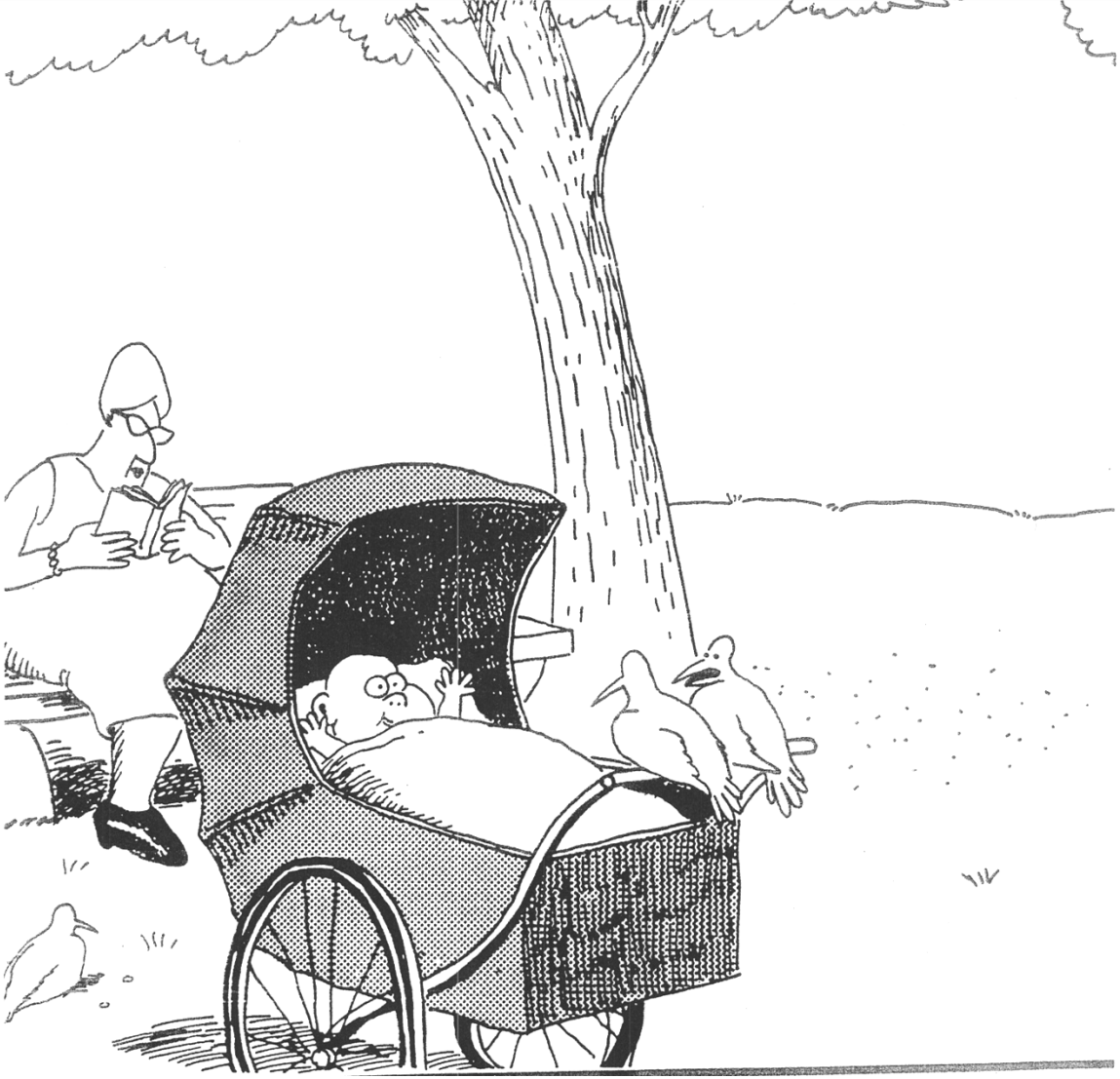
(humans) Operational procedures, checklists, training, ….

© Copyright Nancy Leveson, June 2011

Standard Approach does not Handle

- Component interaction accidents
- Systemic factors (affecting all components and barriers)
- Software and software requirements errors
- Human behavior (in a non-superficial way)
- System design errors
- Indirect or non-linear interactions and complexity
- Migration of systems toward greater risk over time (e.g., in search for greater efficiency and productivity)

© Copyright Nancy Leveson, June 2011



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

© Copyright Nancy Leveson, June 2011

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

© Copyright Nancy Leveson, June 2011

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
- Basis for System Engineering and System Safety

© Copyright Nancy Leveson, June 2011

Systems Theory (2)

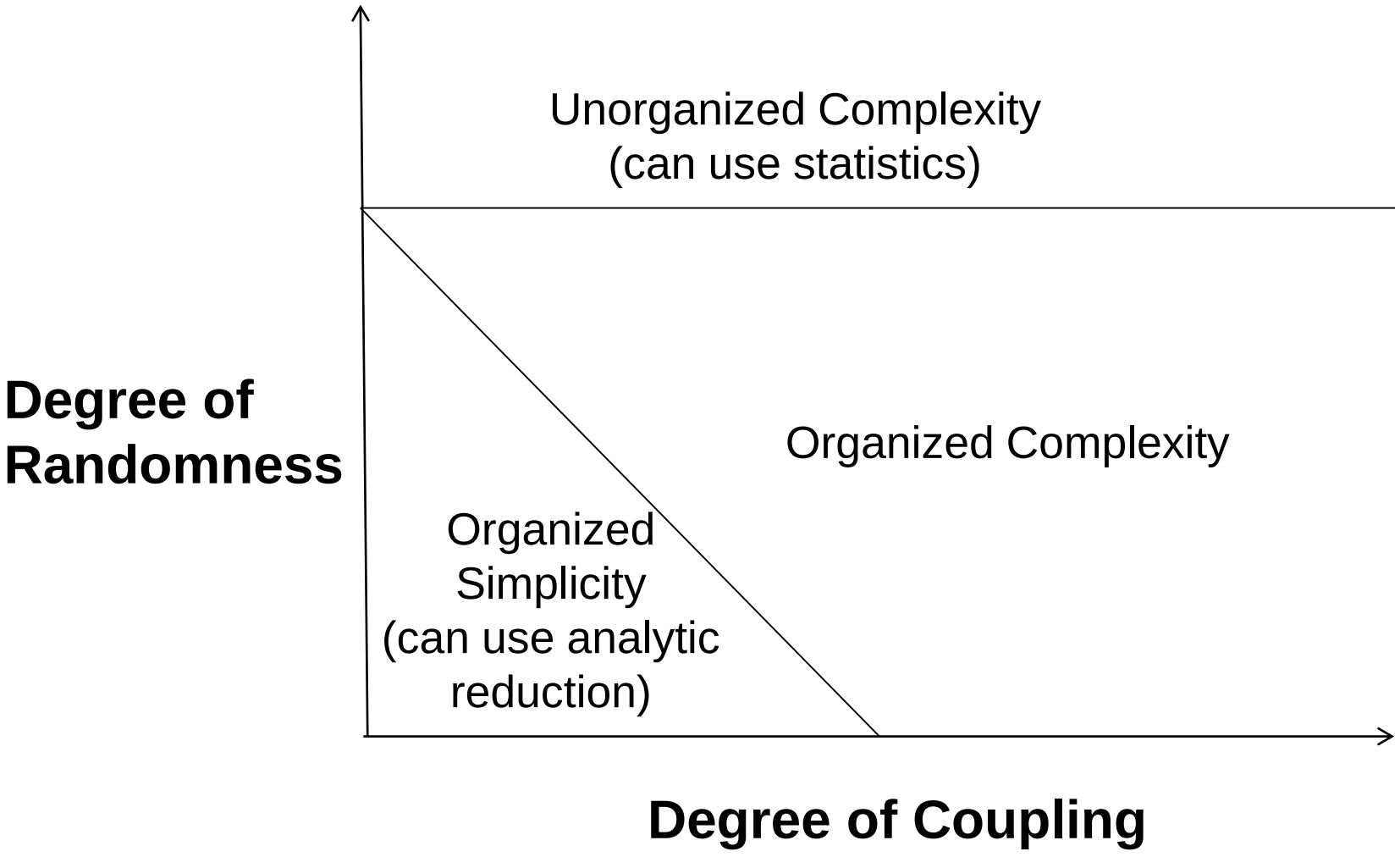
- Focuses on systems taken as a whole, not on parts taken separately
- Emergent properties
  - Some properties can only be treated adequately in their entirety, taking into account all social and technical aspects

    "The whole is greater than the sum of the parts"

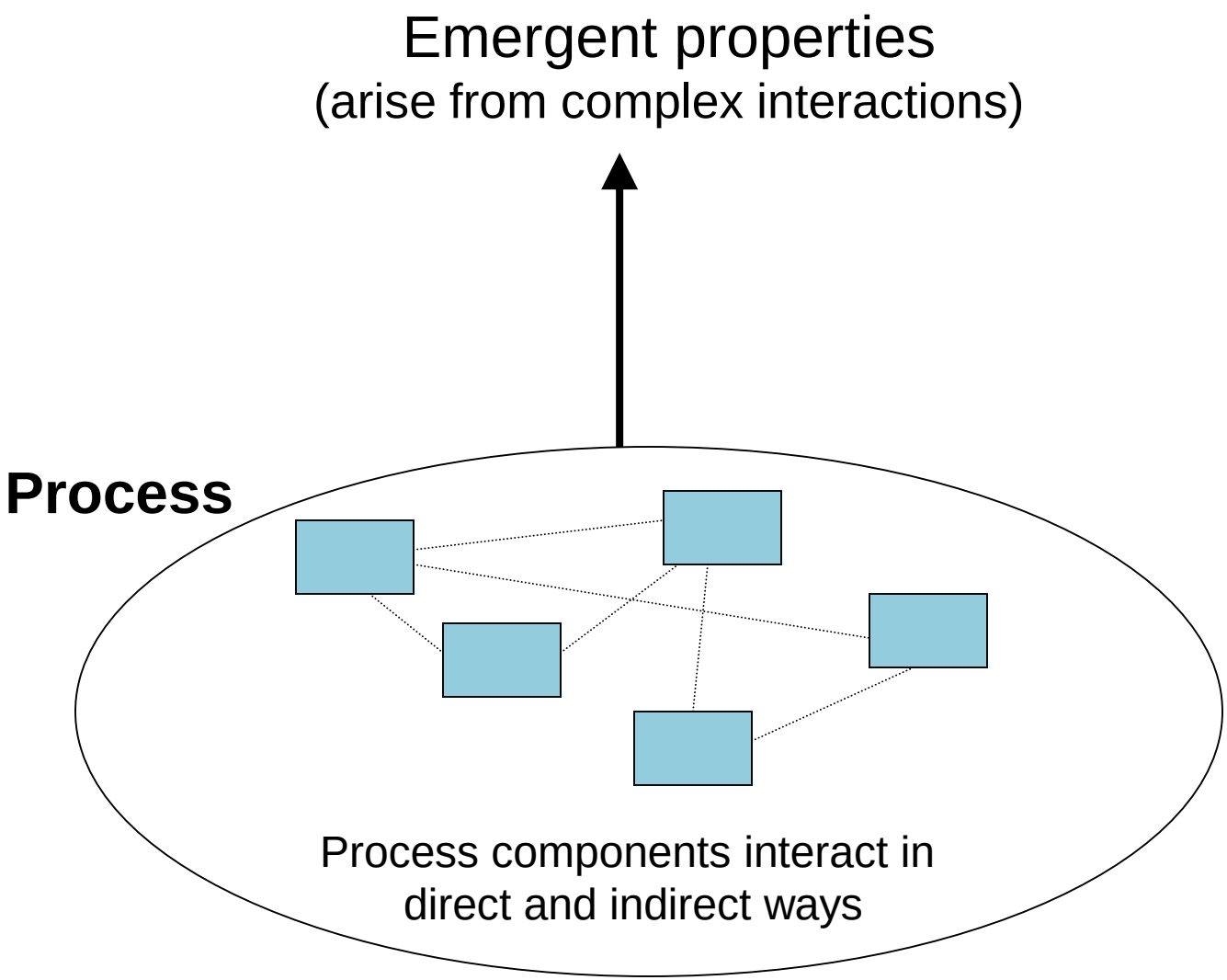
  - These properties arise from relationships among the parts of the system

    How they interact and fit together

© Copyright Nancy Leveson, June 2011



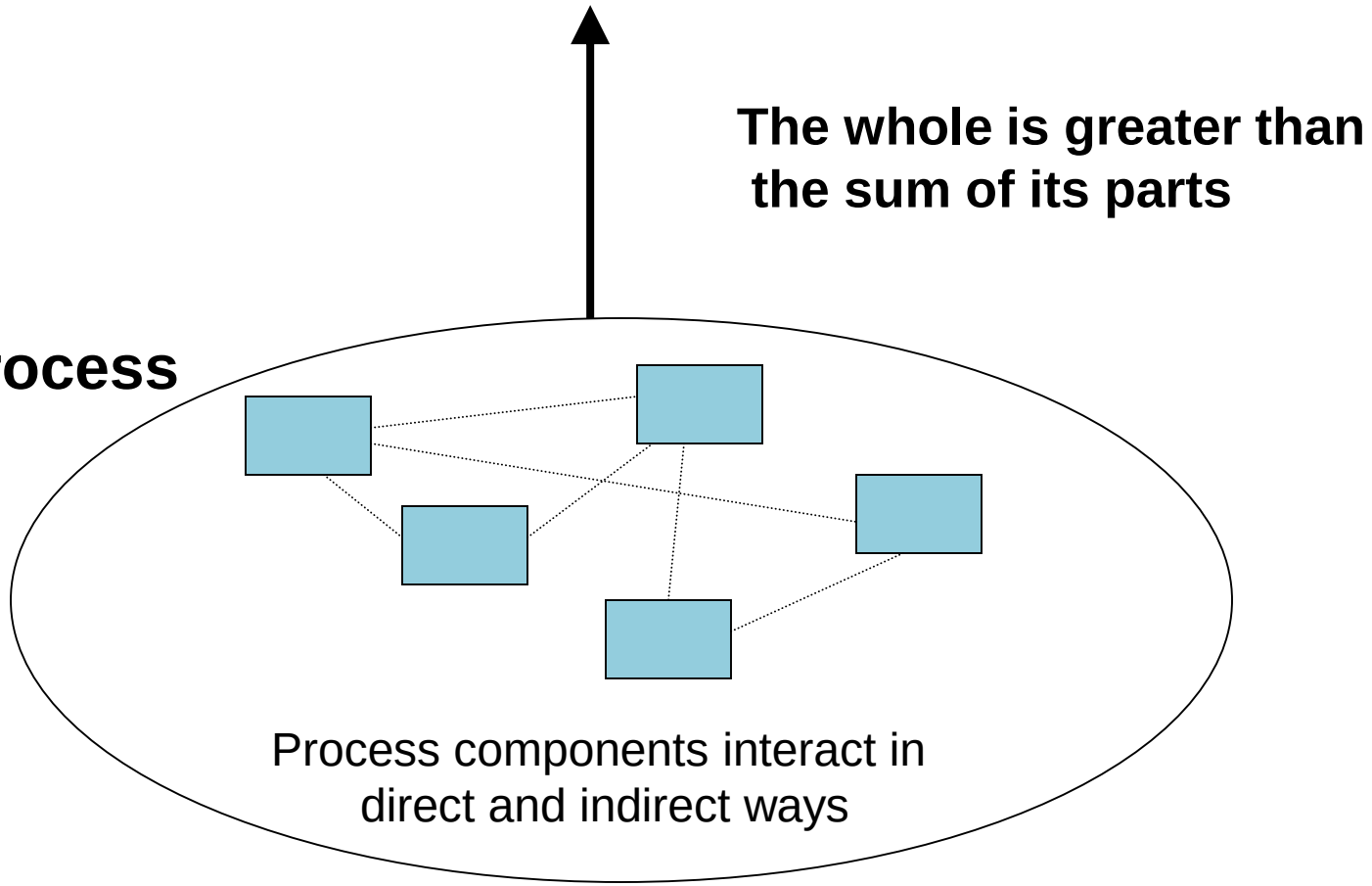
© Copyright Nancy Leveson, June 2011



Safety and security are emergent properties

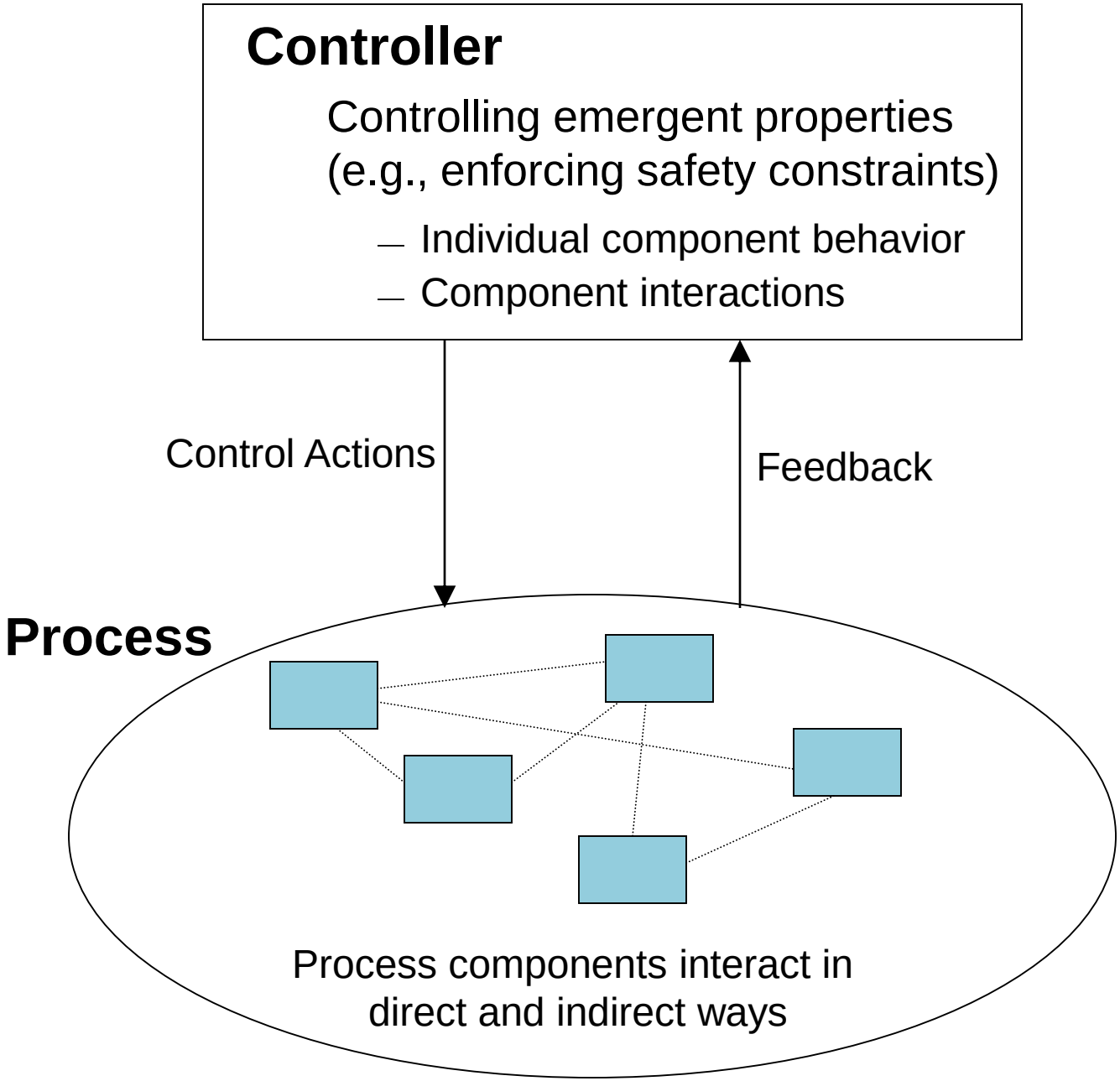
© Copyright Nancy Leveson, June 2011

Emergent properties
(arise from complex interactions)

