Engineering a Safer (and More Secure) World

Nancy Leveson

Col. Bill Young

MIT

Safety vs. 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
  - An integrated approach to safety and security is possible
  - New paradigm for safety will work for security too

© Copyright Nancy Leveson, July 2013

Traditional Approach to Safety

- Traditionally view safety as a failure problem
  - Chain of directly related failure events leads to loss
- Forms the basis for most safety engineering and reliability engineering analysis:

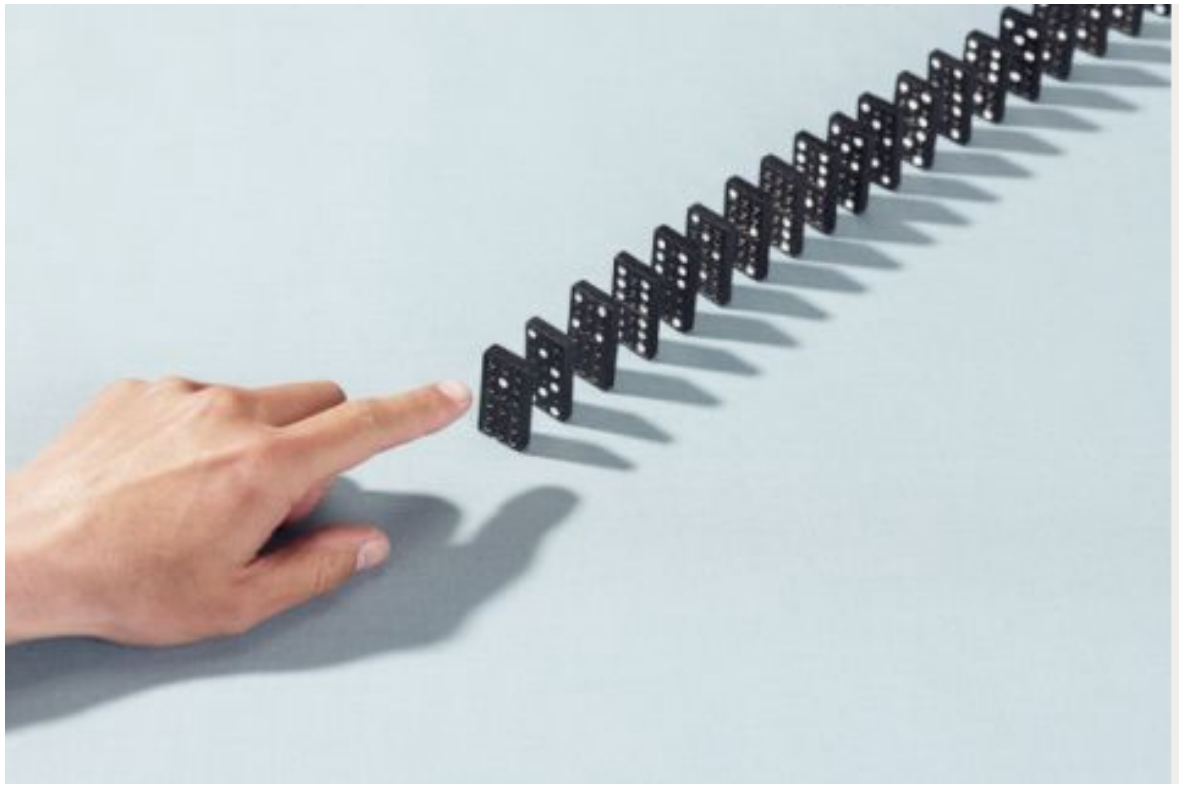
  e,g, FTA, PRA, FMECA, Event Trees, etc.

  and design (Establish barriers between events or try to prevent individual component failures:

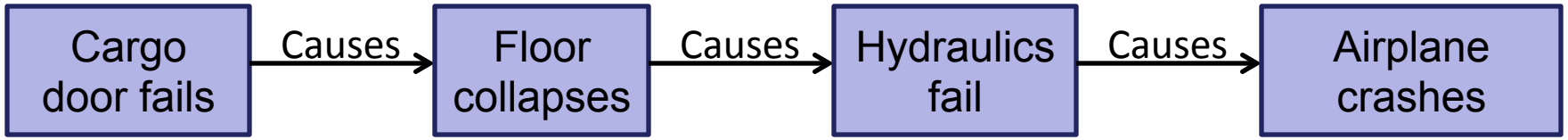
  e.g., redundancy, overdesign, safety margins, interlocks, fail-safe design, …….

© Copyright Nancy Leveson, July 2013

Domino "Chain of events" Model



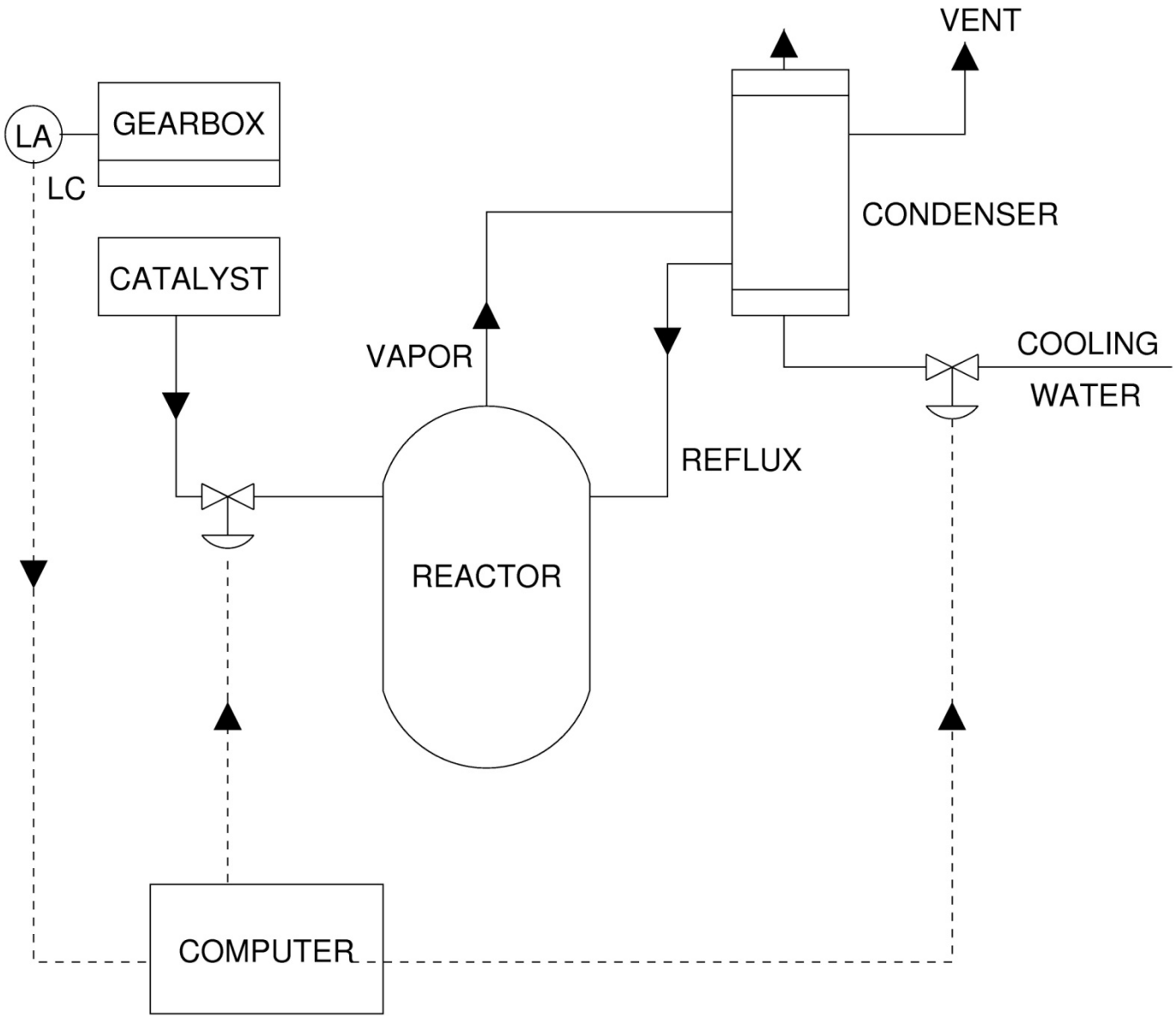
DC-10:



