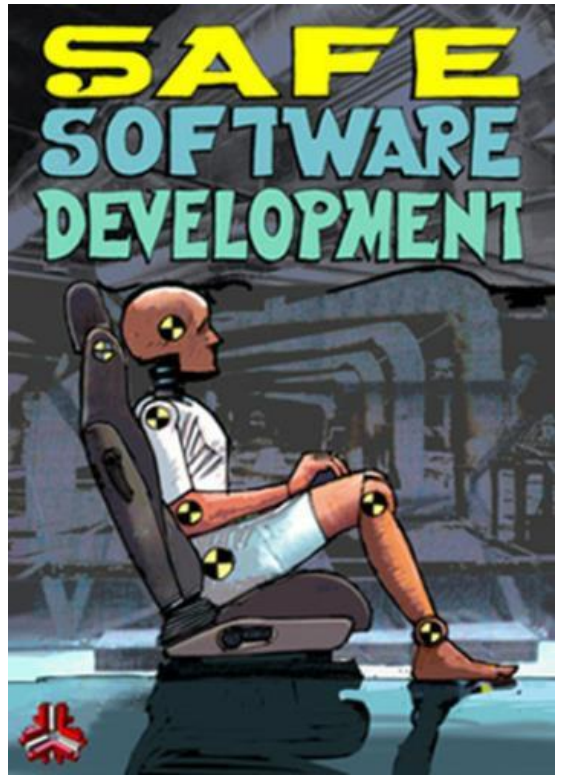


UT Dallas

Introduction to Software Safety

Part 1

Background, Introduction

How Software Contributes to Safety

and why we need to be concerned about it

Software Safety

- **This is an introduction to the topic of software safety**
- **It is based on existing government and commercial standards**
- **It is intended to explain:**
  - what software safety means,
  - how software can contribute to safety problems, and
  - what techniques are used to deal with safety-critical software

Part 1 Agenda

- **Background, Introduction**
- **How Software Contributes to Safety and why we need to be concerned about it.**
- **Exercise 1**
- **The Safety Process**
- **Exercise 2 (homework)**

Part 2

- **What we can Do About Software Safety**

Part 1 Agenda

➢ Background, Introduction

- How Software Contributes to Safety and why we need to be concerned about it.
- Exercise 1
- The Safety Process
- Exercise 2 (homework)

Why Be Concerned about Software Safety?

Can Software Harm Anyone?

Software by itself seems pretty harmless …

So Why are Many Industries and Government Organizations Concerned about Software Safety?

- **Air travel**
- **Military**
- **Power generation (especially nuclear)**
- **Transportation (including modern automobiles)**
- **Dam construction**
- **Medical Devices & Records**
- **…**

All deal with potentially dangerous equipment that, more and more, *requires complex software to function*

Software Involvement in Modern Aircraft

- About 1/3 of the development cost of a modern aircraft is for software
- Over 80% of the functions (by count) are performed in software
- Many cases of "pilot error" involve software assistance

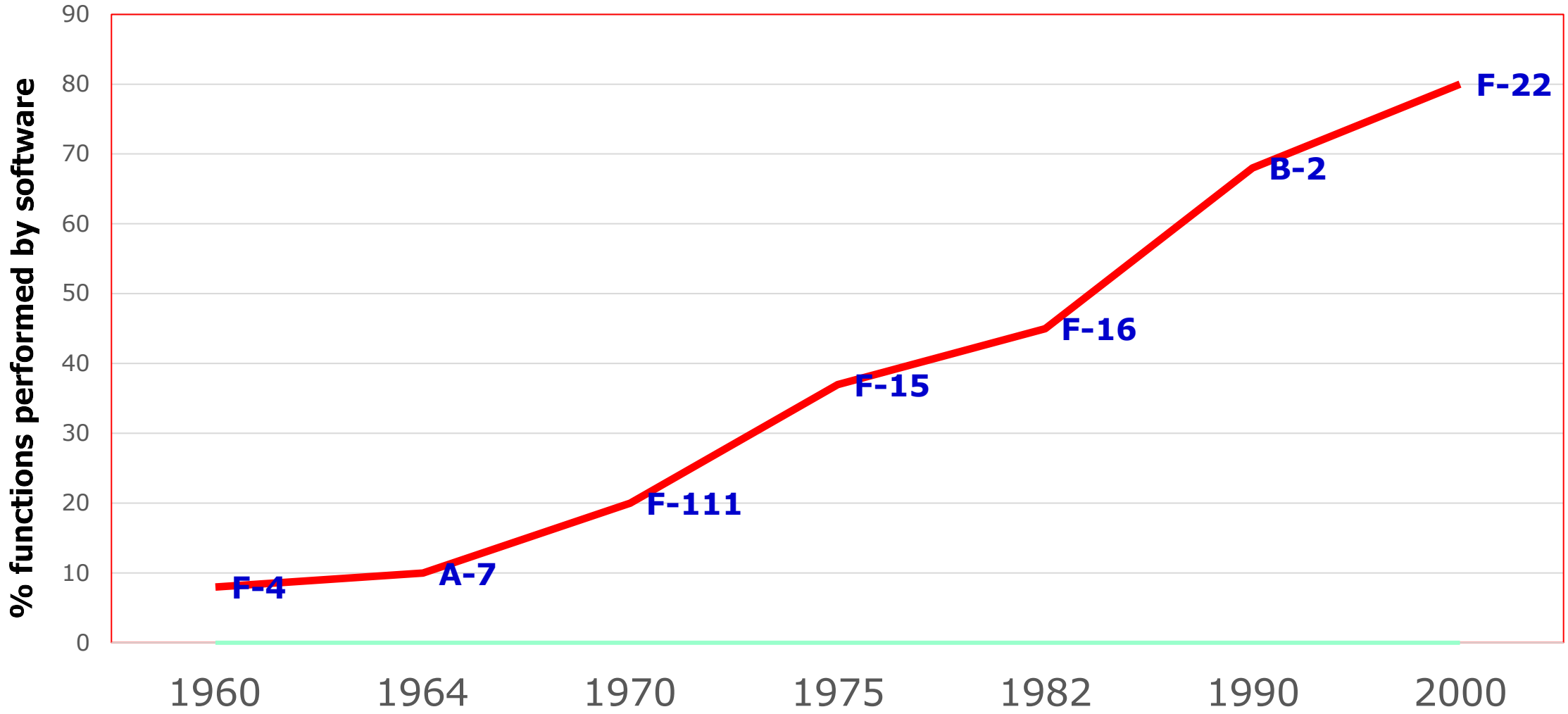
Errors that Brought Down the Boeing 737 Max