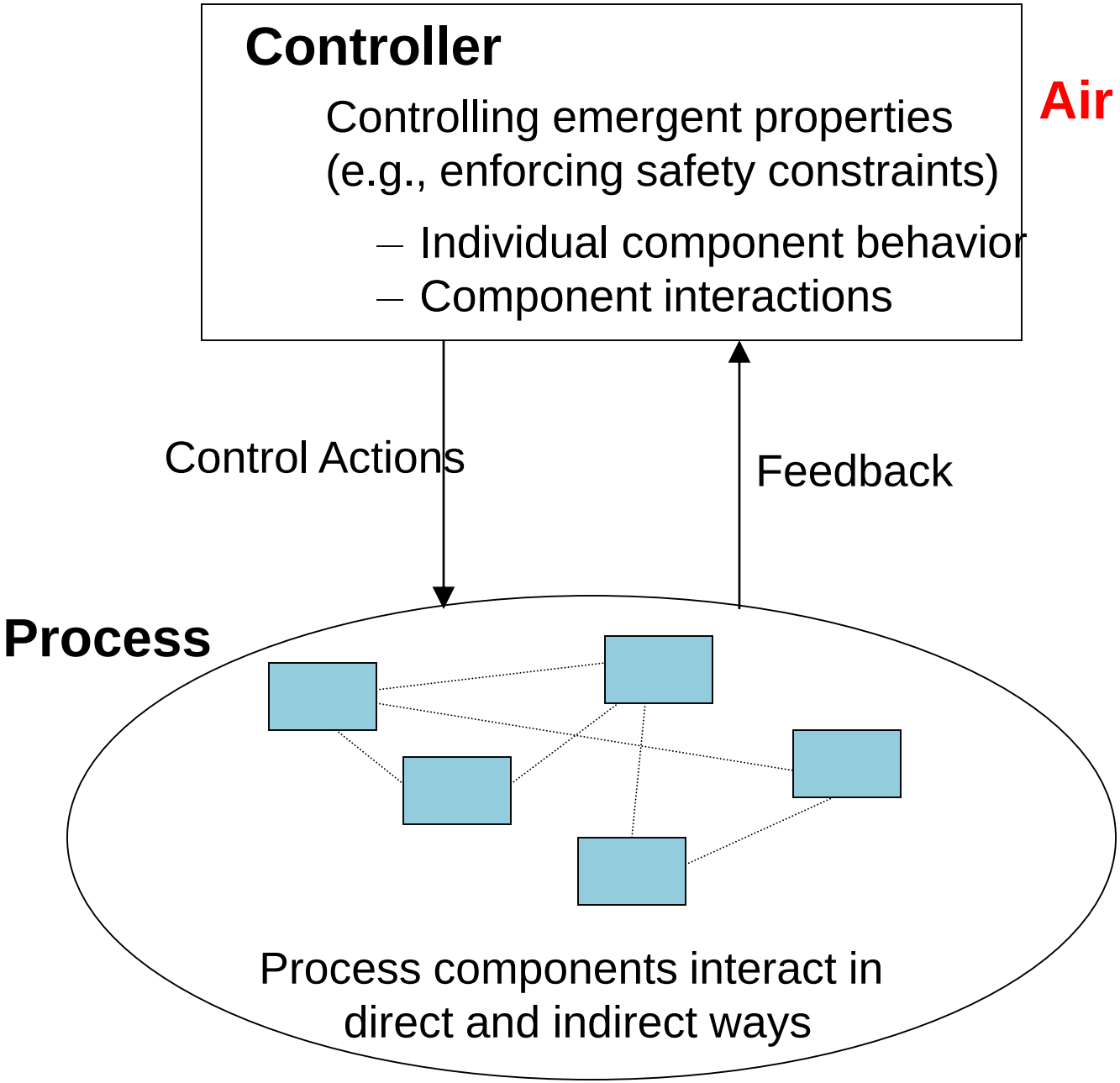


Safety and security are emergent properties

© Copyright Nancy Leveson, June 2011



© Copyright Nancy Leveson, June 2011



Air Traffic Control:
Safety
Throughput

© Copyright Nancy Leveson, June 2011

Controls/Controllers Enforce Safety Constraints

- Power must never be on when access door open
- Two aircraft/automobiles must not violate minimum separation
- Aircraft must maintain sufficient lift to remain airborne
- 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
- Public health system must prevent exposure of public to contaminated water and food products
- Pressure in a offshore well must be controlled

© Copyright Nancy Leveson, June 2011

Controls/Controllers Enforce Safety Constraints (2)

- Runway incursions and operations on wrong runways or taxiways must be prevented
- Bomb must not detonate without positive action by authorized person
- Submarine must always be able to blow the ballast tanks and return to surface
- Truck drivers must not drive when sleep deprived
- Fire must not be initiated on a friendly target

These are the High-Level Functional Safety Requirements to Address During Design

© Copyright Nancy Leveson, June 2011

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

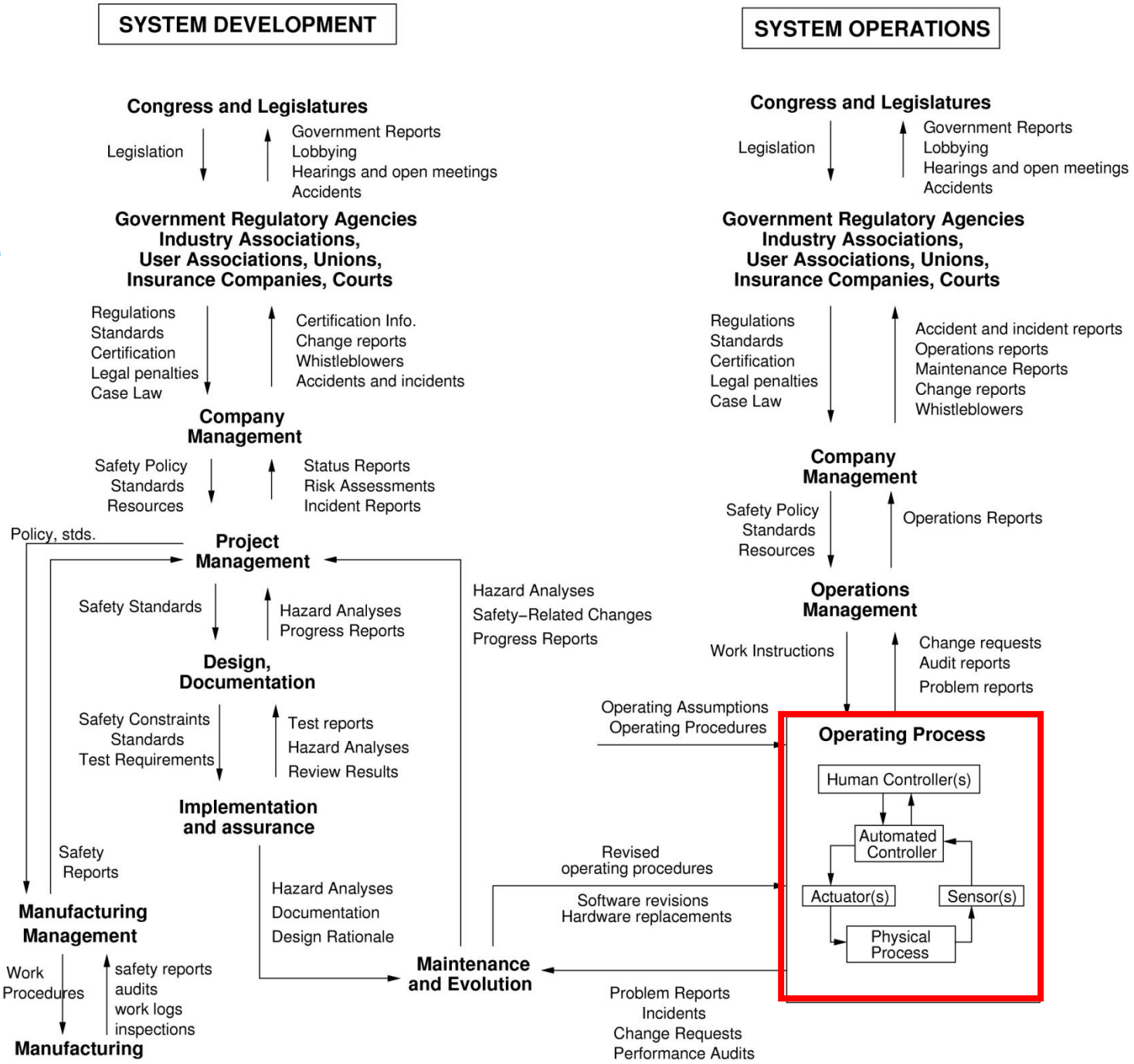
- Manufacturing processes and procedures
- Maintenance processes
- Operations

or through social controls

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

© Copyright Nancy Leveson, June 2011

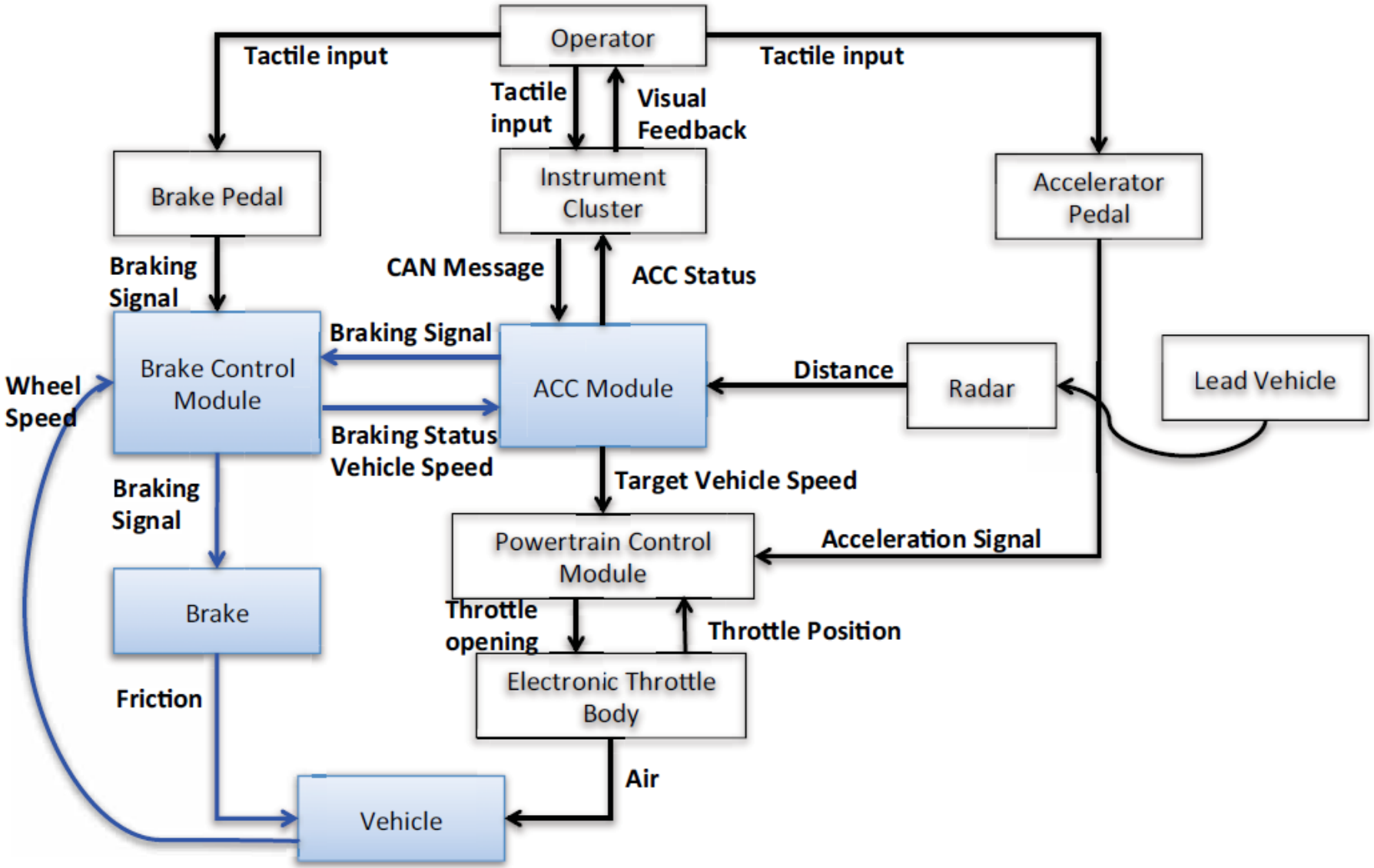
Example Safety Control Structure (SMS)



© Copyright Nancy Leveson, June 2011

(Qi Hommes)

Example: ACC – BCM Control Loop

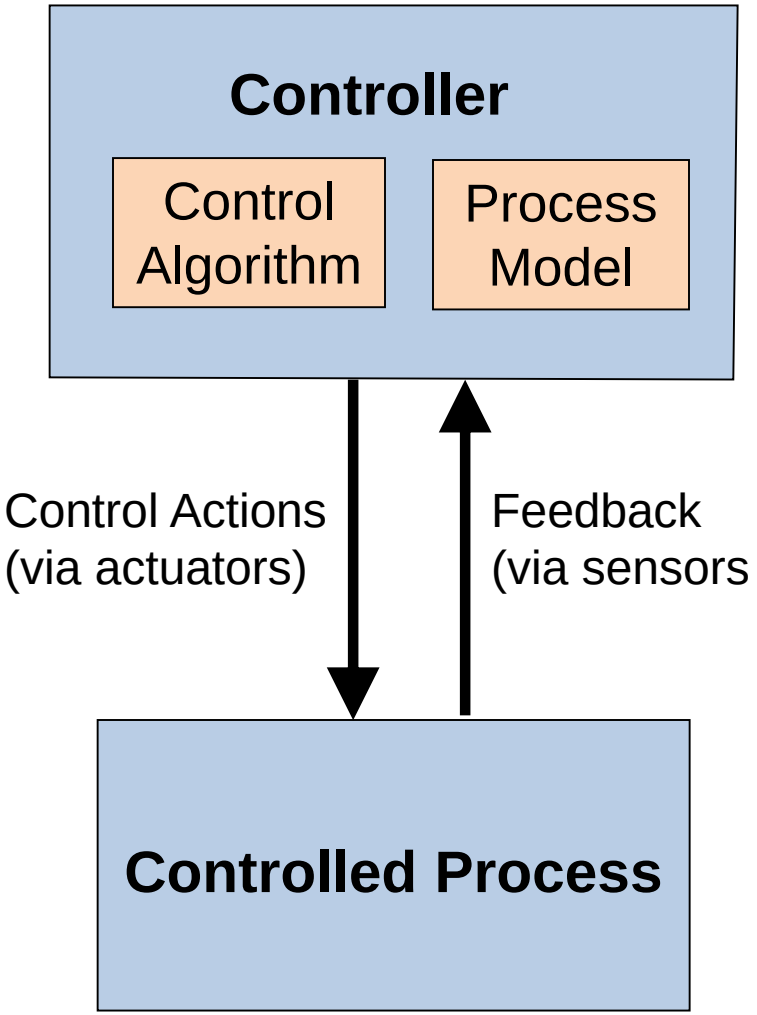


© Copyright Nancy Leveson, June 2011

Safety as a Control Problem

- **Goal: Design an effective control structure that eliminates or reduces adverse events**.
  - Need clear definition of expectations, responsibilities, authority, and accountability at all levels of safety control structure
  - Need appropriate feedback
  - Entire control structure must together enforce the system safety property (constraints)
    - Physical design (inherent safety)
    - Operations
    - Management
    - Social interactions and culture

Role of Process Models in Control

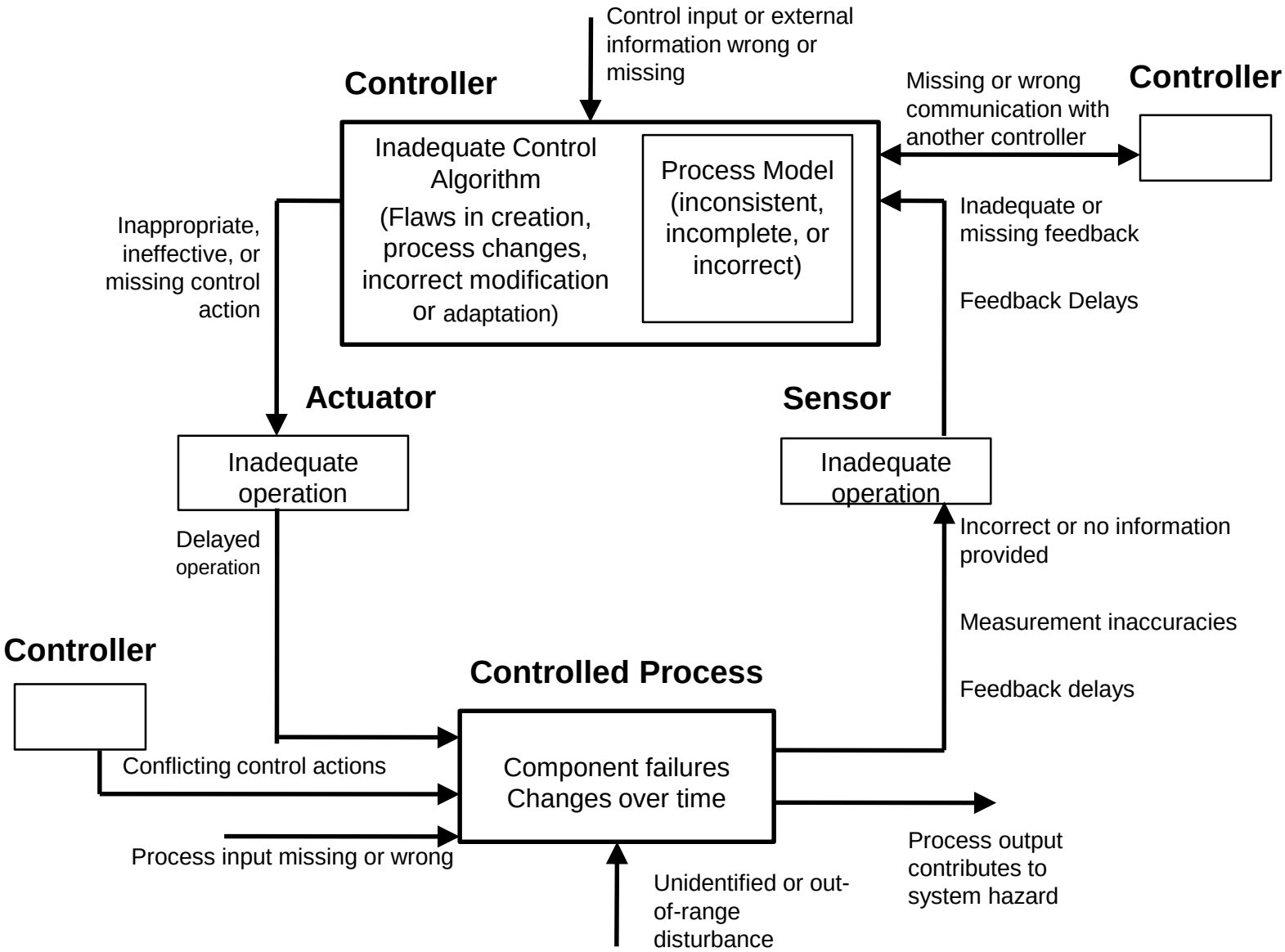


- Controllers use a **process model** to determine control actions
- Accidents often occur when the process model is incorrect
- Four types of unsafe control actions:
  - Control commands required for safety are not given
  - Unsafe ones are given
  - Potentially safe commands given too early, too late
  - Control stops too soon or applied too long