Failure Event-Based

© Copyright Nancy Leveson, July 2013

Accident with No Component Failures



© Copyright Nancy Leveson, July 2013

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 interactive complexity and tight coupling
  - Exacerbated by introduction of computers and software but problem is system design errors

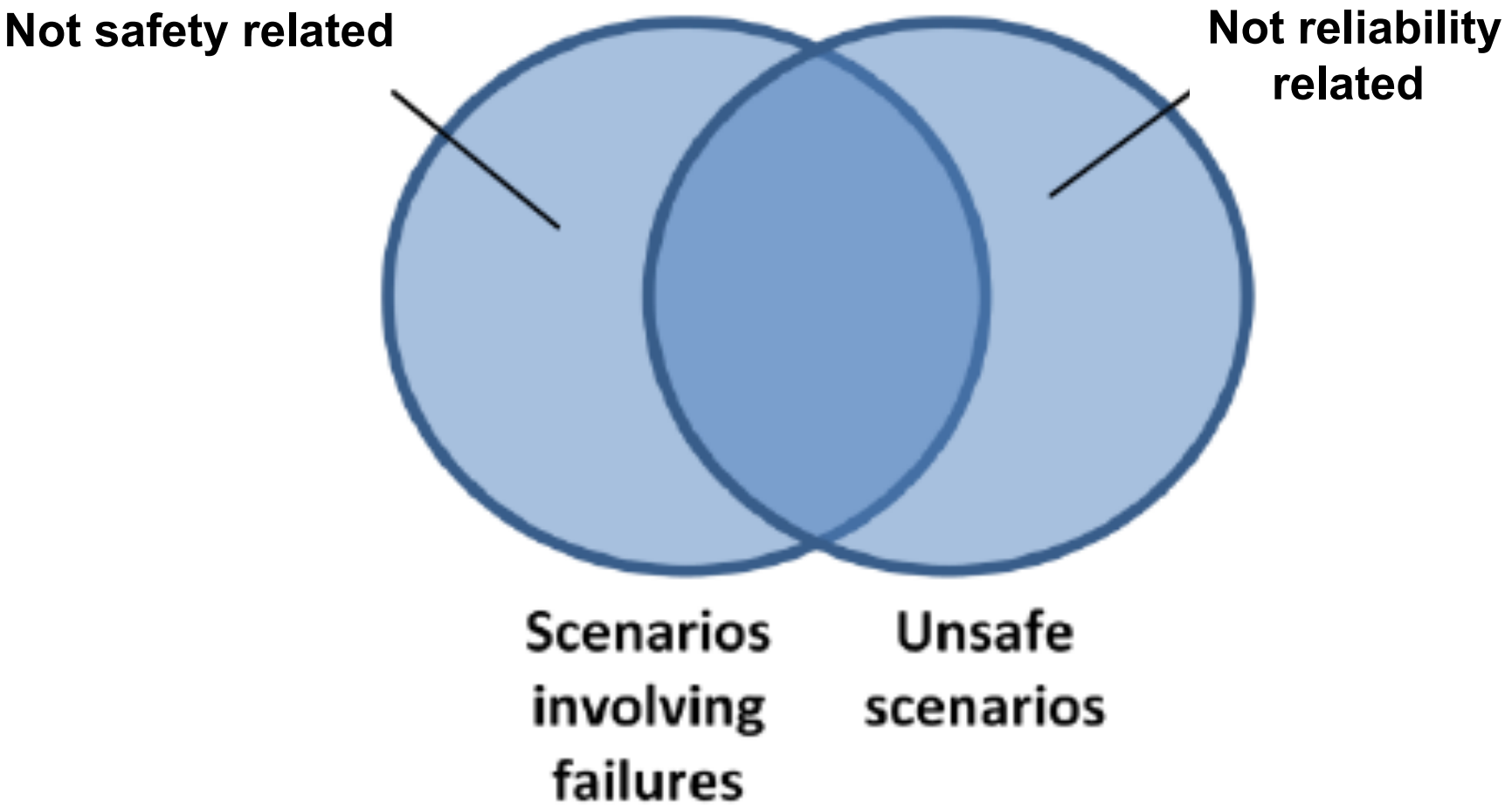
© Copyright Nancy Leveson, July 2013

Interactive Complexity

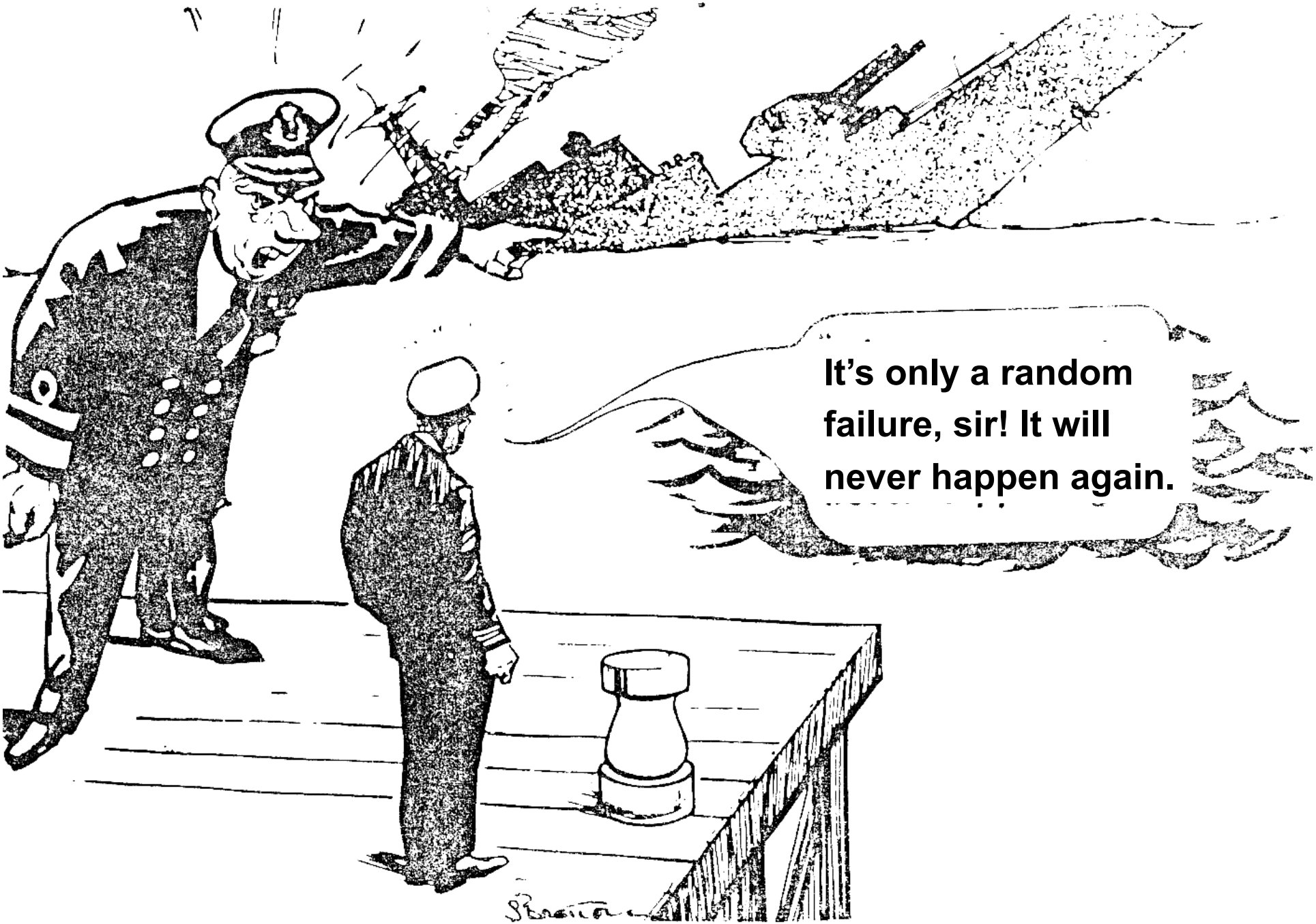
- Arises in interactions among system components
  - Software allows us to build highly coupled and interactively complex systems
  - Coupling causes interdependence
  - Increases number of interfaces and potential interactions
- Too complex to anticipate all potential interactions
- May lead to accidents even when no individual component failures