(Leveson, 2003); (Leveson, 2011)

© Copyright Nancy Leveson, June 2011

Identifying Causal Scenarios for Unsafe Control



© Copyright Nancy Leveson, June 2011

STAMP (System-Theoretic Accident Model and Processes)

- Defines safety/security as a control problem (vs. failure problem)
- Applies to very complex systems
- Includes software, humans, operations, management
- Based on general system theory
- Expands the traditional model of the accident causation (cause of losses)
  - Not just a chain of directly related failure events
  - Losses are complex processes

© Copyright Nancy Leveson, June 2011

Safety as a Dynamic Control Problem (STAMP)

- Hazards result from lack of enforcement of safety constraints in system design and operations
- Goal is to control the behavior of the components and systems as a whole to ensure safety constraints are enforced in the operating system
- A change in emphasis:

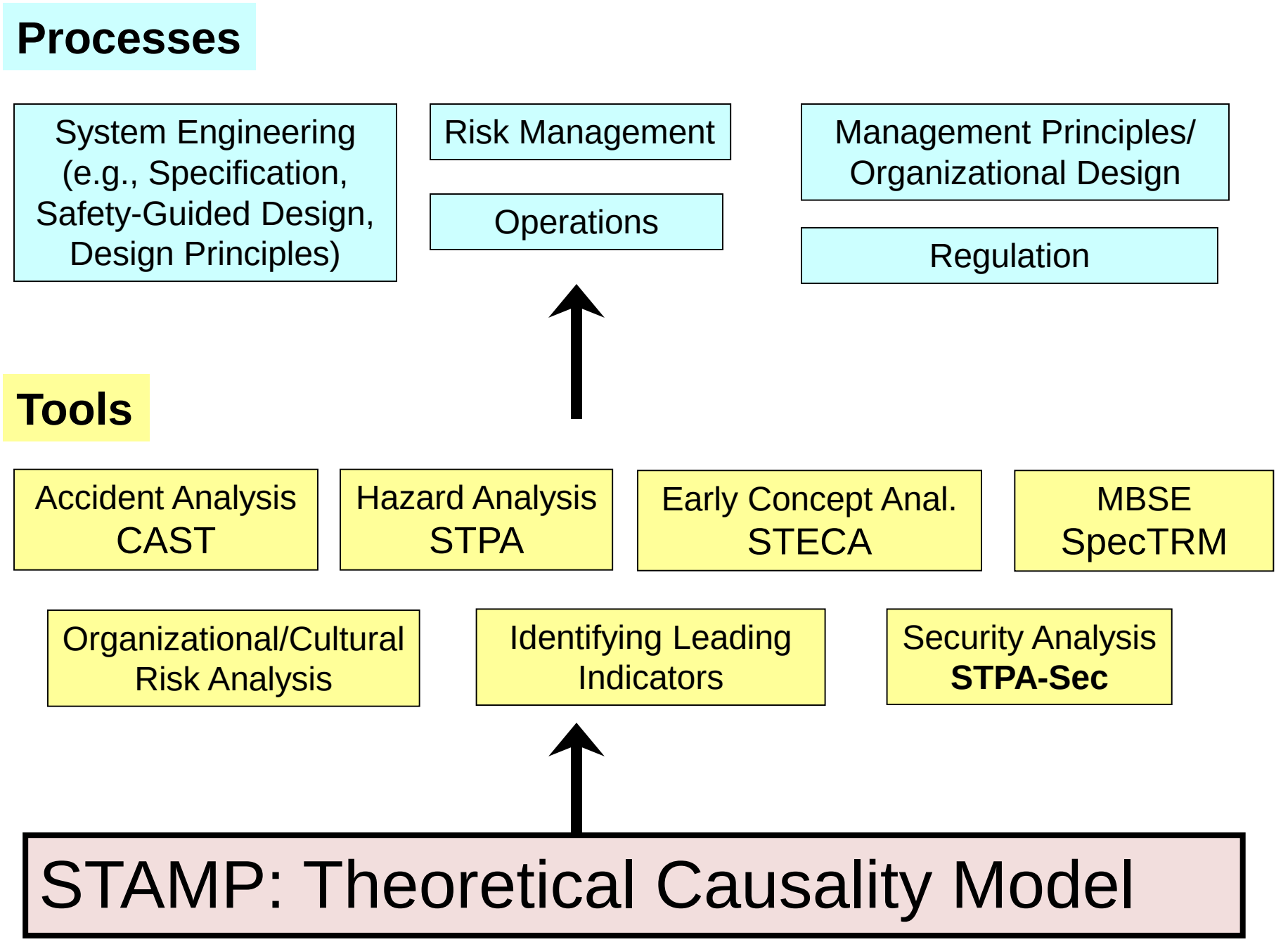
"enforce safety/security constraints on system behavior"

(note that enforcing constraints might require preventing failures or handling them but includes more than that)

© Copyright Nancy Leveson, June 2011

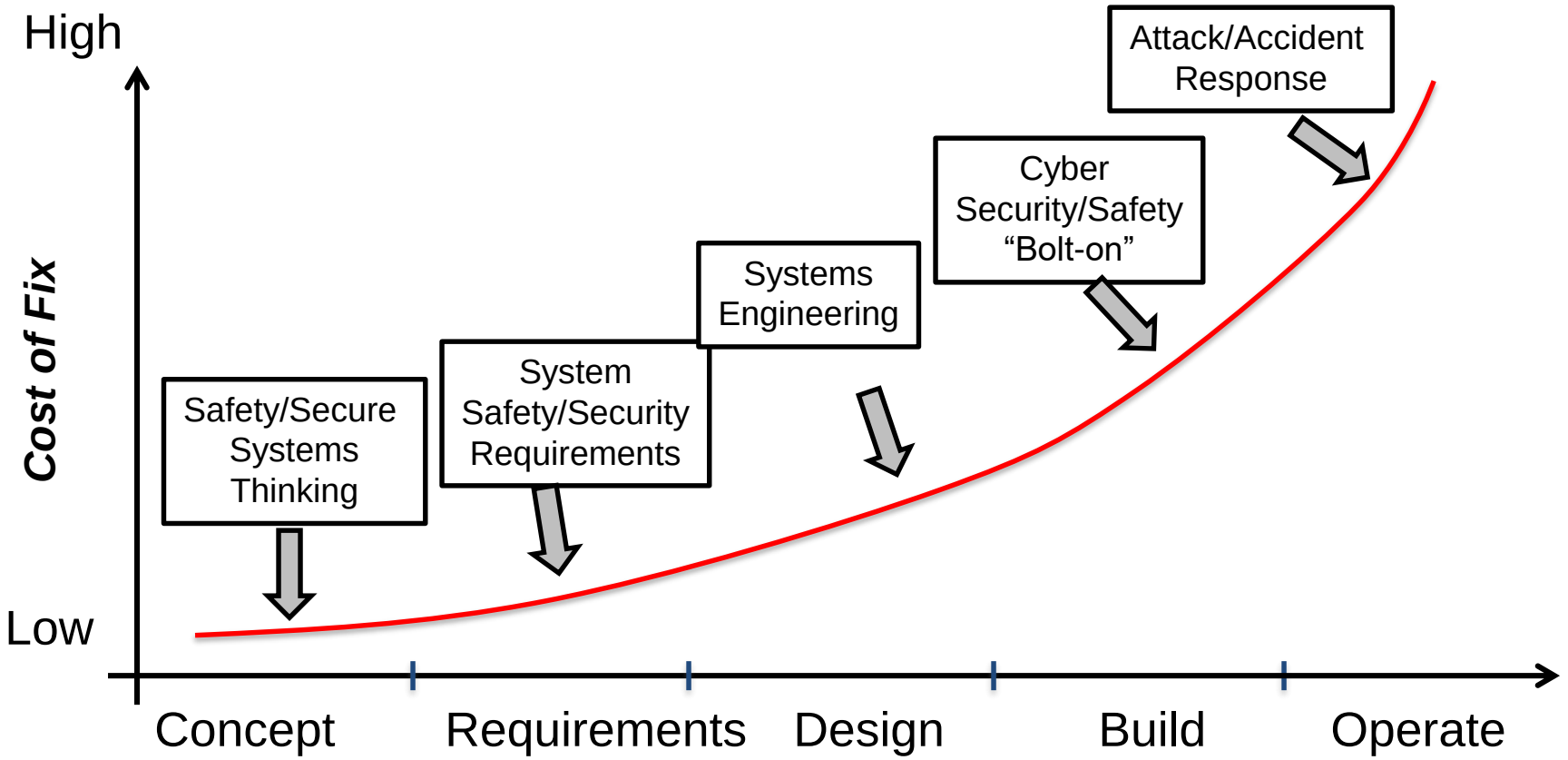
What kinds of tools are available?

© Copyright Nancy Leveson, June 2011



© Copyright Nancy Leveson, June 2011

Build safety and security into system from beginning



© Copyright Nancy Leveson, June 2011

How is it being used?
Does it work?
Is it useful?

© Copyright Nancy Leveson, June 2011

Is it Practical?

- STPA has been or is being used in a large variety of industries
  - Spacecraft
  - Aircraft / Air Traffic Control
  - UAVs (RPAs)
  - Defense systems
  - Automobiles
  - Medical Devices and Hospital Safety
  - Chemical plants
  - Oil and Gas
  - Nuclear and Electric Power
  - Finance
  - Robotic Manufacturing / Workplace Safety
  - Etc.

© Copyright Nancy Leveson, June 2011

Uses Beyond Traditional System Safety

- Quality
- Producibility (of aircraft)
- Nuclear security
- Banking and finance
- Engineering process optimization
- Organizational culture
- Workplace safety

© Copyright Nancy Leveson, June 2011

Is it Effective?

- Most of these systems are very complex (e.g., the new U.S. missile defense system)
- In all cases where a comparison was made (to FTA, HAZOP, FMEA, ETA, etc.)
  - STPA found the same hazard causes as the old methods
  - Plus it found more causes than traditional methods
  - In some evaluations, found accidents that had occurred that other methods missed (e.g., EPRI)
  - Cost was orders of magnitude less than the traditional hazard analysis methods
  - Same results for security evaluations by CYBERCOM

© Copyright Nancy Leveson, June 2011

Summary

- More comprehensive and powerful approach to safety (and security and any emergent property)
- Includes social, organizational, operator, software-related factors
- Top-down system engineering approach; easily integrated into system engineering processes.
- Handles much more complex systems than traditional safety analysis approaches and costs less

© Copyright Nancy Leveson, June 2011

Paradigm Change

- Does not imply what previously done is wrong and new approach correct
- Einstein:

  "Progress in science (moving from one paradigm to another) is like climbing a mountain"

As move further up, can see farther than on lower points

© Copyright Nancy Leveson, June 2011

Paradigm Change (2)

New perspective does not invalidate the old one, but extends and enriches our appreciation of the valleys below

Value of new paradigm often depends on ability to accommodate successes and empirical observations made in old paradigm.

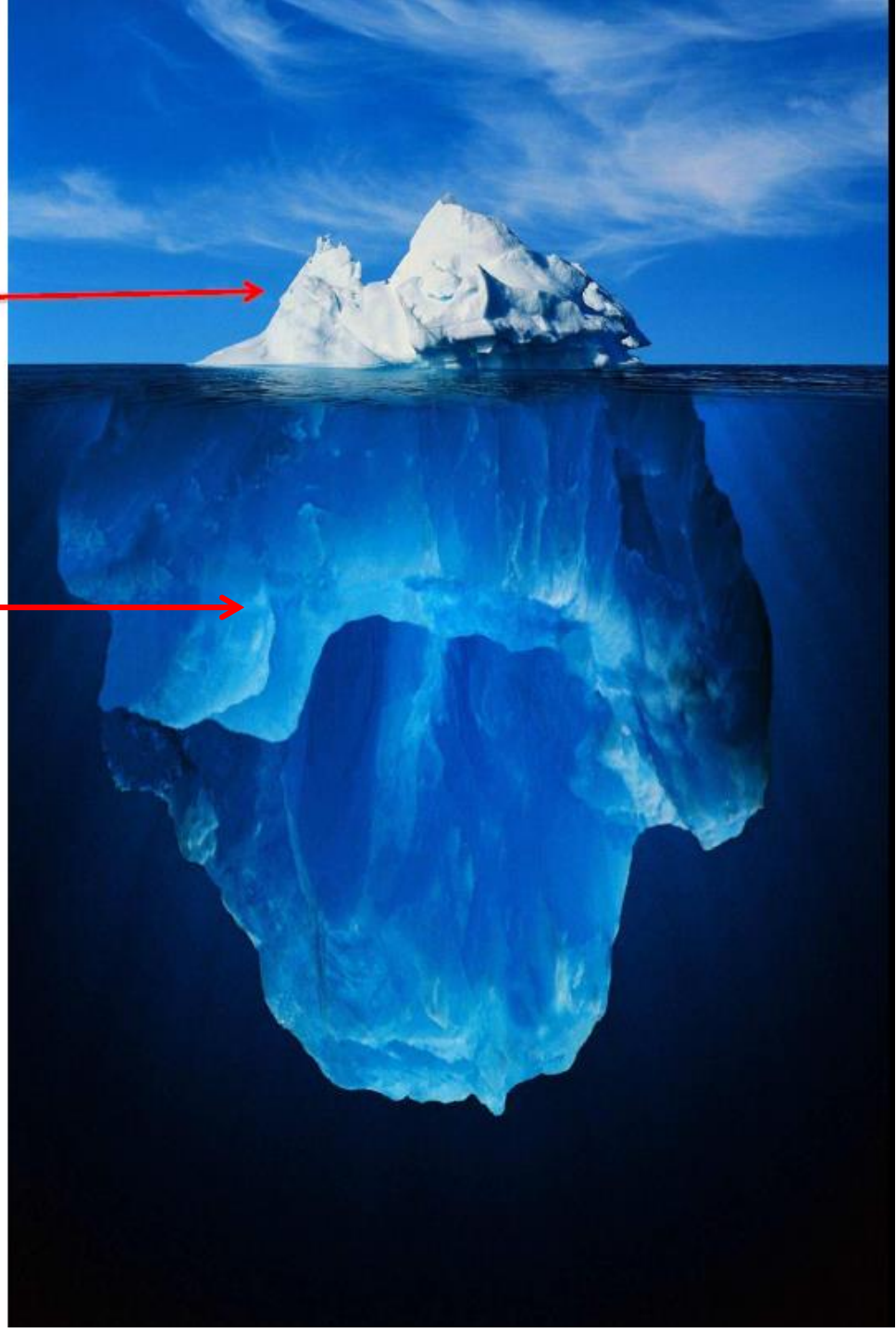
New paradigms offer a broader, rich perspective for interpreting previous answers.

© Copyright Nancy Leveson, June 2011

LEVERAGE

Event-based thinking

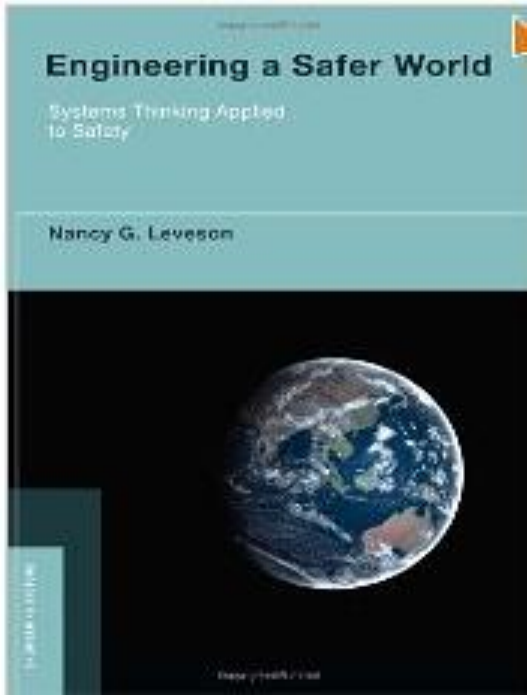
Systems Thinking



WYOUNG@MIT.EDU
© Copyright William Young, 2012

Nancy Leveson, *Engineering a Safer World:*

Systems Thinking Applied to Safety



MIT Press, January 2012

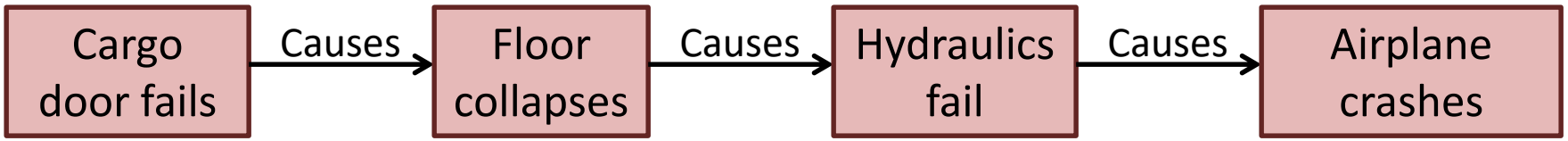
© Copyright Nancy Leveson, June 2011

Accident: An undesired and unplanned event that results in a loss, including loss of human life or human injury, property damage, environmental pollution, mission, damage to a company's reputation, etc.

(This is the same as what defense world calls a **Mishap)**

Domino "Chain of events" Model

DC-10:



Event-based

© Copyright Nancy Leveson, June 2011
© Copyright John Thomas 2013

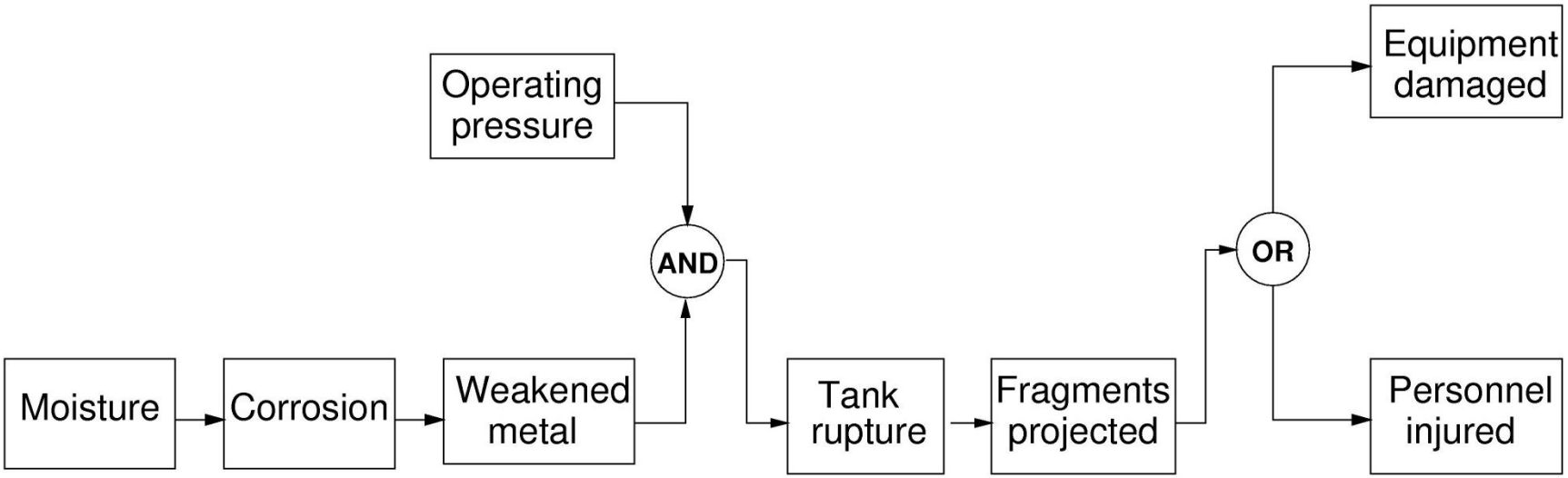
Event Chain

- E1: Worker washes pipes without inserting a slip blind.
- E2: Water leaks into MIC tank
- E3: Gauges do not work
- E4: Operator does not open valve to relief tank
- E3: Explosion occurs
- E4: Relief valve opens
- E5: Flare tower, vent scrubber, water curtain do not work
- E5: MIC vented into air
- E6: Wind carries MIC into populated area around plant.

What was the “root cause”?

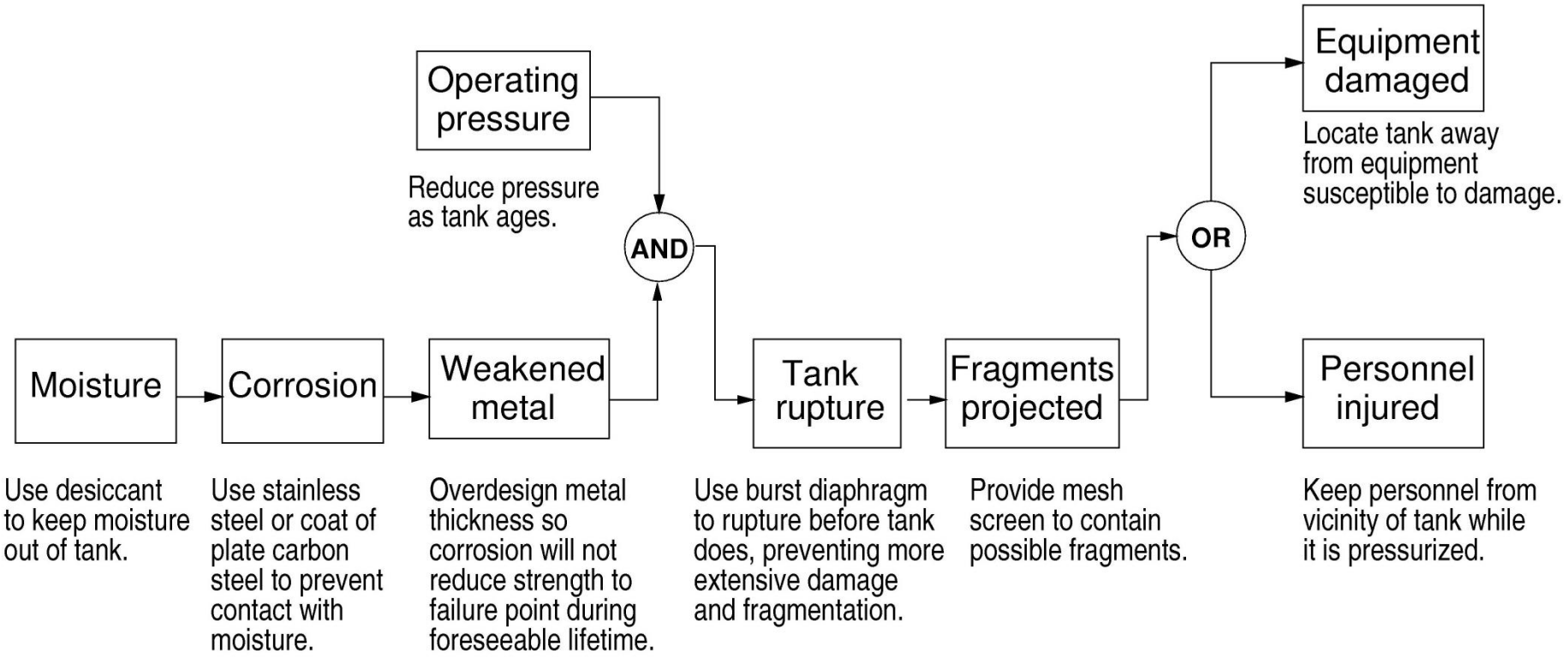
© Copyright Nancy Leveson, June 2011

Chain-of-Events Example



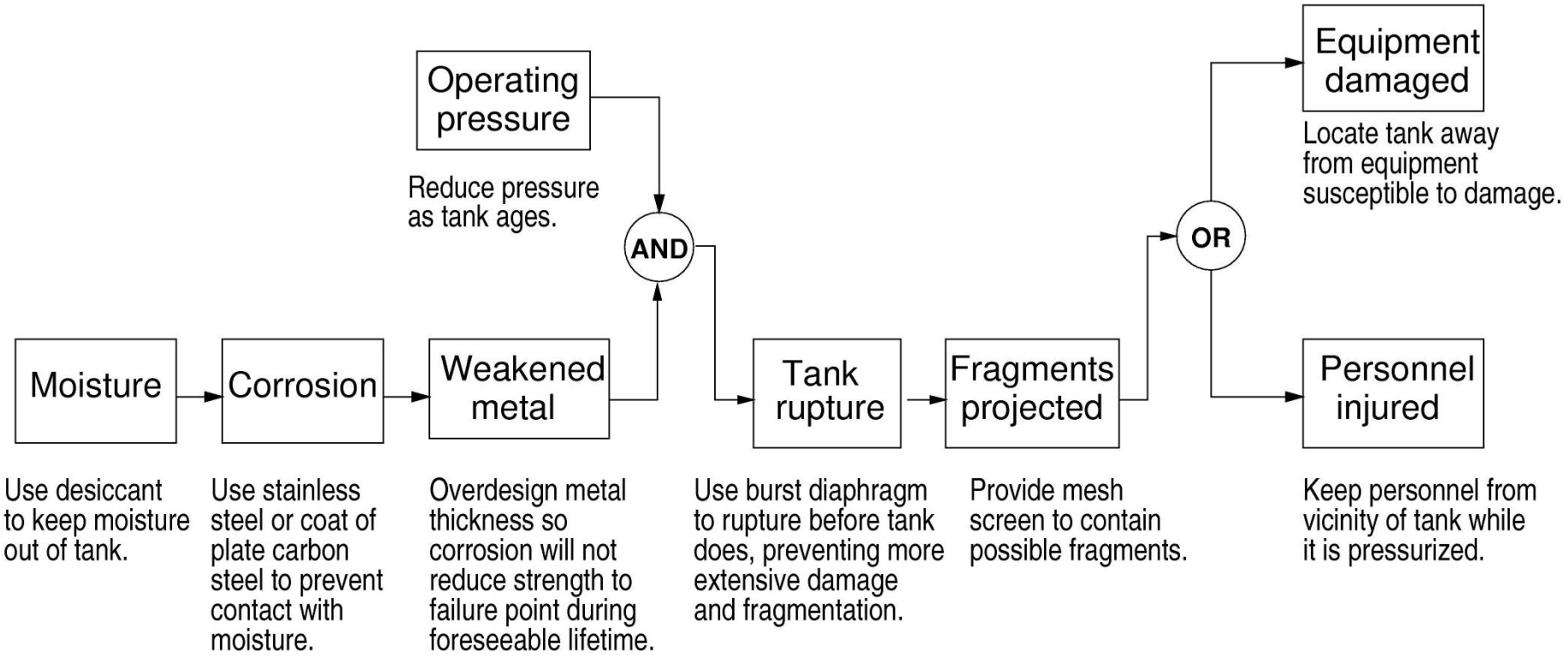
© Copyright Nancy Leveson, June 2011

Chain-of-events example



© Copyright Nancy Leveson, June 2011

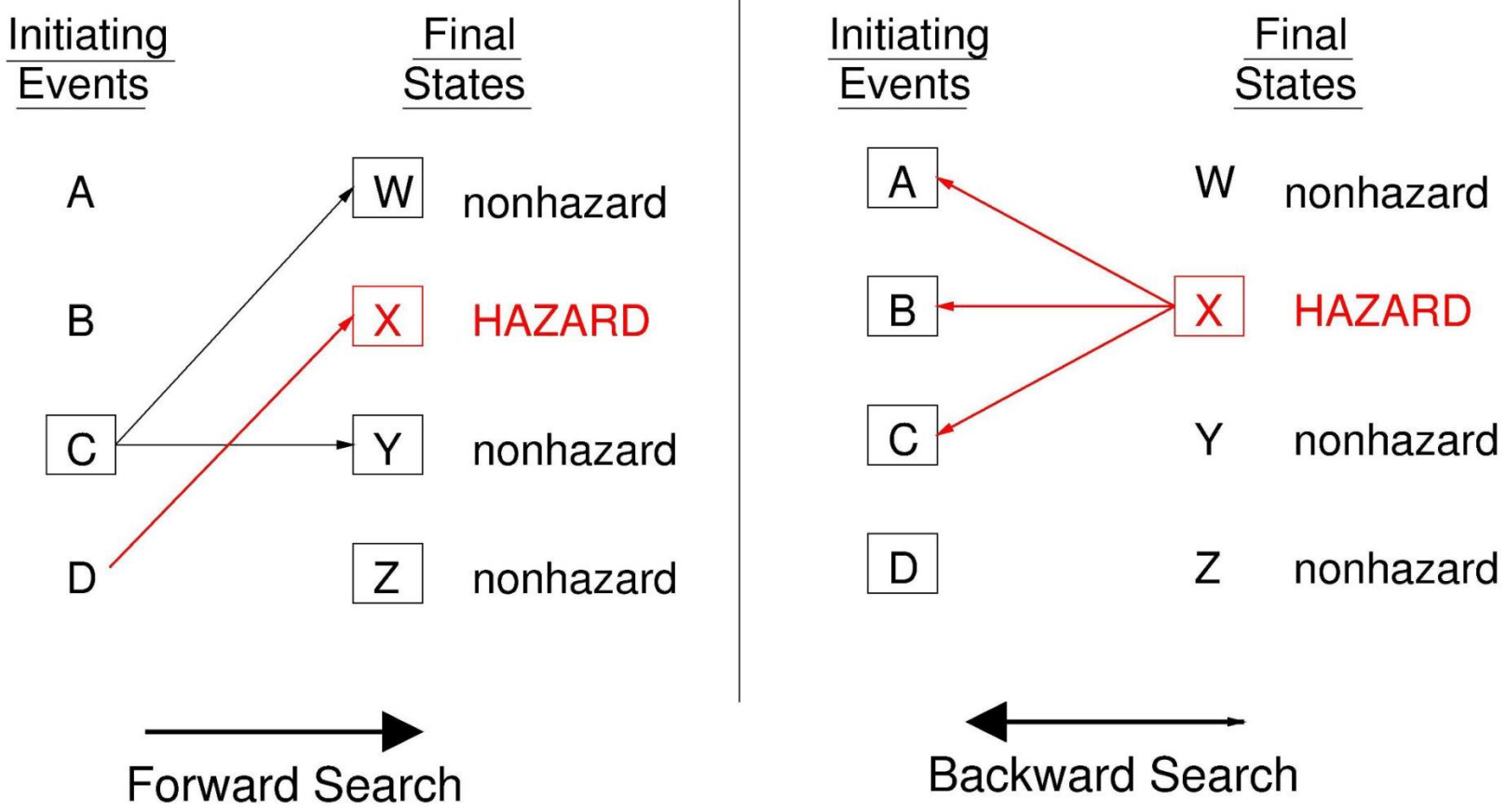
Chain-of-events example



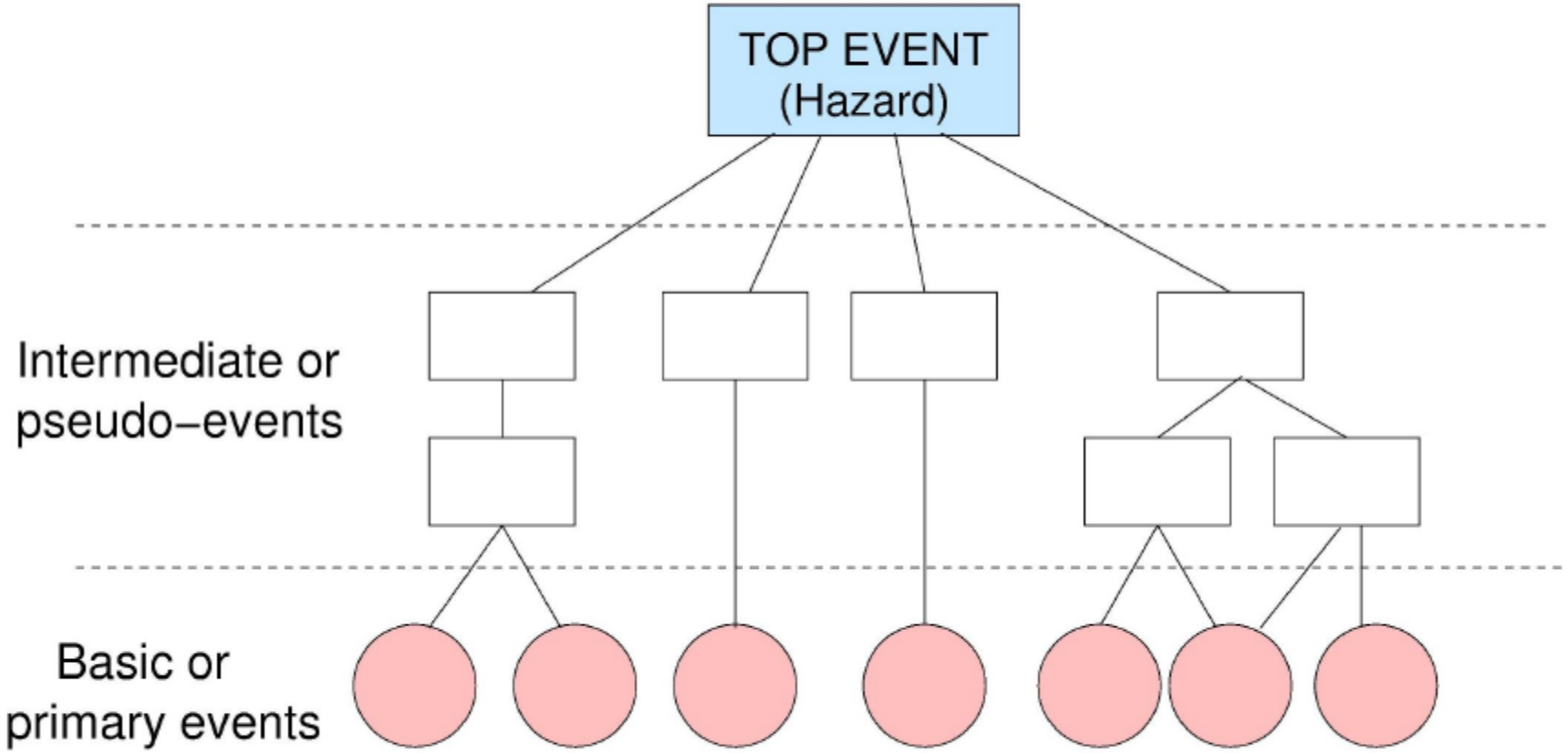
How are the event chains identified?