© Copyright Nancy Leveson, July 2013

Confusing Safety and Reliability



© Copyright Nancy Leveson, July 2013


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

Safety ≠ Reliability

- Safety and reliability are NOT the same
  - Sometimes increasing one can even decrease the other.
  - Making all the components highly reliable will have no impact on component interaction accidents.
- For relatively simple, electro-mechanical systems with primarily component failure accidents, reliability engineering can increase safety.
- But this is untrue for complex, software-intensive socio-technical systems.

© Copyright Nancy Leveson, July 2013

Software-Related Accidents

- Are usually caused by flawed requirements
  - Incomplete or wrong assumptions about operation of controlled system or required operation of computer
  - Unhandled controlled-system states and environmental conditions

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

© Copyright Nancy Leveson, July 2013

Software-Related Accidents (2)

- Software may be highly reliable and “correct” and still be unsafe:
  - Correctly implements requirements but specified behavior unsafe from a system perspective.
  - Requirements do not specify some particular behavior required for system safety (incomplete)
  - Software has unintended (and unsafe) behavior beyond what is specified in requirements.

© Copyright Nancy Leveson, July 2013

Limitations of Traditional Approach (1)

- Systems are becoming more complex
  - Accidents often result from interactions among components, not just component failures
  - Too complex to anticipate all potential interactions
    - By designers
    - By operators
  - Indirect and non-linear interactions

© Copyright Nancy Leveson, July 2013

Limitations of Traditional Approach (2)

- Omits or oversimplifies important factors
  - Component interaction accidents (vs. component failure accidents)
  - Indirect or non-linear interactions and complexity
  - Systemic factors in accidents
  - Human “errors”
  - System design errors (including software errors)
  - Evolution and change over time

© Copyright Nancy Leveson, July 2013

© Copyright Nancy Leveson, July 2013

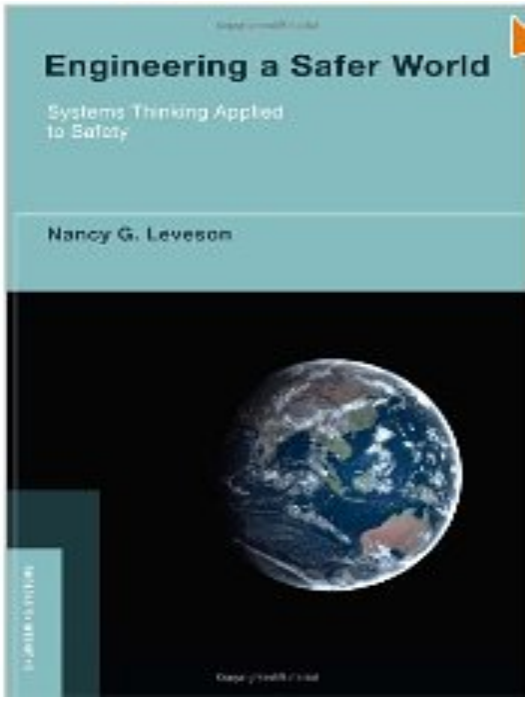
So What Do We Need to Do?
"Engineering a Safer World"

- Expand our accident causation models
- Create new, more powerful and inclusive hazard analysis techniques
- Use new system design techniques
  - Safety-driven design
  - Improved system engineering
- Improve accident analysis and learning from events
- Improve control of safety during operations
- Improve management decision-making and safety culture

© Copyright Nancy Leveson, July 2013

Nancy Leveson, *Engineering a Safer World:*

Systems Thinking Applied to Safety



MIT Press, January 2012

© Copyright Nancy Leveson, July 2013

STAMP
(System-Theoretic Accident Model and Processes)

- A new, more powerful accident causation model
- Based on systems theory, not reliability theory
- Treats accidents as a dynamic control problem (vs. a failure problem)
- Includes
  - Entire socio-technical system (not just technical part)
  - Component interaction accidents
  - Software and system design errors
  - Human errors

© Copyright Nancy Leveson, July 2013

Introduction to Systems Theory

Ways to cope with complexity

1. Analytic Reduction
2. Statistics

© Copyright Nancy Leveson, July 2013

Analytic Reduction

- Divide system into distinct parts for analysis

  Physical aspects → Separate physical components

  Behavior → Events over time

- Examine parts separately

- Assumes such separation possible:

  1. The division into parts will not distort the phenomenon
     - Each component or subsystem operates independently
     - Analysis results not distorted when consider components separately

© Copyright Nancy Leveson, July 2013

Analytic Reduction (2)

2. Components act the same when examined singly as when playing their part in the whole
   - Events not subject to feedback loops and non-linear interactions
3. Principles governing the assembling of components into the whole are themselves straightforward
   - Interactions among subsystems simple enough that can be considered separate from behavior of subsystems themselves
   - Precise nature of interactions is known
   - Interactions can be examined pairwise

Called **Organized Simplicity**

© Copyright Nancy Leveson, July 2013

Statistics

- Treat system as a structureless mass with interchangeable parts
- Use Law of Large Numbers to describe behavior in terms of averages
- Assumes components are sufficiently regular and random in their behavior that they can be studied statistically

Called **Unorganized Complexity**

© Copyright Nancy Leveson, July 2013

Complex, Software-Intensive Systems

- 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

Called **Organized Complexity**

© Copyright Nancy Leveson, July 2013

Systems Theory

- Developed for biology (von Bertalanffly) and engineering (Norbert Weiner)
- Basis of system engineering and system safety
  - ICBM systems of the 1950s
  - Developed to handle systems with “organized complexity”

© Copyright Nancy Leveson, July 2013

Systems Theory (2)

- Focuses on systems taken as a whole, not on parts taken separately
  - Some properties can only be treated adequately in their entirety, taking into account all social and technical aspects
  - These properties derive from relationships among the parts of the system