© Copyright Nancy Leveson, June 2011

Forward vs. Backward Search

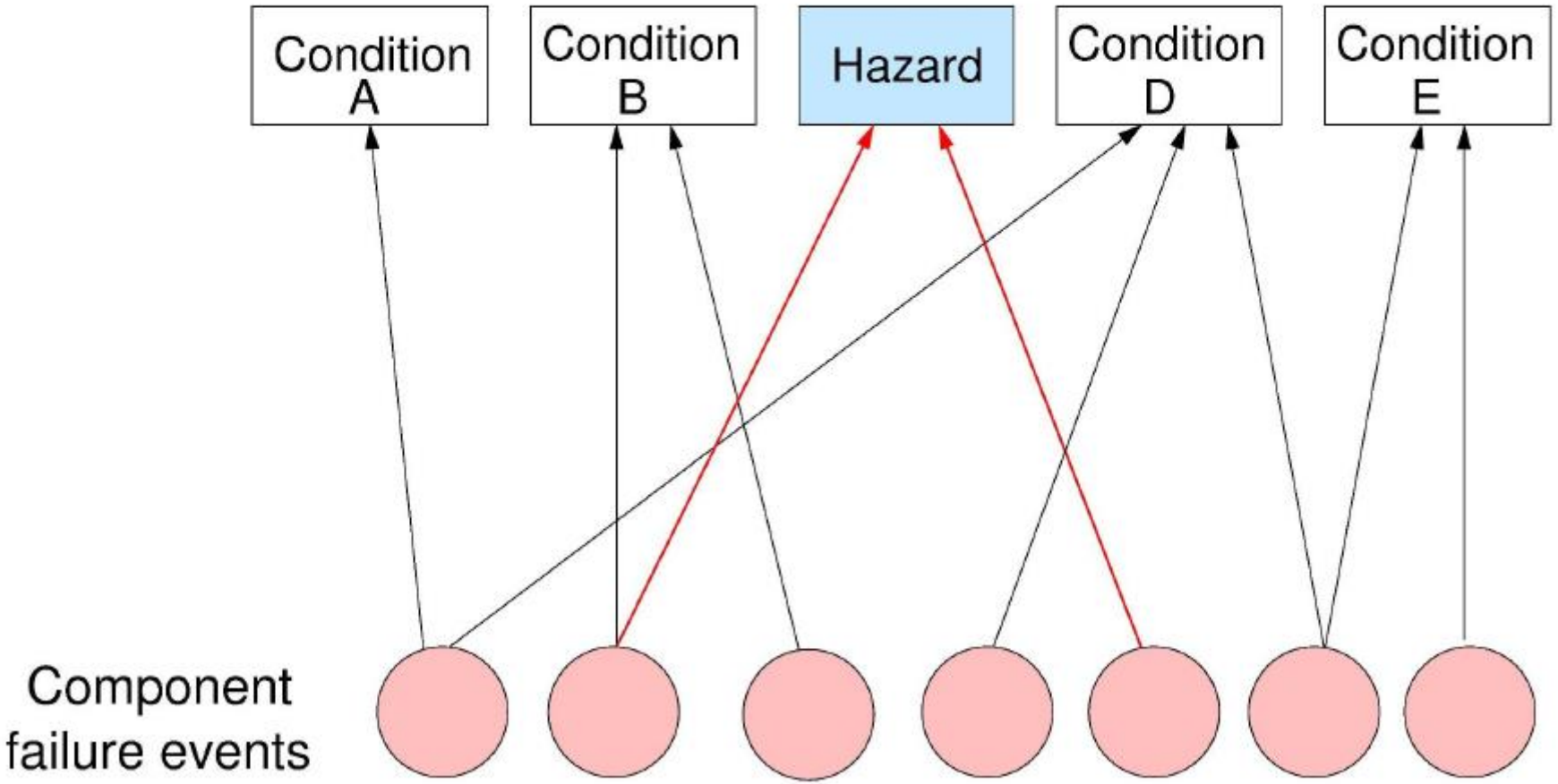


Top-Down Search



© Copyright Nancy Leveson, June 2011

Bottom-Up Search



© Copyright Nancy Leveson, June 2011

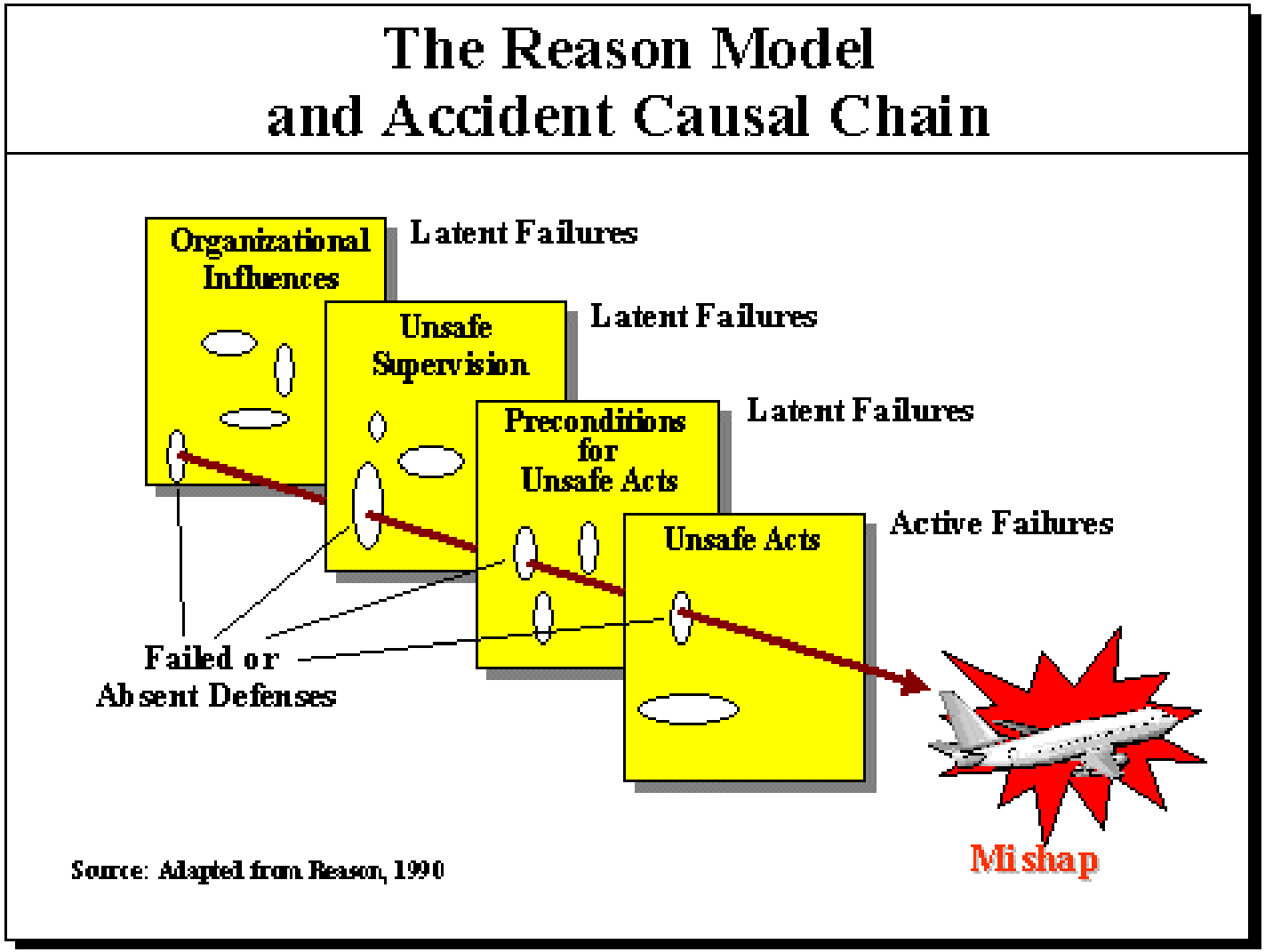
Limitations

- Component failure accidents only
  - Not accidents arising from interactions among non-failed components, e.g., system design flaws
- Single component failures only
  - What about non-events (systemic factors)? safety culture?, conditions that influence behavior, changes over time …
- Requires detailed system design (limits early analysis)
- Works best on hardware/mechanical components
  - Not software, human operators, organizational factors

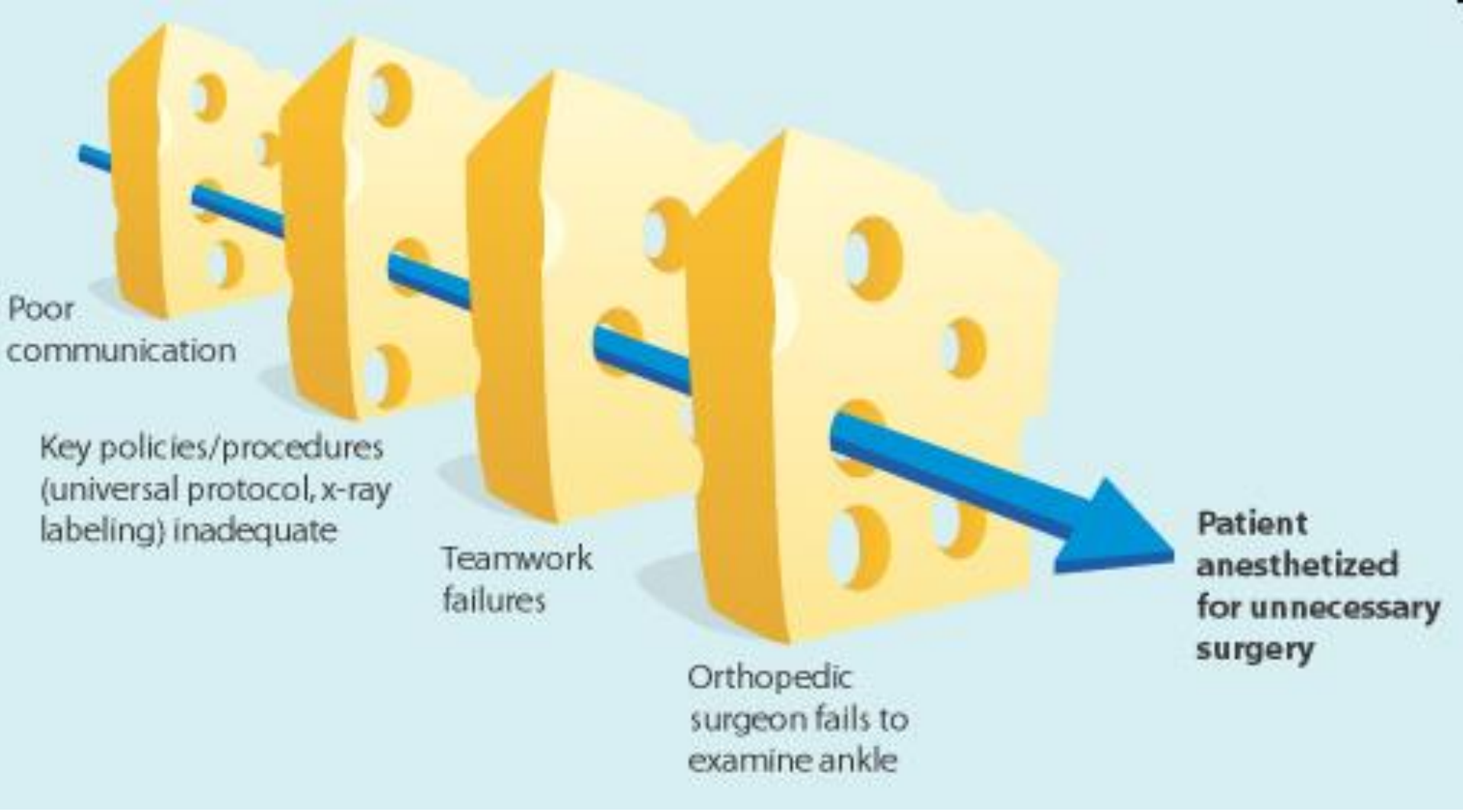
Limitations

- Inefficient, analyzes important + unimportant
  - Can result in thousands of pages of worksheets
- Tends to encourage redundancy as a solution (which may not be very effective and may be very costly)
- Failure modes must already be known
  - Best for standard parts with few and well-known failure modes

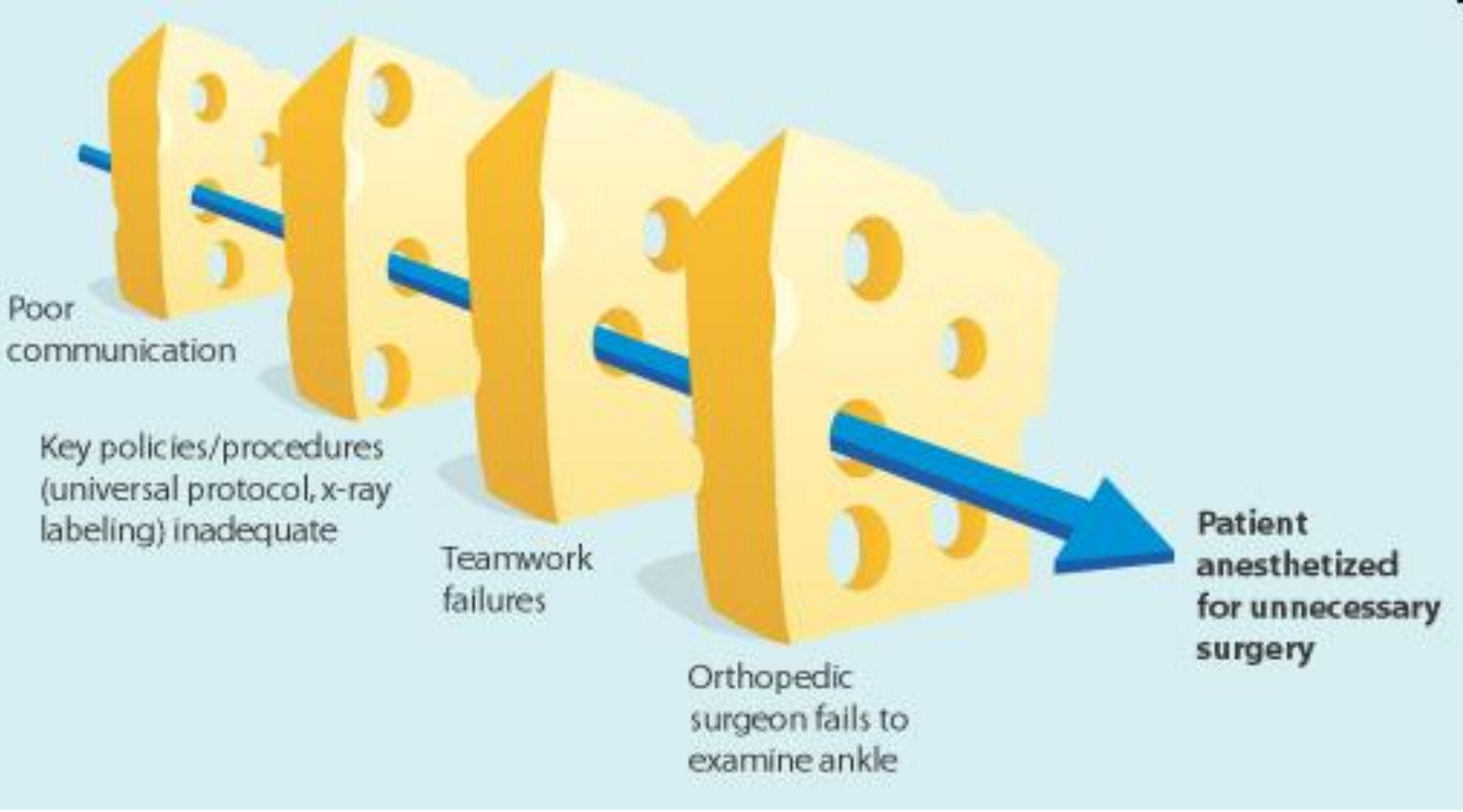
Reason Swiss Cheese = Domino Model



© Copyright Nancy Leveson, June 2011

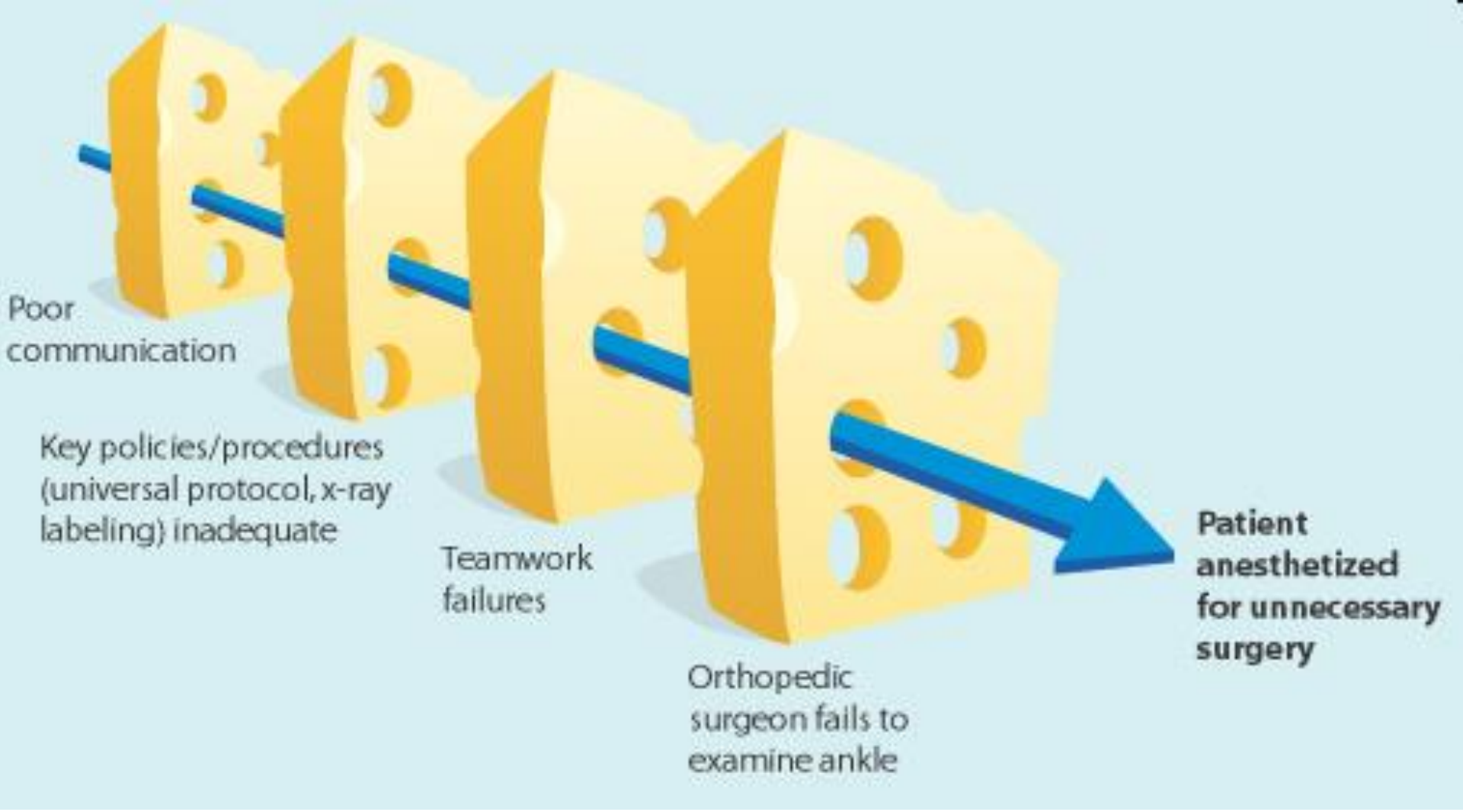


© Copyright Nancy Leveson, June 2011



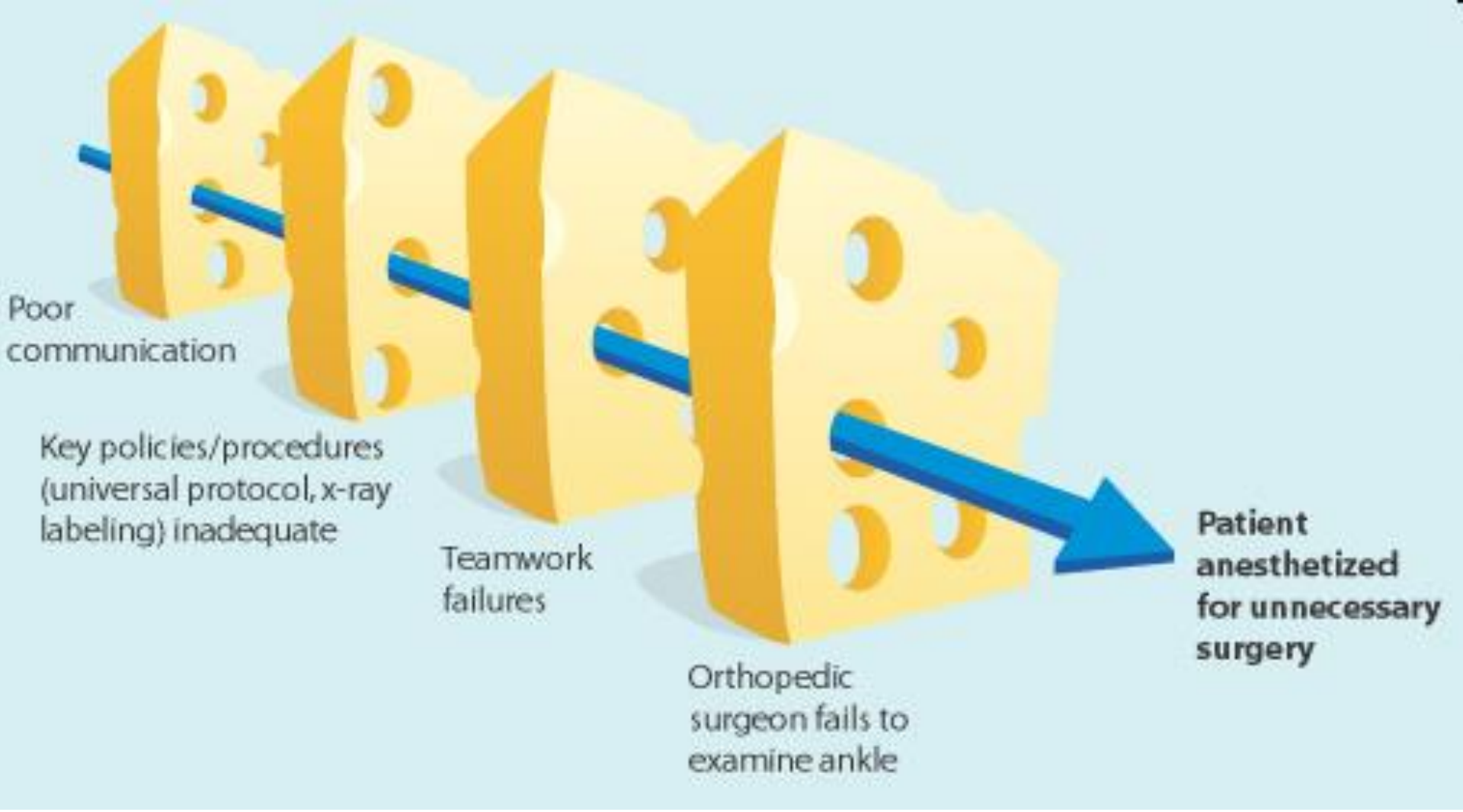
Ignores common cause failures of defenses (systemic accident factors)