    How they interact and fit together

- Two pairs of ideas
  1. Hierarchy and emergence
  2. Communication and control

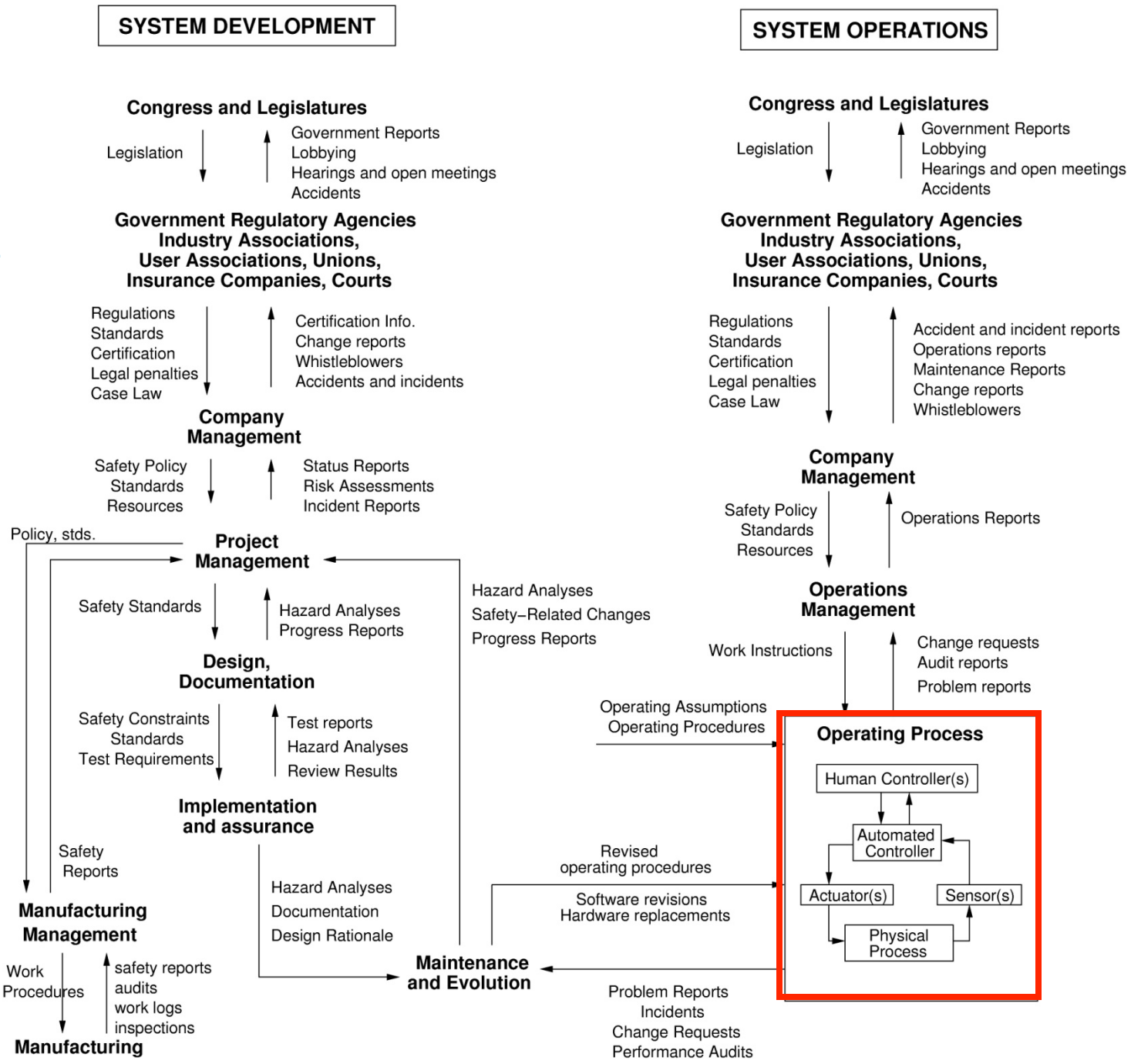
© Copyright Nancy Leveson, July 2013

Hierarchy and Emergence

- Complex systems can be modeled as a hierarchy of organizational levels
  - Each level more complex than one below
  - Levels characterized by emergent properties
    - Irreducible
    - Represent constraints on the degree of freedom of components at lower level
- Safety is an emergent system property
  - It is NOT a component property
  - It can only be analyzed in the context of the whole
- Security is another emergent property