© Copyright Nancy Leveson, June 2011



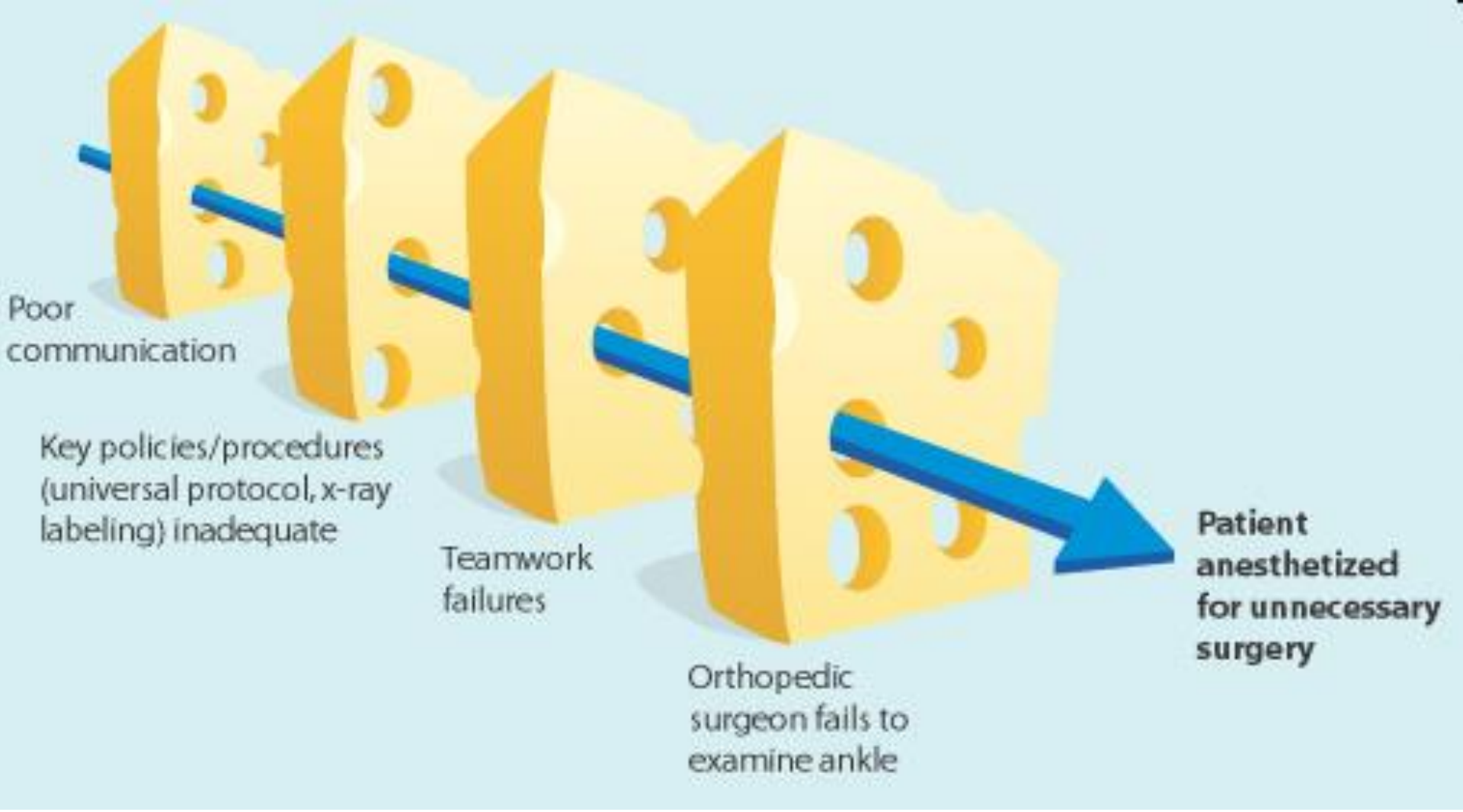
Does not include migration to states of higher risk

© Copyright Nancy Leveson, June 2011



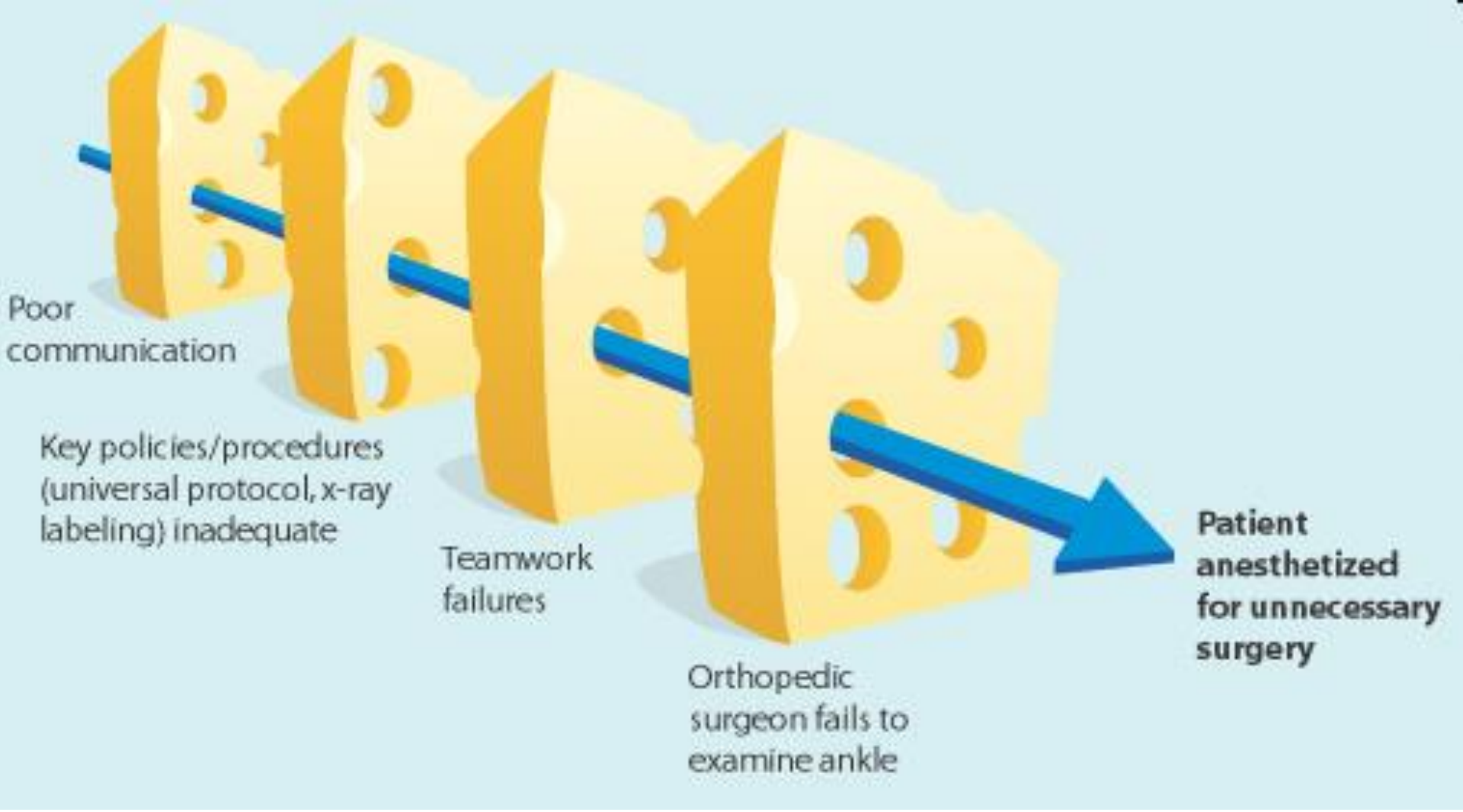
Assumes accidents are random events coming together accidentally

© Copyright Nancy Leveson, June 2011



Assumes some (linear) causality or precedence in the cheese slices (and holes)

© Copyright Nancy Leveson, June 2011



Just a chain of events, no explanation of "why" events occurred

© Copyright Nancy Leveson, June 2011

Our current tools are all 40-65 years old but our technology is very different today

1940 1950 1960 1970 1980 1990 2000 2010 2020

FMEA

FTA HAZOP

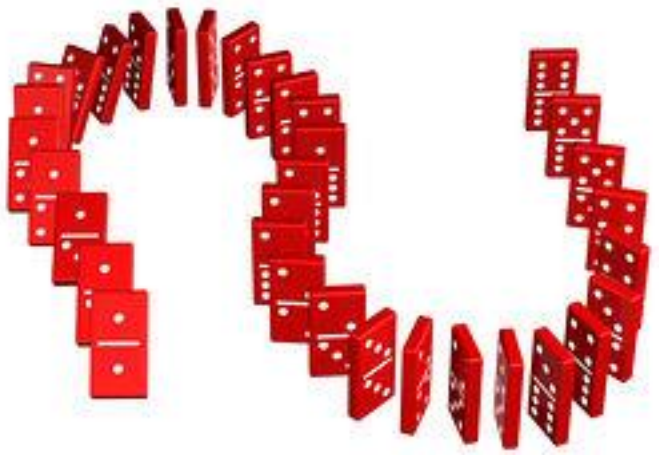
ETA

Bow Tie (CCA) FTA + ETA

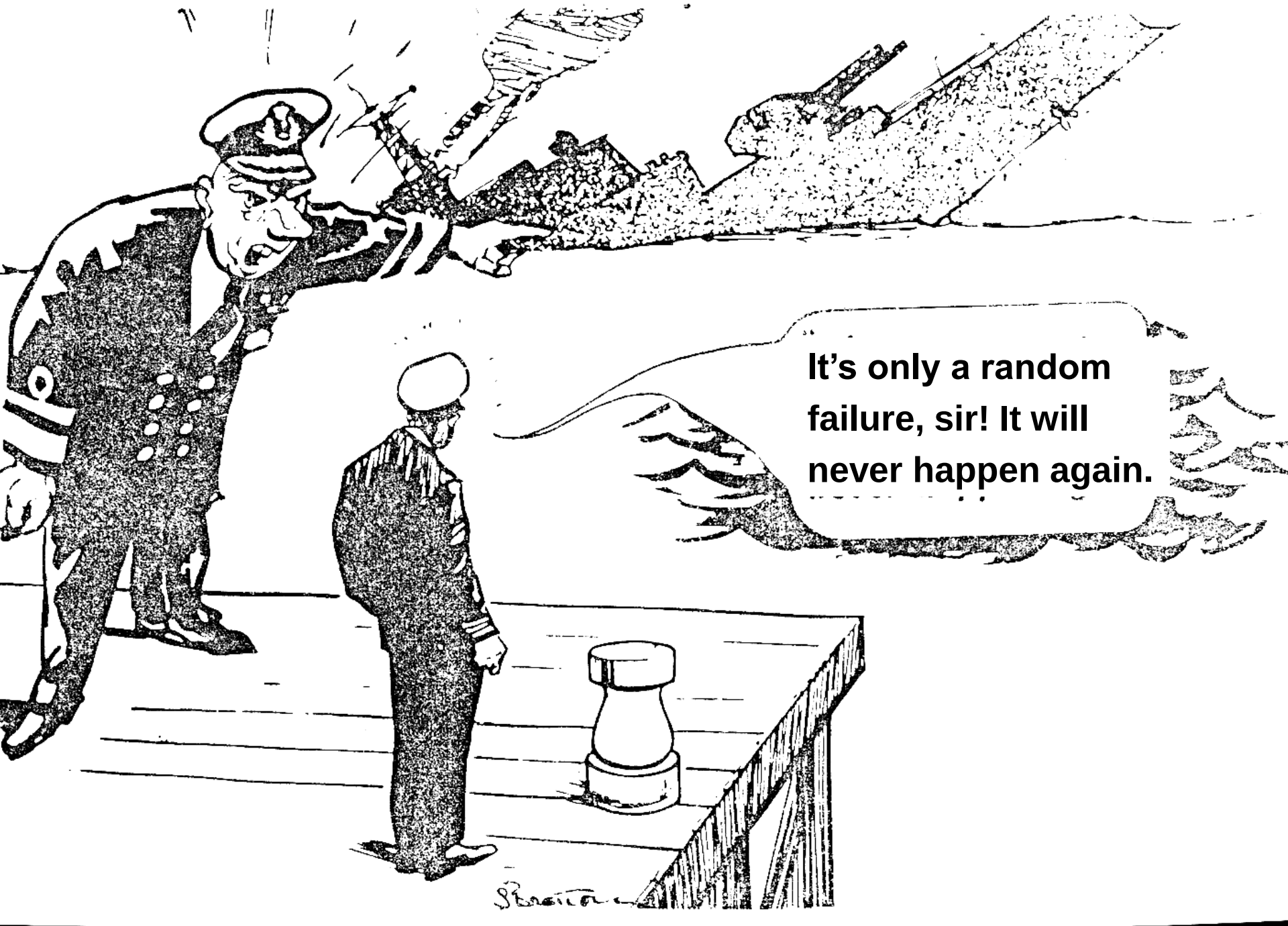
- Introduction of computer control
- Exponential increases in complexity
- New technology
- Changes in human roles

Assumes accidents caused by component failures

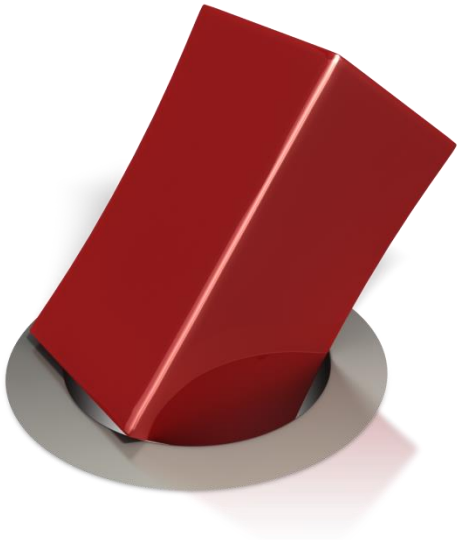
Traditional Safety Analysis Methods



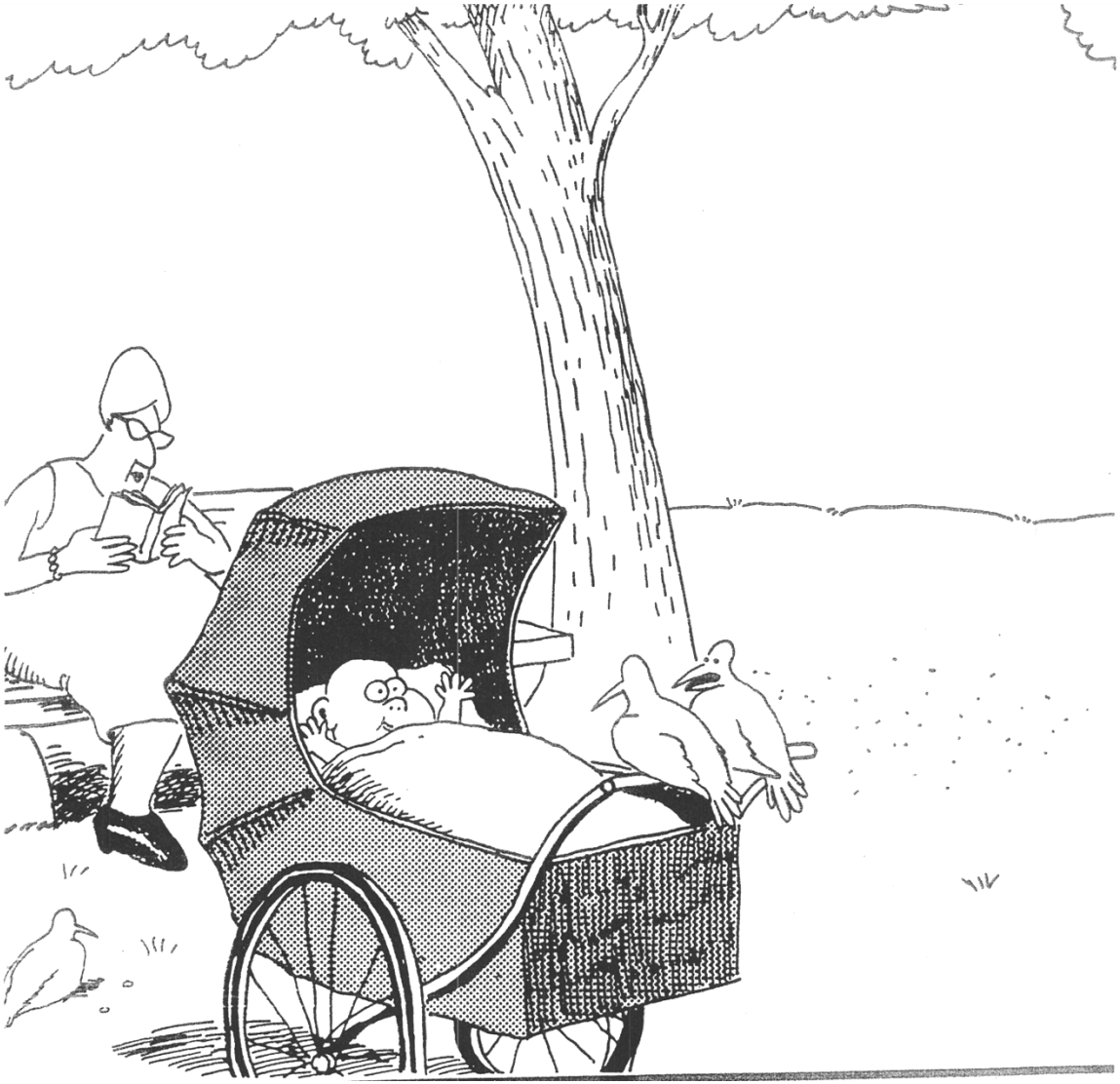
1. Assume accidents caused by chain of failure events
2. Identify the potential accident chains
3. Try to prevent the identified scenarios (chains)
    a. Establish barriers between events or
    b. Prevent component failures


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

.Trying to shoehorn new technology and new levels of complexity into old methods will not work. We need something new.

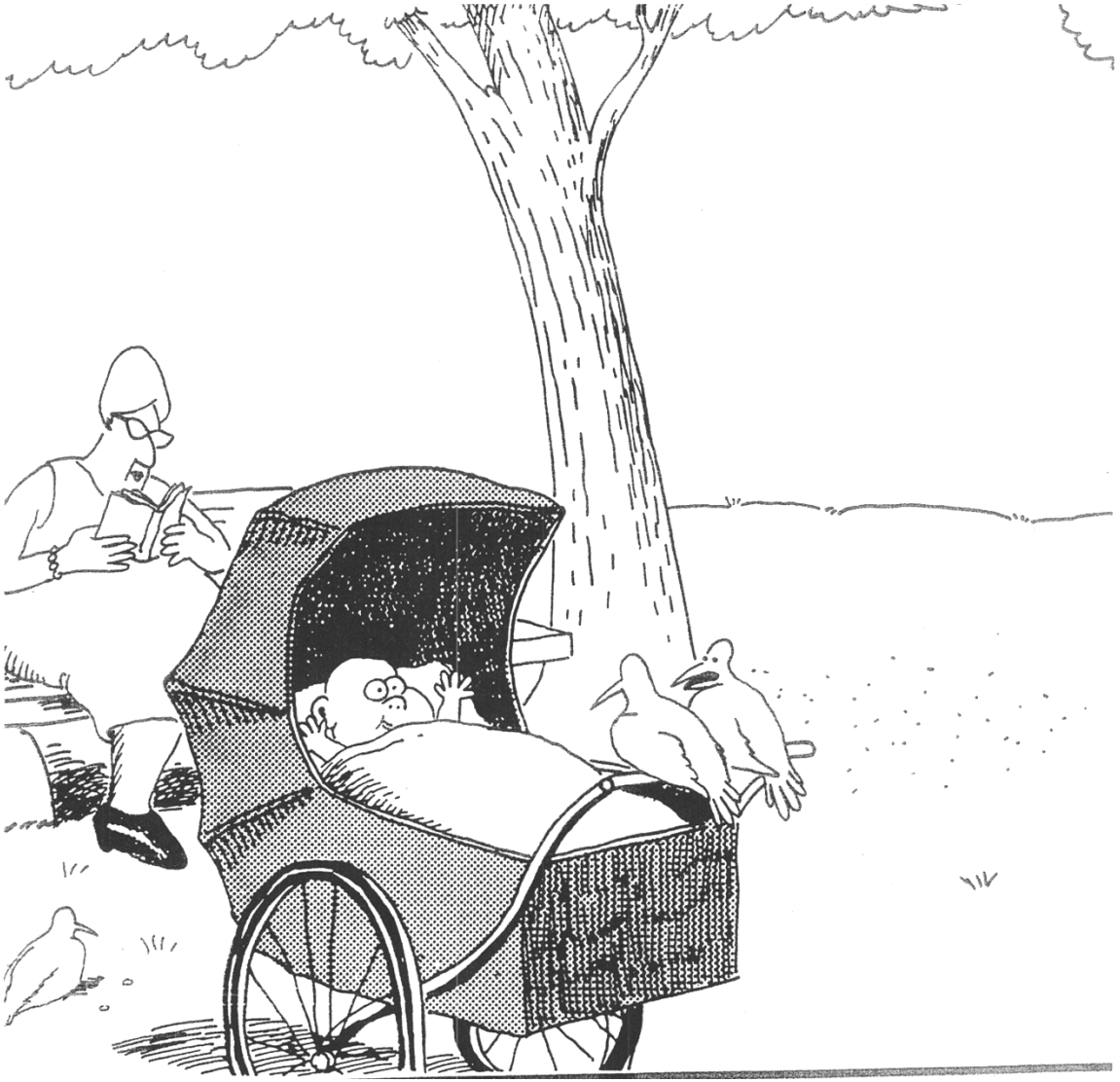


© Copyright Nancy Leveson, June 2011



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

© Copyright Nancy Leveson, June 2011



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

We Need New Tools for the New Problems

© Copyright Nancy Leveson, June 2011

STPA: System-Theoretic Process Analysis

- A top-down, system engineering analysis technique
- Identifies safety (or X) constraints (system and component requirements)
- Identifies scenarios leading to violation of constraints (requirements); use results to design or redesign system to be safer
- Can be used on technical design and organizational design
- Supports a safety-driven design process where
  - Analysis influences and shapes early design decisions
  - Analysis iterated and refined as design evolves
- Easily integrates into system engineering and MBSE tools