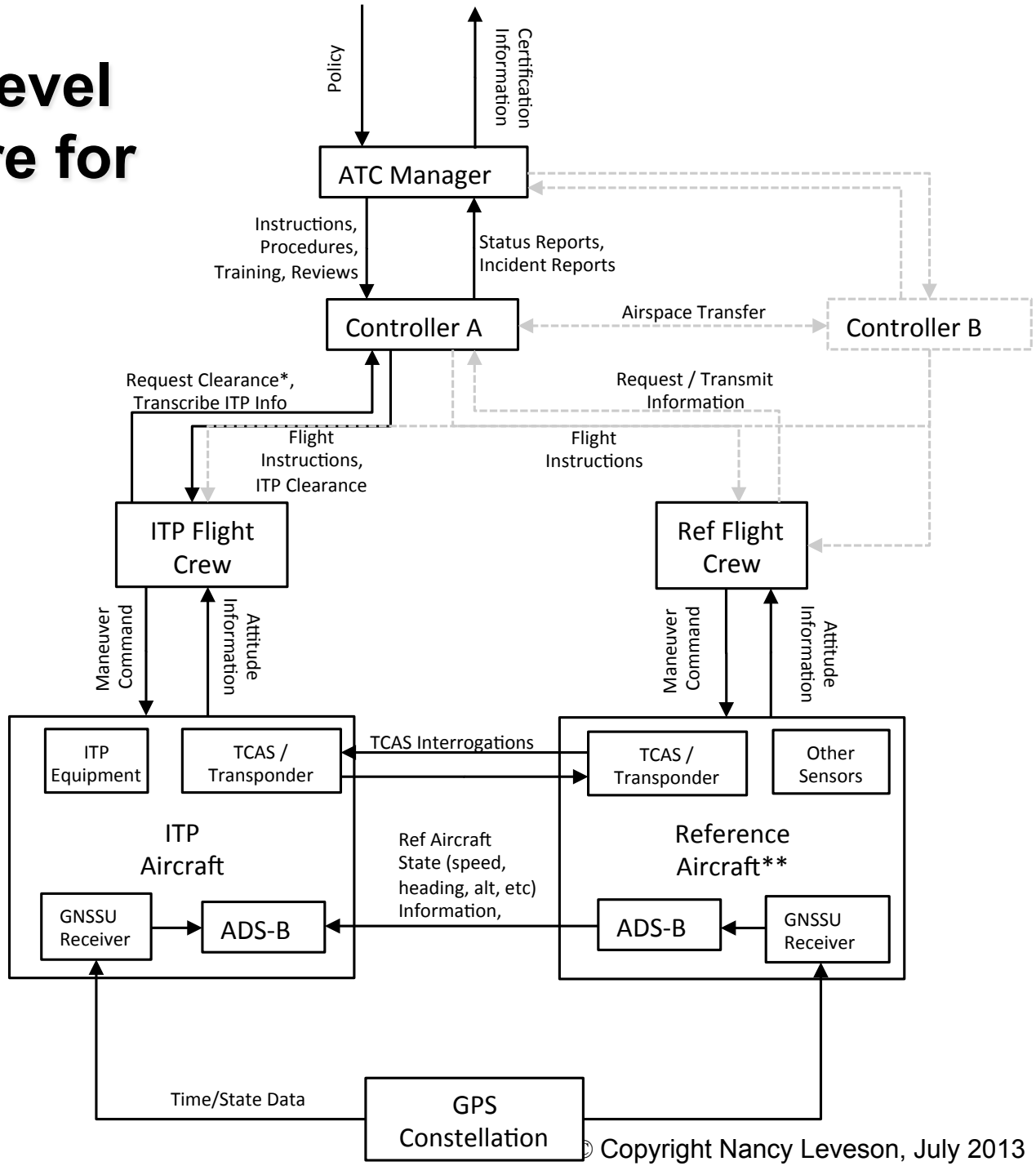
© Copyright Nancy Leveson, July 2013

Example Safety Control Structure



© Copyright Nancy Leveson, July 2013

Example High-Level Control Structure for ITP



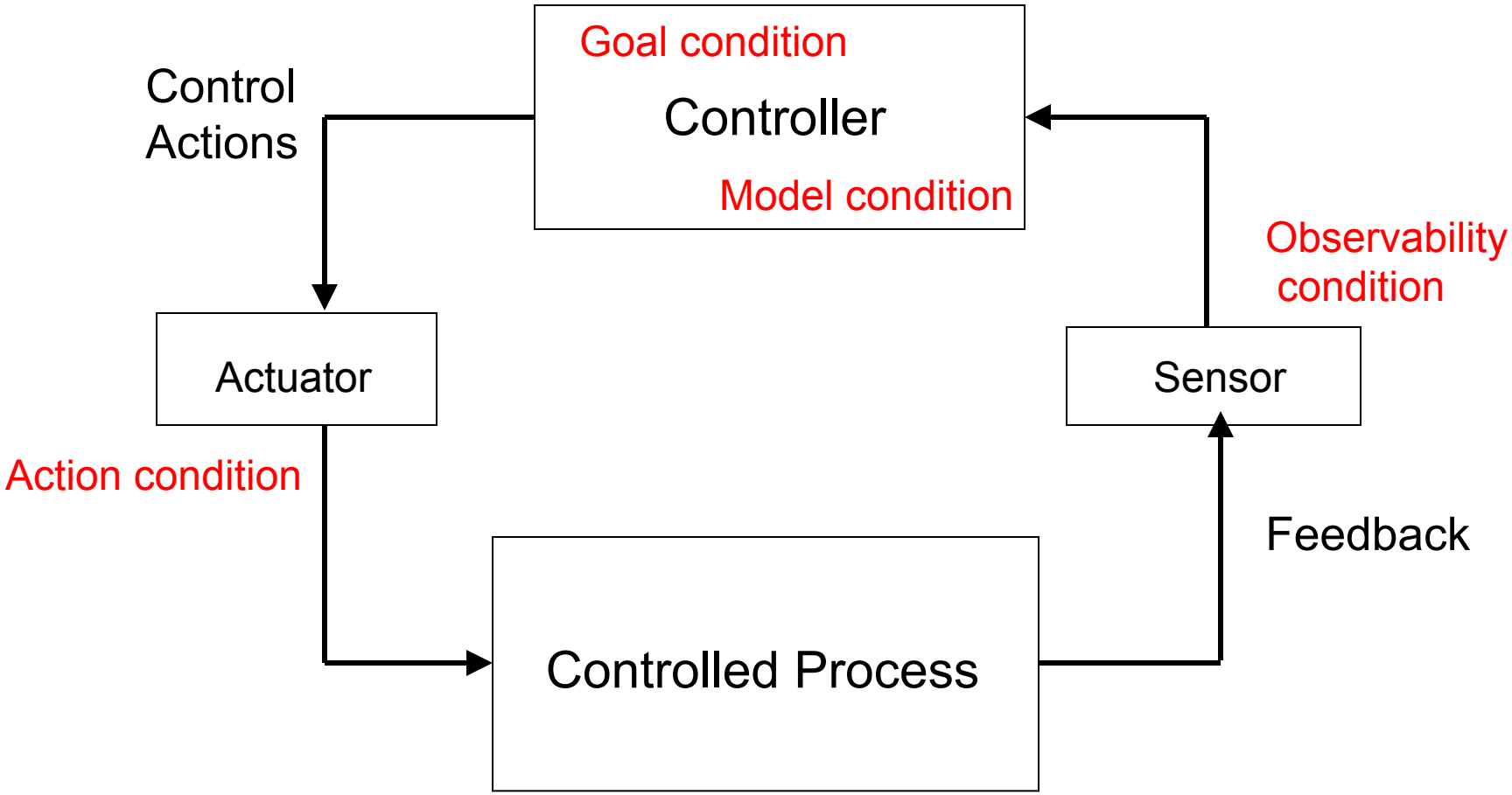
© Copyright Nancy Leveson, July 2013

Communication and Control

- Hierarchies characterized by control processes working at the interfaces between levels
- A control action imposes constraints upon the activity at a lower level of the hierarchy
- Systems are viewed as interrelated components kept in a state of dynamic equilibrium by feedback loops of information and control
- Control in open systems implies need for communication

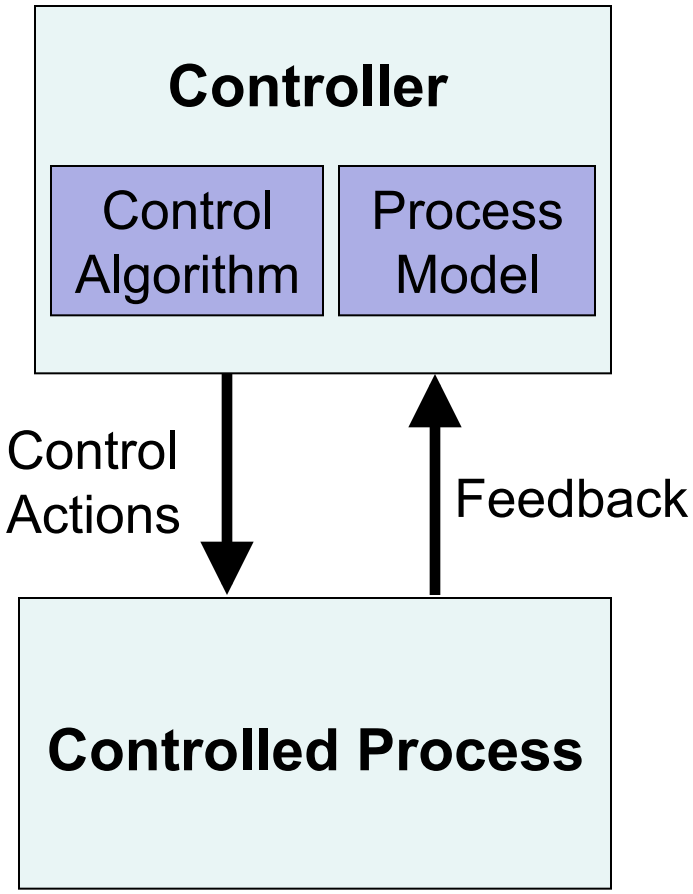
© Copyright Nancy Leveson, July 2013

Control processes operate between levels of hierarchy



© Copyright Nancy Leveson, July 2013

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 hazardous 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, July 2013

STAMP: Safety as a Control Problem

- Safety is an emergent property that arises when system components interact with each other within a larger environment
  - A set of constraints related to behavior of system components (physical, human, social) enforces that property
  - Accidents occur when interactions violate those constraints (a lack of appropriate constraints on the interactions)
- Goal is to control the behavior of the components and systems as a whole to ensure safety constraints are enforced in the operating system.

© Copyright Nancy Leveson, July 2013

STAMP (2)

- Accidents involve a complex, dynamic "process"
  - Not simply chains of failure events
  - Arise in interactions among humans, machines and the environment
- Treat safety as a dynamic control problem rather than a reliability problem

© Copyright Nancy Leveson, July 2013

Examples of Safety Constraints

- Power must never be on when access door open
- Two aircraft must not violate minimum separation
- Aircraft must maintain sufficient lift
- Public health system must prevent exposure of public to contaminated water and food products

© Copyright Nancy Leveson, July 2013

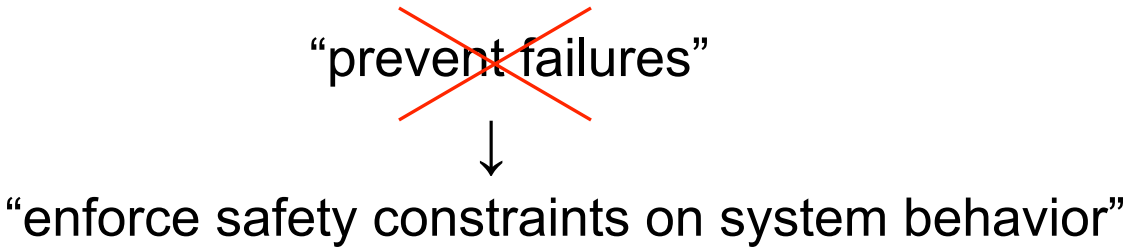
Safety as a Dynamic Control Problem

- Examples
  - O-ring did not control propellant gas release by sealing gap in field joint of Challenger Space Shuttle
  - Software did not adequately control descent speed of Mars Polar Lander
  - At Texas City, did not control the level of liquids in the ISOM tower
  - In Deepwater Horizon, did not control the pressure in the well
  - Financial system did not adequately control the use of financial instruments

© Copyright Nancy Leveson, July 2013

Safety as a Dynamic Control Problem (2)

- Events are the result of the inadequate control
  - Result from lack of enforcement of safety constraints in system design and operations
- Losses (accidents) are the result of complex dynamic processes where the safety constraints are not enforced by the safety control structure
- A change in emphasis:

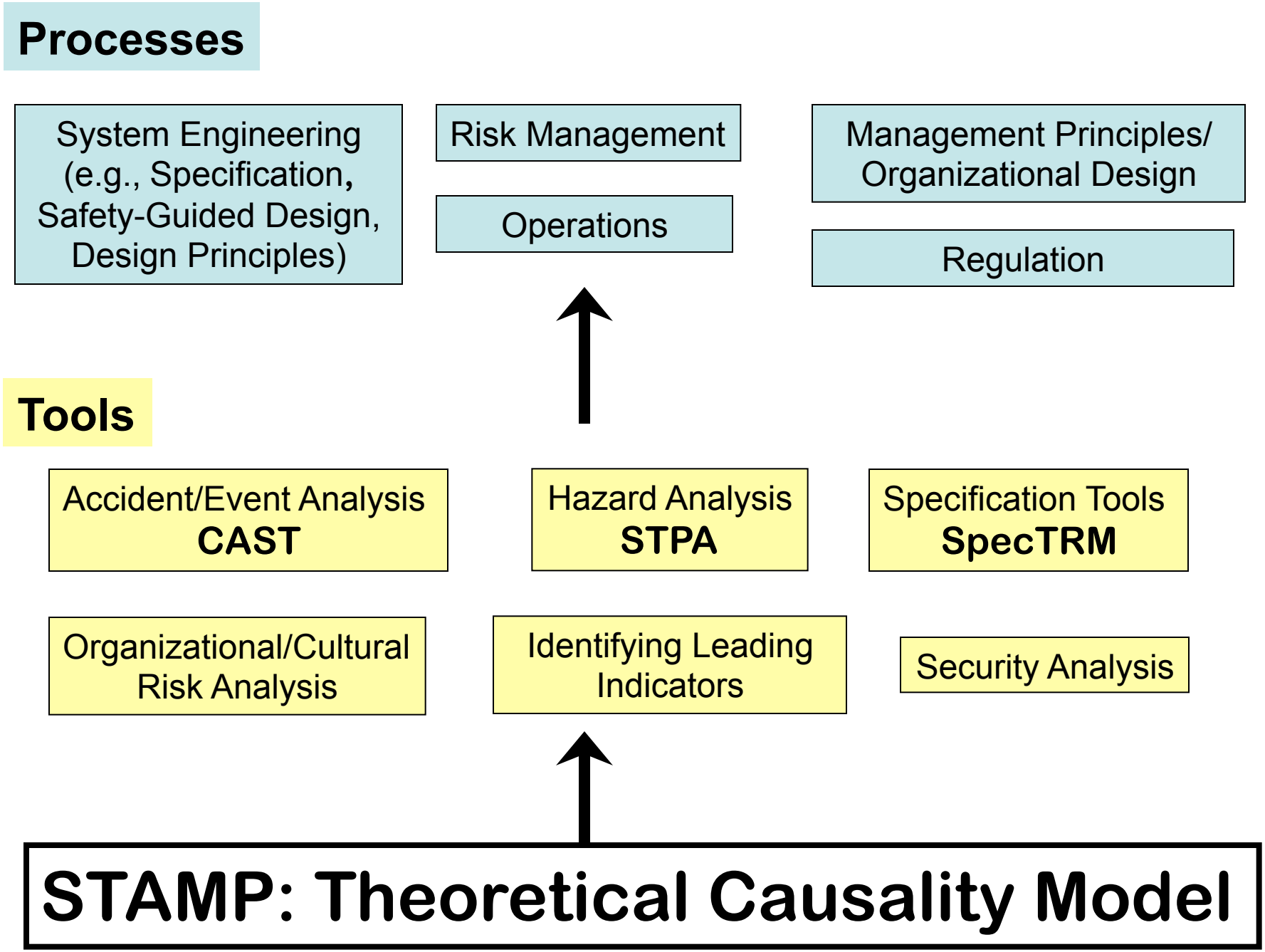


© Copyright Nancy Leveson, July 2013

Safety as a Control Problem

- Identify the safety constraints
- Design a control structure to enforce constraints on system behavior and adaptation
  - Physical design (inherent safety)
  - Operations
  - Management
  - Social interactions and culture

© Copyright Nancy Leveson, July 2013



© Copyright Nancy Leveson, July 2013

STPA

- Integrated into system engineering
  - Can be used from beginning of project
  - Safety-guided design
- Works on social and organizational aspects of systems
- Generates system and component safety requirements (constraints)
- Identifies flaws in system design and scenarios leading to violation of a safety requirement (i.e., a hazard)