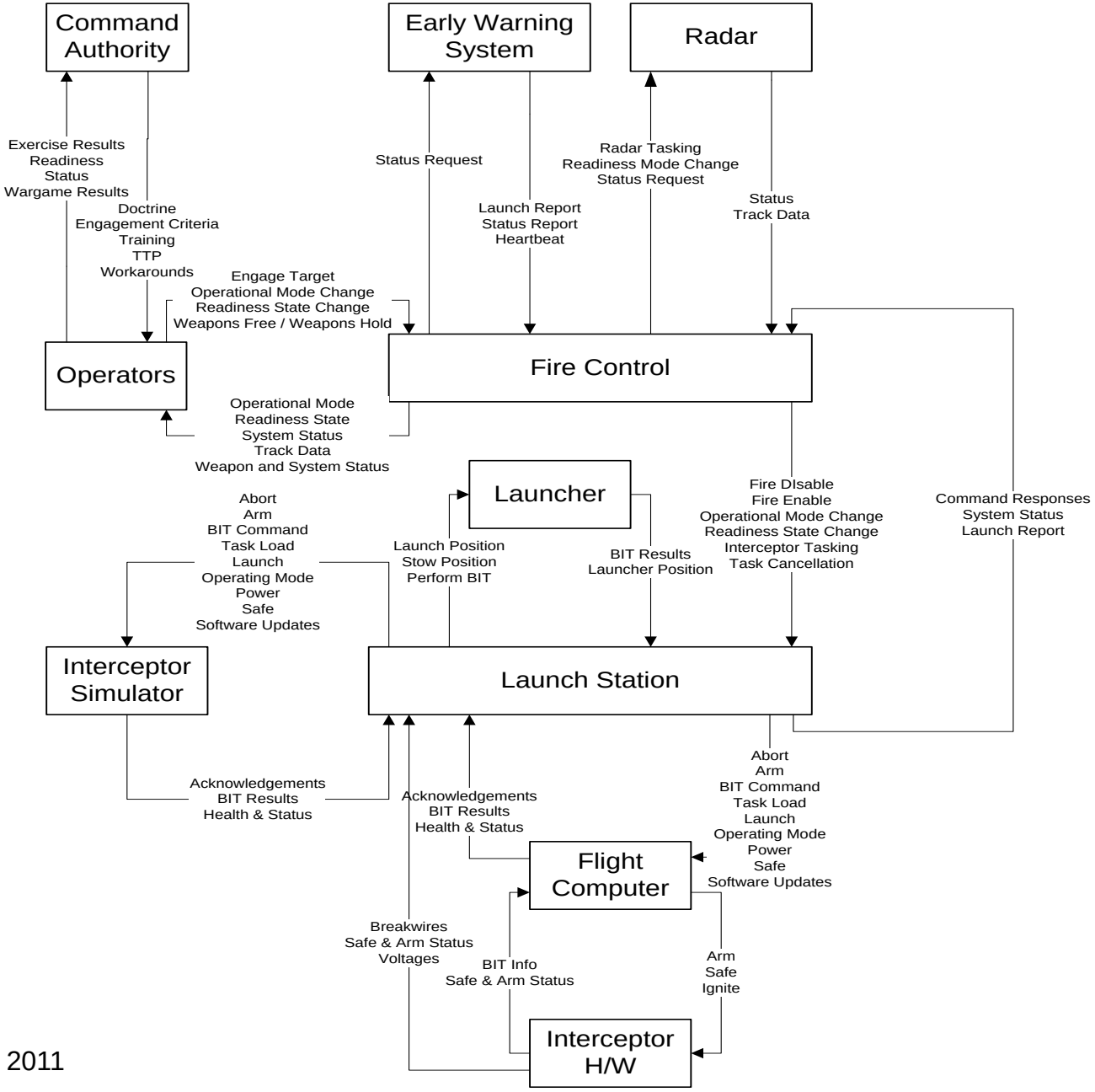
© Copyright Nancy Leveson, June 2011

Example U.S. BDMS (for MDA)

- Non-advocate safety assessment just prior to deployment and field testing
- Hazard was inadvertent launch
- Analysis done by two people over 5 months
- Deployment and testing held up for 6 months because so many scenarios identified for inadvertent launch. In many of these scenarios:
  - All components were operating exactly as intended
  - Complexity of component interactions led to unanticipated system behavior
- STPA also identified component failures that could cause inadequate control (most analysis techniques consider only these failure events)

© Copyright Nancy Leveson, June 2011

Safety Control Structure for FMIS



© Copyright Nancy Leveson, June 2011

Example Hazard Scenarios Found

- Operator could do something strange but possible at same time that radars detect a potential (but not dangerous) threat
  - Could lead to software issuing an instruction to enable firing an interceptor at the non-threat
  - Problem was a missing software requirement to handle this case
- Identified timing conditions that could lead to incorrectly launching an interceptor
- Simulator data could be taken as real data

© Copyright Nancy Leveson, June 2011

CAST: Accident Analysis Technique

- Provides a framework or process to assist in understanding entire accident process and identifying systemic factors
- Get away from blame ("who") and shift focus to "why" and how to prevent in the future
- Reduces hindsight bias
- Goal is to determine
  1. **Why people behaved the way they did**
  2. **Weaknesses in the safety control structure that allowed the loss to occur**

© Copyright Nancy Leveson, June 2011

Applies to Security Too (AF Col. Bill Young)

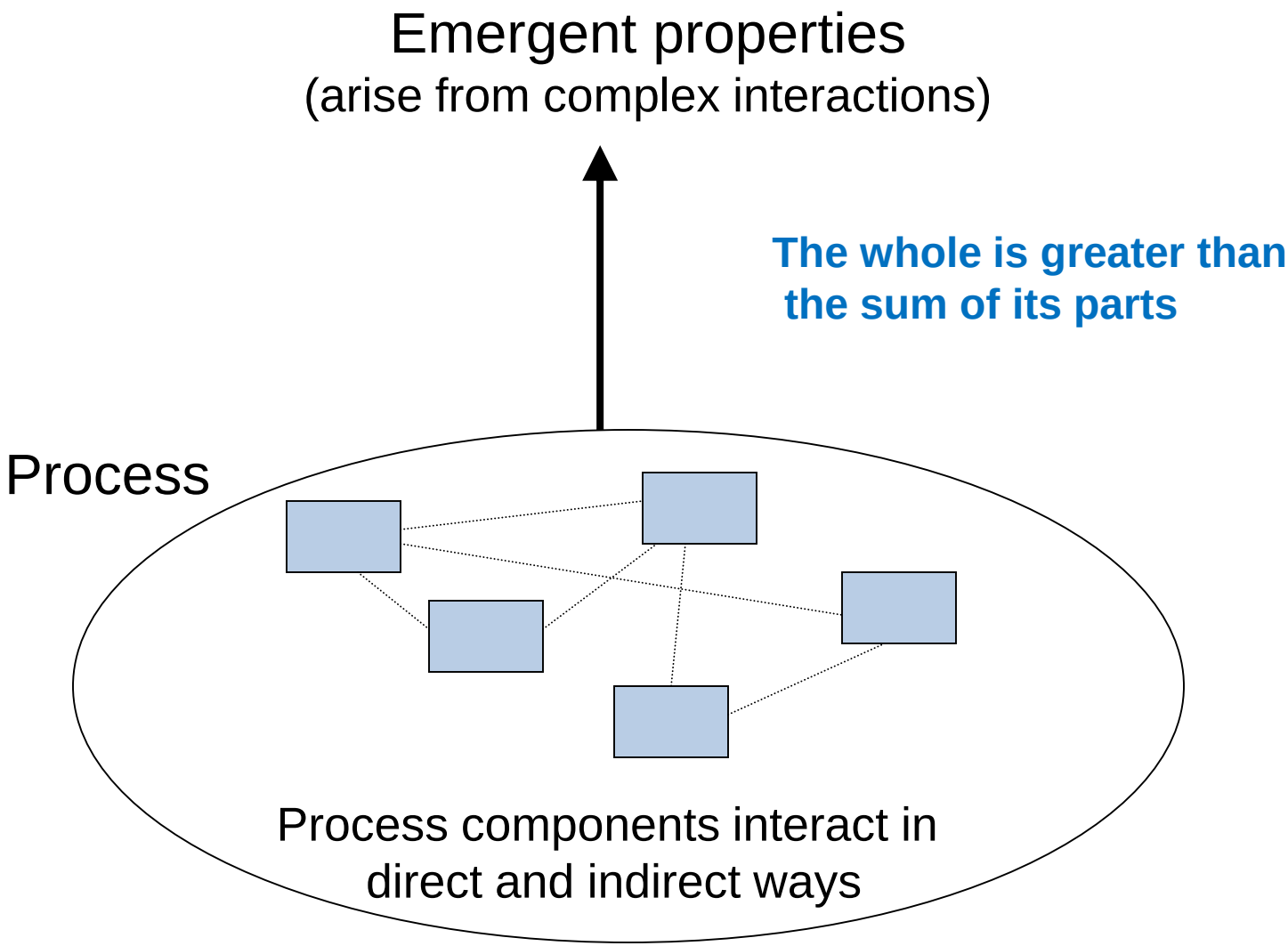
- Today currently primarily focus on tactics
  - Cyber security often framed as battle between adversaries and defenders (tactics)
  - Requires correctly identifying attackers motives, capabilities, targets
- Can reframe problem in terms of strategy
  - Identify and control system vulnerabilities (vs. reacting to potential threats)
  - Top-down strategy vs. bottom-up tactics approach
  - Tactics tackled later

© Copyright Nancy Leveson, June 2011

Integrated Approach to Safety and Security:

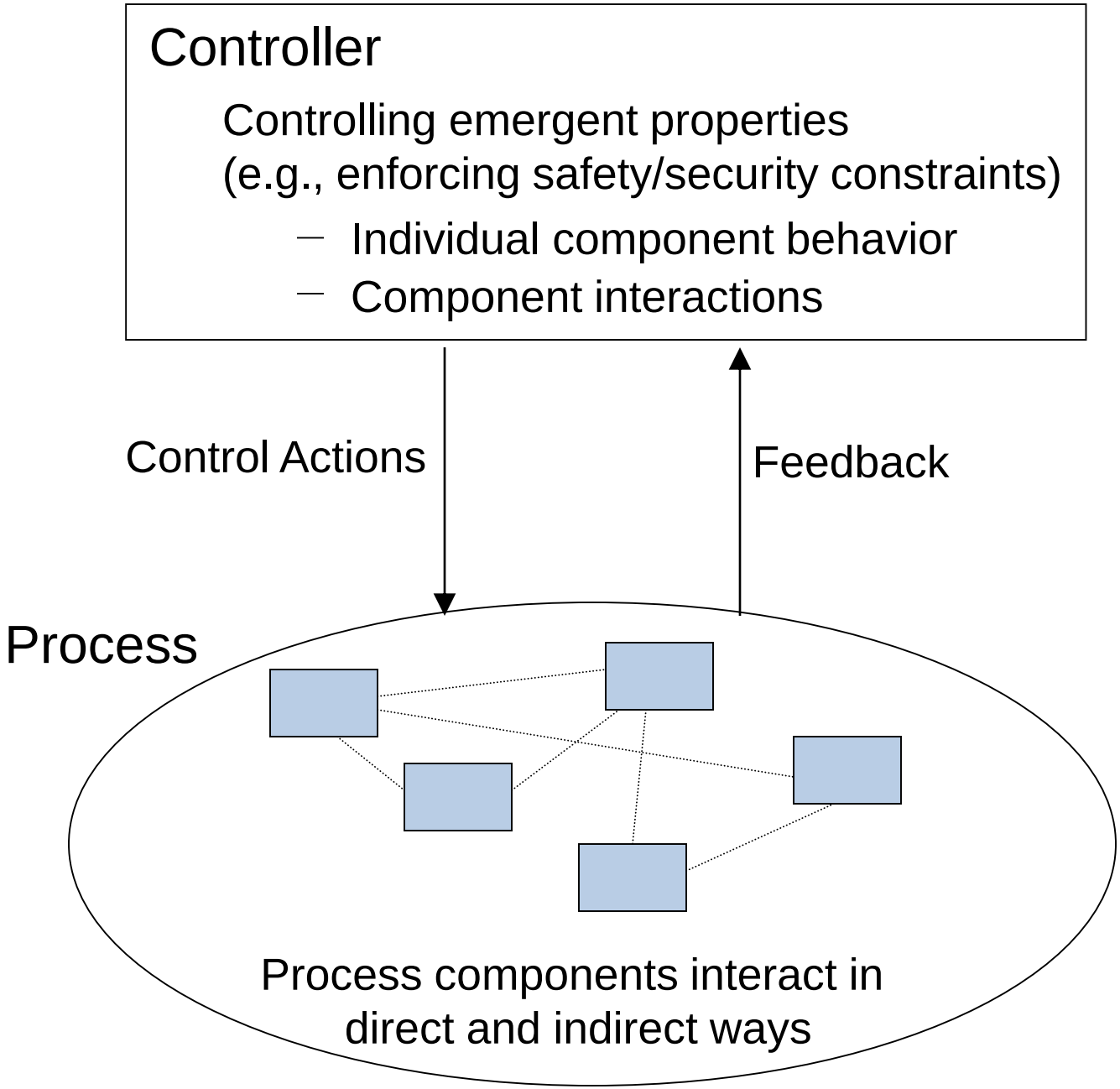
- Safety: prevent losses due to unintentional actions by benevolent actors
- Security: prevent losses due to intentional actions by malevolent actors
- Key difference is intent
- Common goal: loss prevention
  - Ensure that critical functions and services provided by networks and services are maintained
  - New paradigm for safety will work for security too
    - May have to add new causes, but rest of process is the same
  - A top-down, system engineering approach to designing safety and security into systems