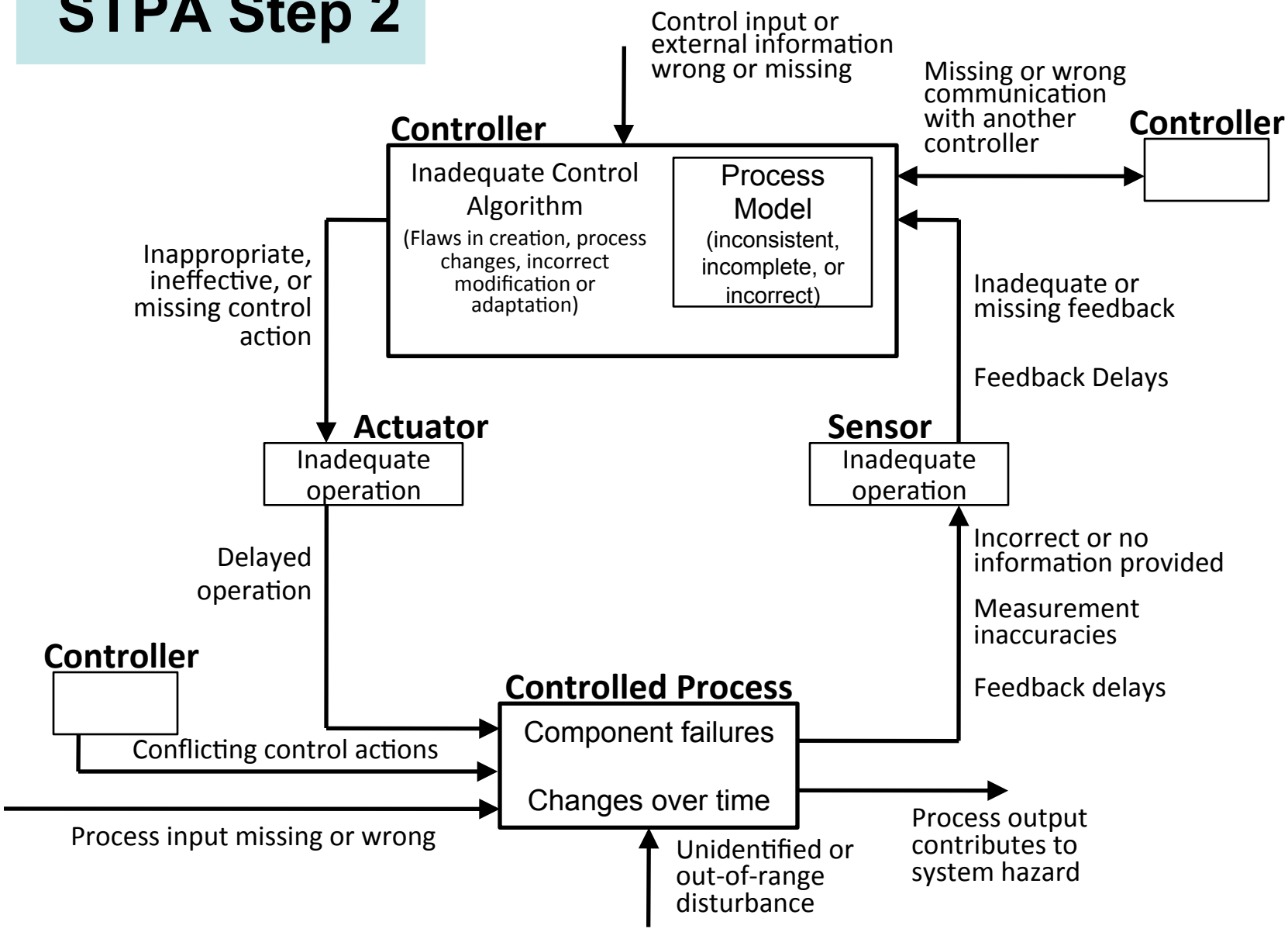
© Copyright Nancy Leveson, July 2013

Steps in STPA

- Identify accidents
- Identify hazards
- Construct functional control structure
- Identify unsafe control actions
- Define system and component safety requirements
- Identify causal scenarios for unsafe control actions
- Augment system and component safety requirements

© Copyright Nancy Leveson, July 2013

STPA Step 2



© Copyright Nancy Leveson, July 2013

Is it Practical?

- STPA has been or is being used in a large variety of industries
  - Spacecraft
  - Aircraft and Integrated Modular Avionics
  - Air Traffic Control
  - UAVs (RPAs)
  - Defense
  - Automobiles
  - Medical Devices
  - Chemical plants
  - Oil and Gas
  - Nuclear and Electrical Power
  - $C0_2$ Capture, Transport, and Storage
  - Etc.

© Copyright Nancy Leveson, July 2013

Is it Practical? (2)

Social and Managerial

- Analysis of the management structure of the space shuttle program (post-Columbia)
- Risk management in the development of NASA's new manned space program (Constellation)
- NASA Mission control ─ re-planning and changing mission control procedures safely
- Food safety
- Safety in pharmaceutical drug development
- Risk analysis of outpatient GI surgery at Beth Israel Deaconess Hospital
- Analysis and prevention of corporate fraud
- UAVs in civilian airspace

© Copyright Nancy Leveson, July 2013

Does it Work?

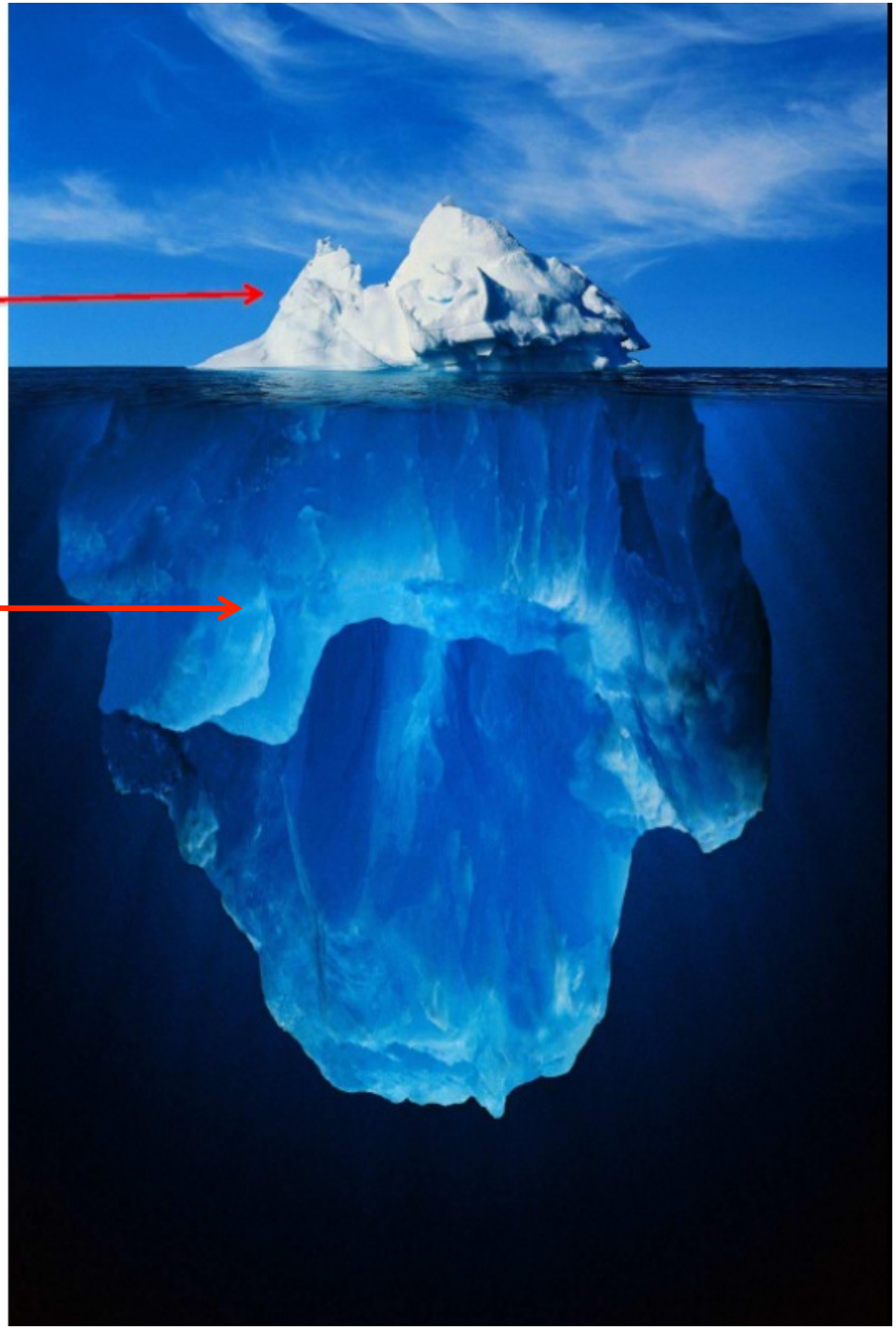
- Most of these systems are very complex (e.g., the U.S. Missile Defense System)
- In all cases where a comparison was made:
  - STPA found the same hazard causes as the old methods
  - Plus it found more causes than traditional methods
  - Sometimes found accidents that had occurred that other methods missed
  - Cost was orders of magnitude less than the traditional hazard analysis methods

© Copyright Nancy Leveson, July 2013

LEVERAGE

Event-based thinking

Systems Thinking



WYOUNG@MIT.EDU
© Copyright William Young, 2012

Integrated Approach to Safety and Security

- Both concerned with losses (intentional or unintentional)
- Starts with defining unacceptable losses
  - "What": essential services to be secured
  - "What" used later to reason thoroughly about "how" best to guard against threats
  - Analysis moves from general to specific
    - Less likely to miss things
    - Easier to review

© Copyright Nancy Leveson, July 2013

Strategy vs. Tactics

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

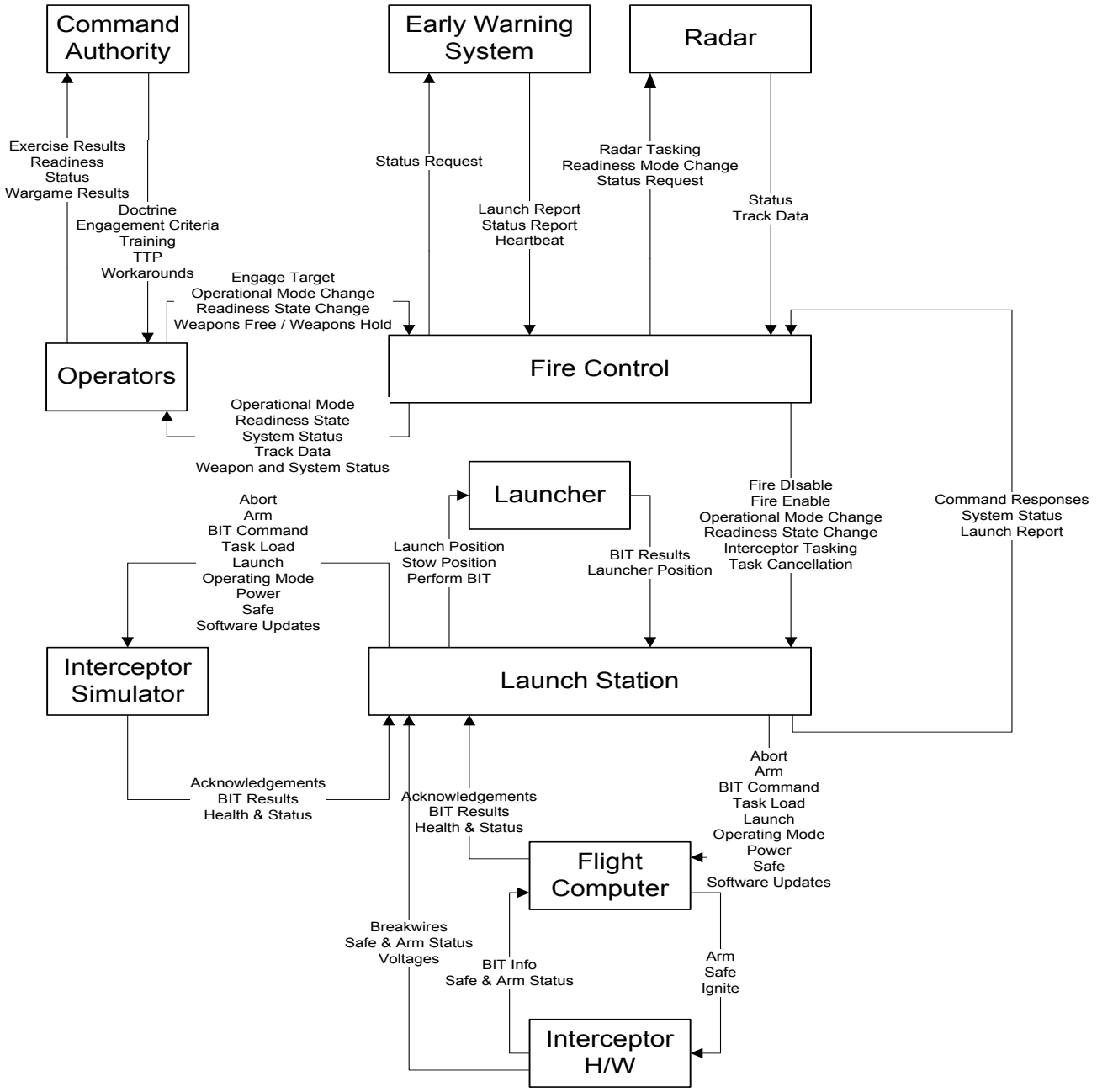
© Copyright Nancy Leveson, July 2013

Top-Down Approach

- Starts with identifying losses and safety/security constraints
- Build functional control model
  - Controlling constraints whether safety or security
  - Includes physical, social, logical and information, operations, and management aspects
- Identify unsafe/unsecure control actions and causes for them
  - May have to add new causes, but rest of process is the same

© Copyright Nancy Leveson, July 2013

Safety Control Structure Diagram for FMIS



© Copyright Nancy Leveson, July 2013

Example: Stuxnet

- Loss: damage to reactor (in this case centrifuges)
- Hazard/Vulnerability: Centrifuges are damaged by spinning too fast
- Constraint: Centrifuges must never spin above maximum speed
- Hazardous control action: Issuing *increase speed* command when already spinning at maximum speed
- One potential cause:
  - Incorrect process model: thinks spinning at less than maximum speed
  - Could be inadvertent or advertent

© Copyright Nancy Leveson, July 2013

Evaluation

- Informal so far but with real red teams
  - Went through STPA-Sec steps
  - Found things they had not thought of before
- Formal experiment in Spring 2014

© Copyright Nancy Leveson, July 2013

Summary

- Key question: How to control vulnerabilities, not how to avoid threats
- Starts with system vulnerabilities and moves down to identify threats (top-down systems engineering approach)

  vs.

  Starting with threats and moving up to vulnerabilities they might exploit to produce a loss (bottom-up approach)
- Elevates security problem from guarding network to higher-level problem of assuring overall function of enterprise.

© Copyright Nancy Leveson, July 2013