© Copyright Nancy Leveson, June 2011

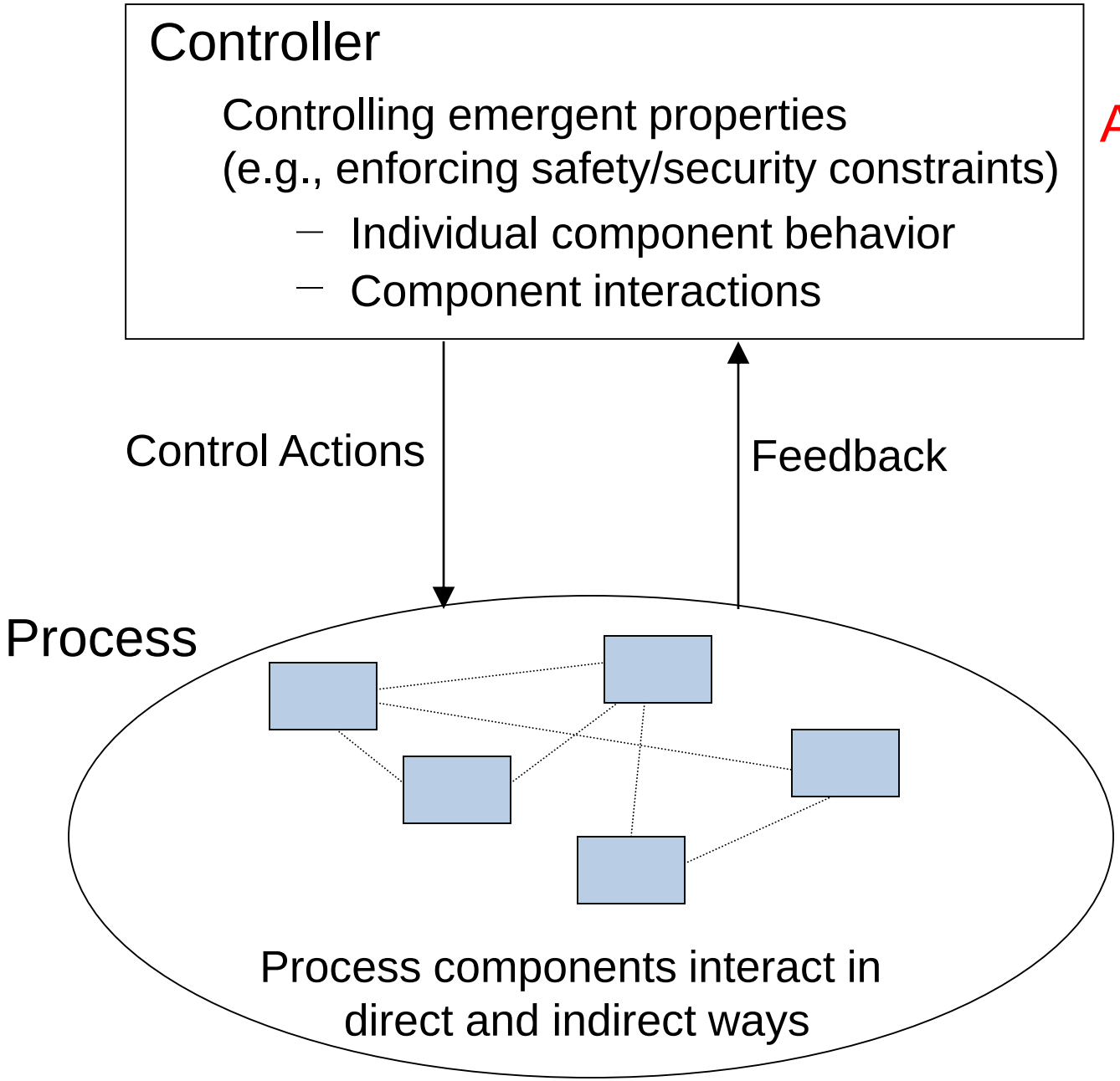


Safety and security are emergent properties

© Copyright Nancy Leveson, June 2011



© Copyright Nancy Leveson, June 2011



Air Traffic Control:
Safety
Throughput

© Copyright Nancy Leveson, June 2011

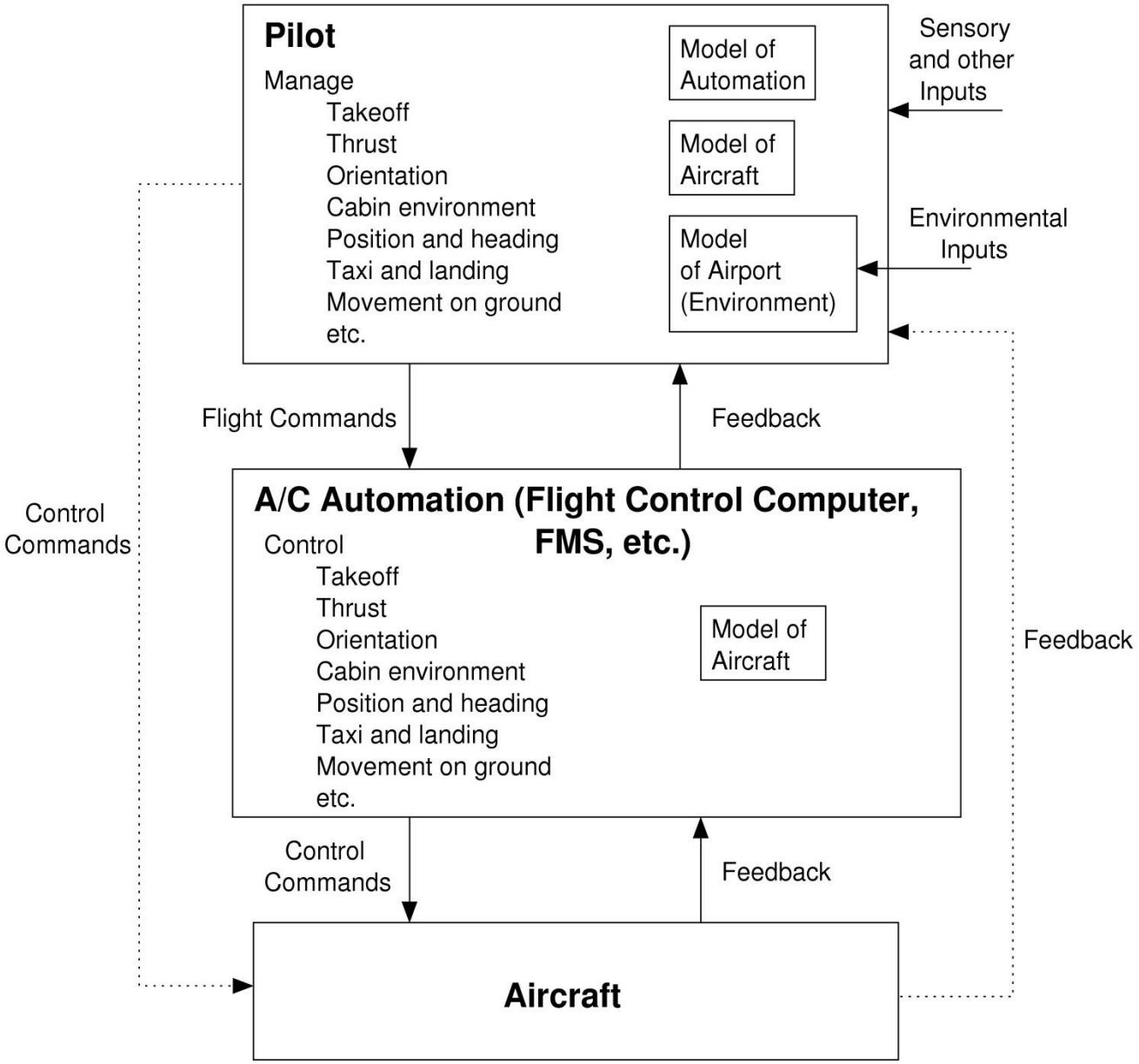
Controls/Controllers Enforce Safety Constraints

- Two aircraft/automobiles must not violate minimum separation
- Aircraft must maintain sufficient lift to remain airborne
- Level of liquid in an ISOM tower must remain below a specified level
- Toxic chemicals/radiation must not be released from plant
- Pressure in a deep water well must always be controlled
- Weapons must never be detonated inadvertently

These are the High-Level Functional Safety Requirements (What/Why) to Address During Design (How)

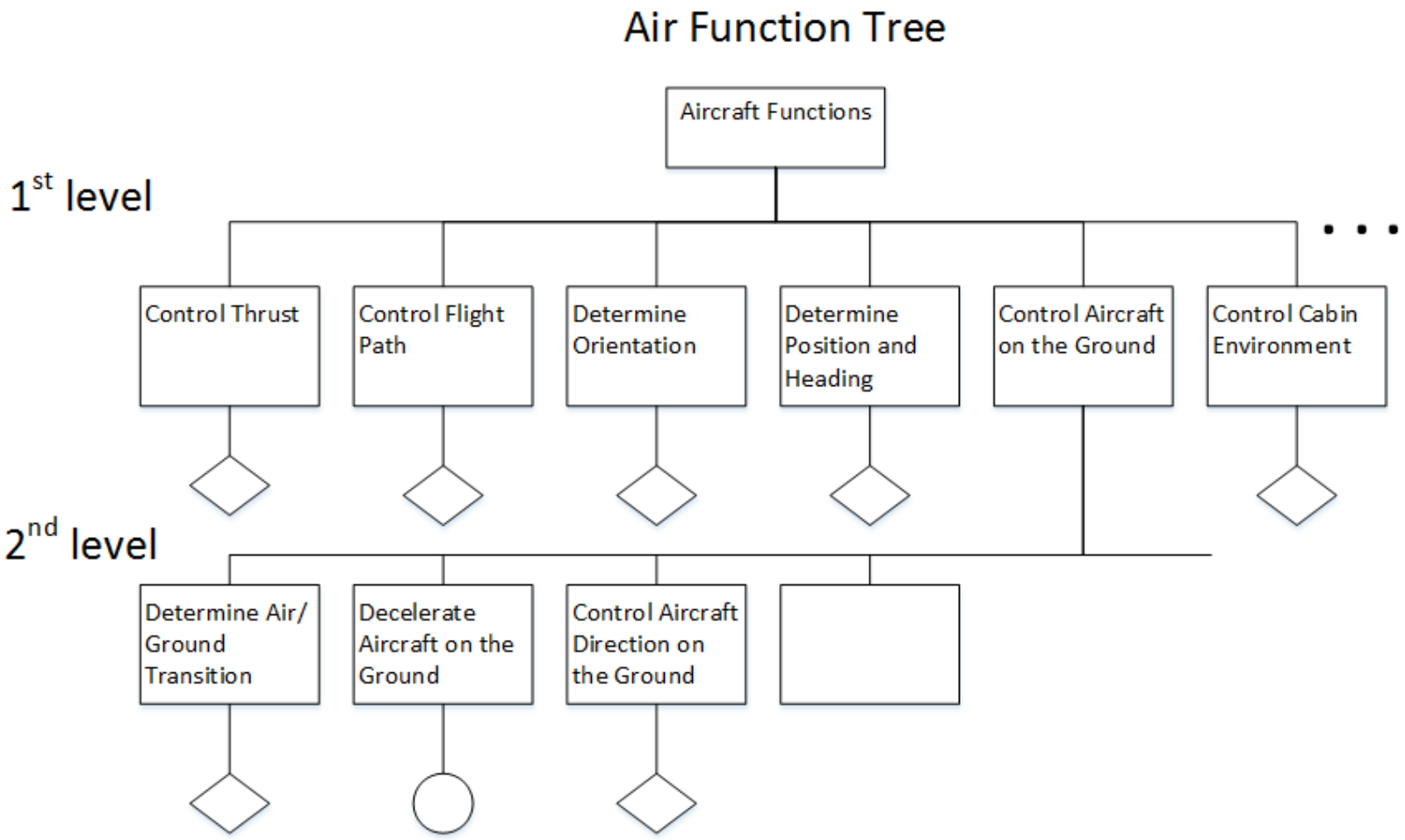
© Copyright Nancy Leveson, June 2011

Control Structure at Aircraft Level

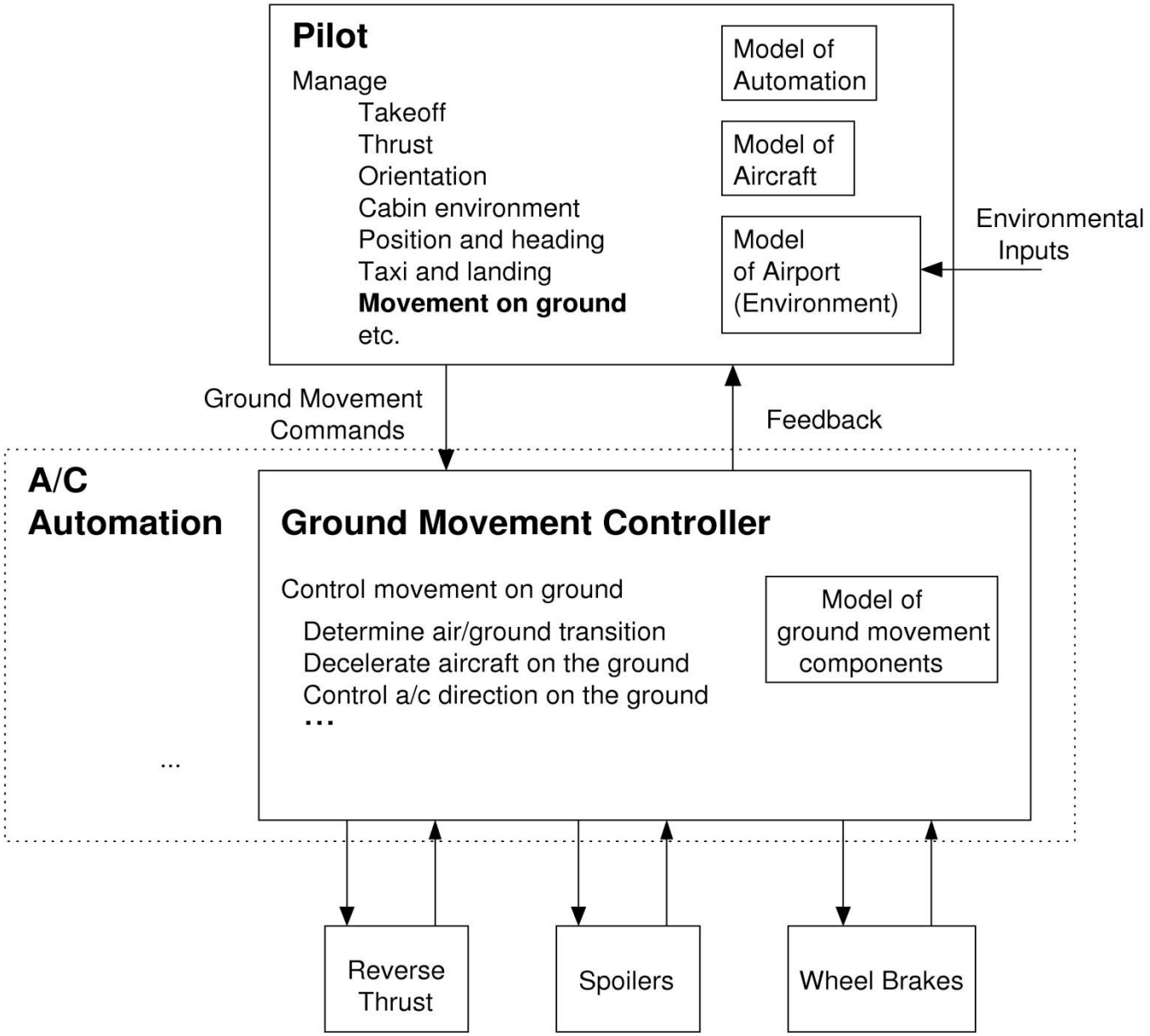


© Copyright Nancy Leveson, June 2011

Comparison with Analytic Reduction (SAE ARP 4761)

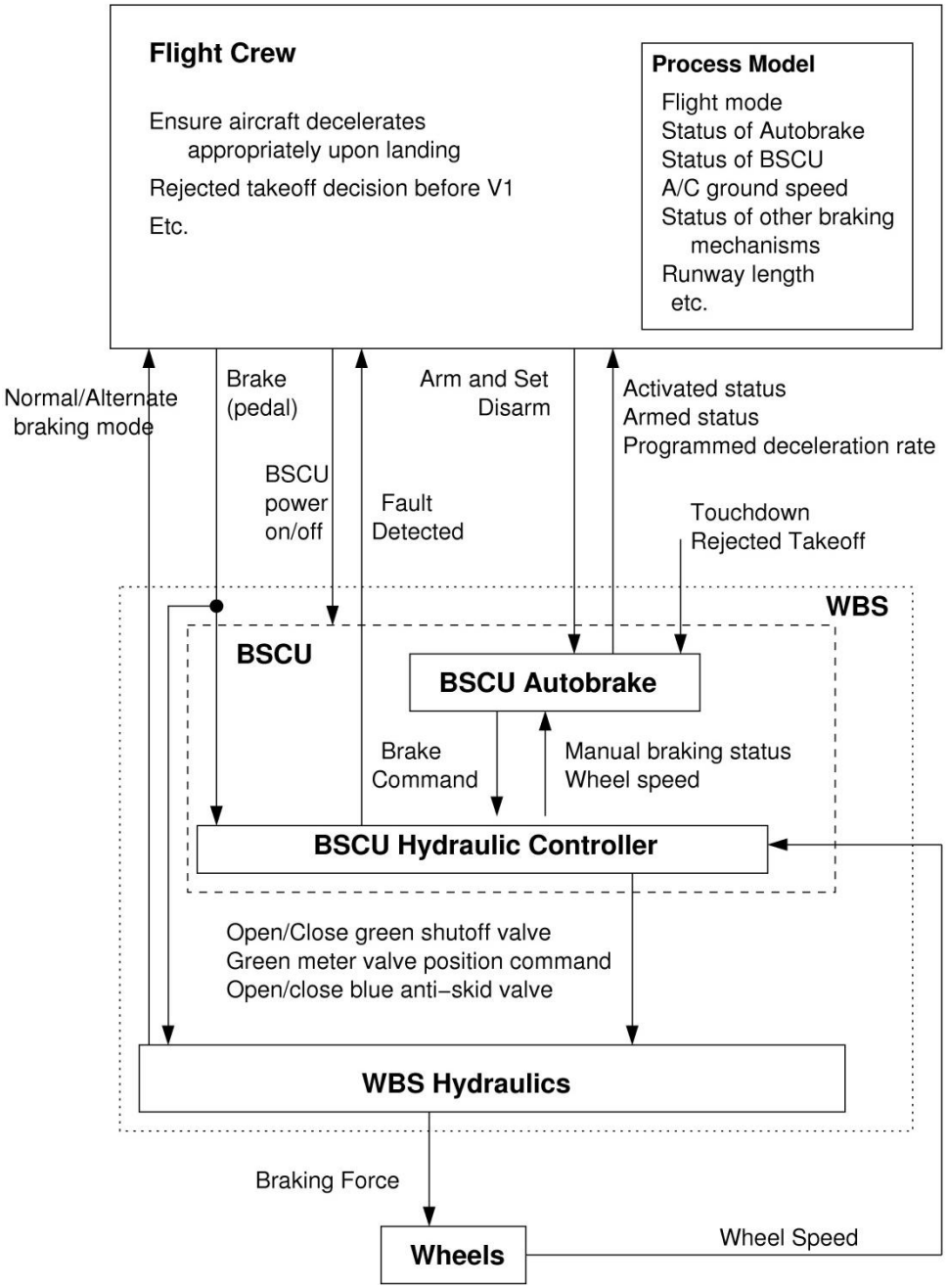


Ground System Control



© Copyright Nancy Leveson, June 2011

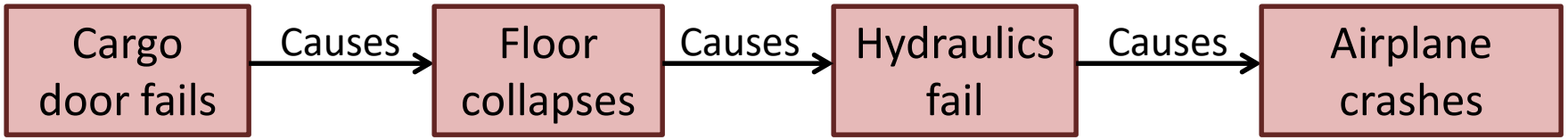
Wheel Brake System Control Structure



Copyright Nancy Leveson, June 2011

Domino "Chain of events" Model

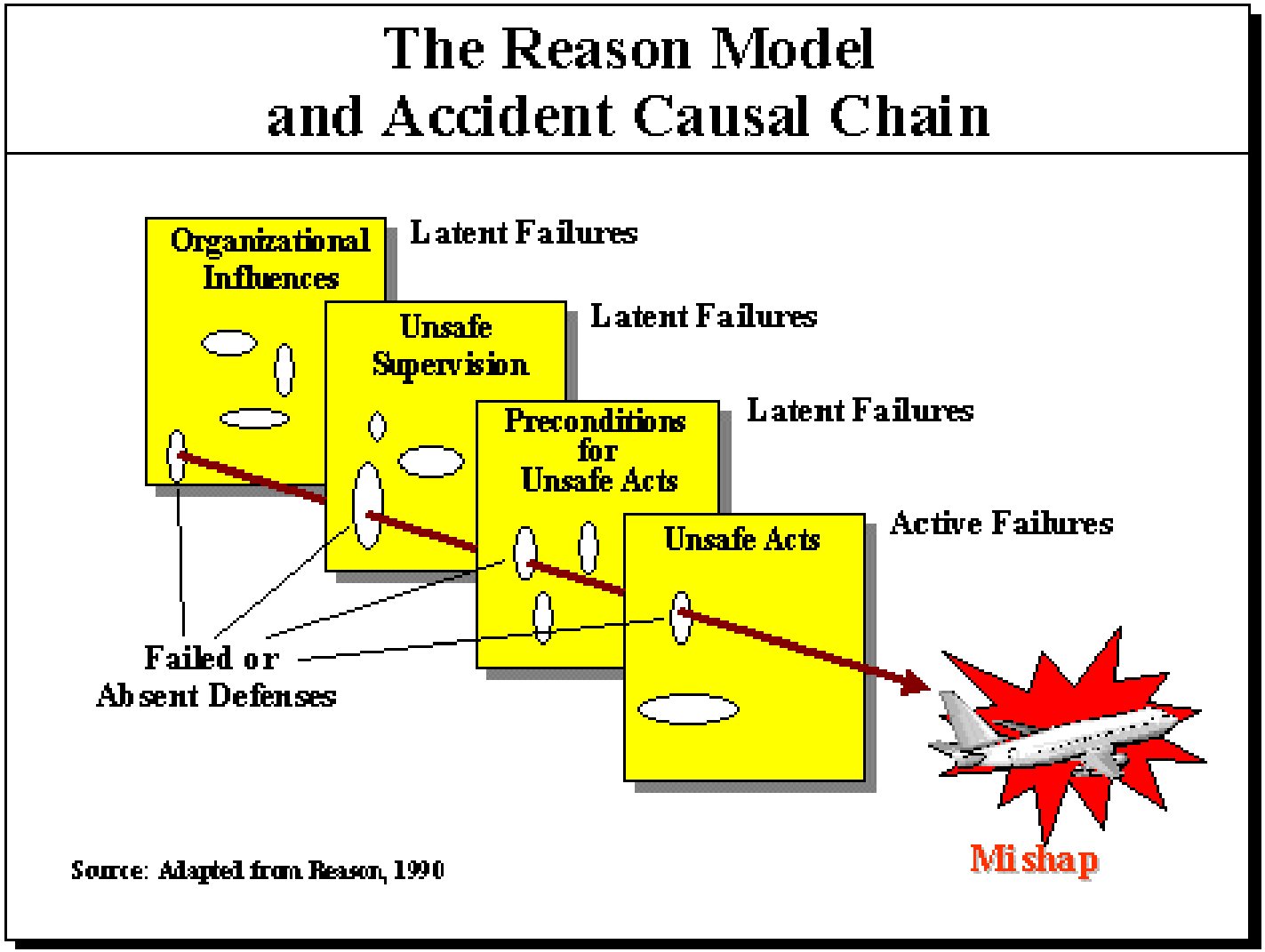
DC-10:



Chain of Failure Events

© Copyright Nancy Leveson, June 2011

Reason Swiss Cheese (1990)



© Copyright Nancy Leveson, June 2011