Some Key Findings

- Boeing rushed the 737 Max to market as quickly as possible.
- Boeing cut corners.
  - Pilots would only need a 2.5-hour iPad training to fly the plane
- MCAS is the new **software system** blamed for the deadly Lion Air and Ethiopian Airlines crashes.
  - But its failure … was the result of Boeing and the FAA's reluctance to properly inform pilots … or to regulate it for safety.
- The FAA has admitted to being **incompetent when regulating software**
  - Nowhere in its amended certification of the 737 Max is MCAS mentioned.

https://www.theverge.com/2019/5/2/18518176/boeing-737-max-crash-problems-human-error-mcas-faa

US Air Force Aircraft Software Dependence

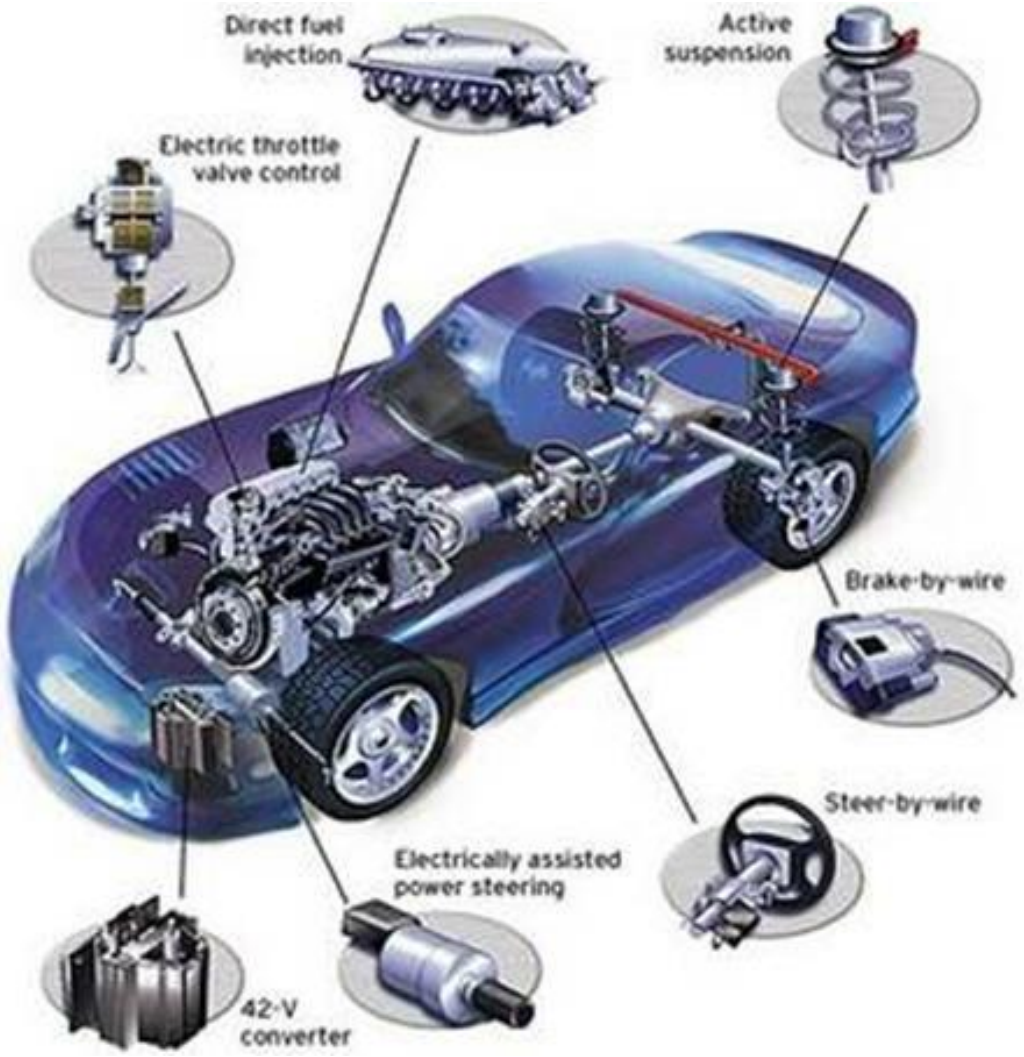


DoD Defense Science Board Task Force on Defense Software, November 2000

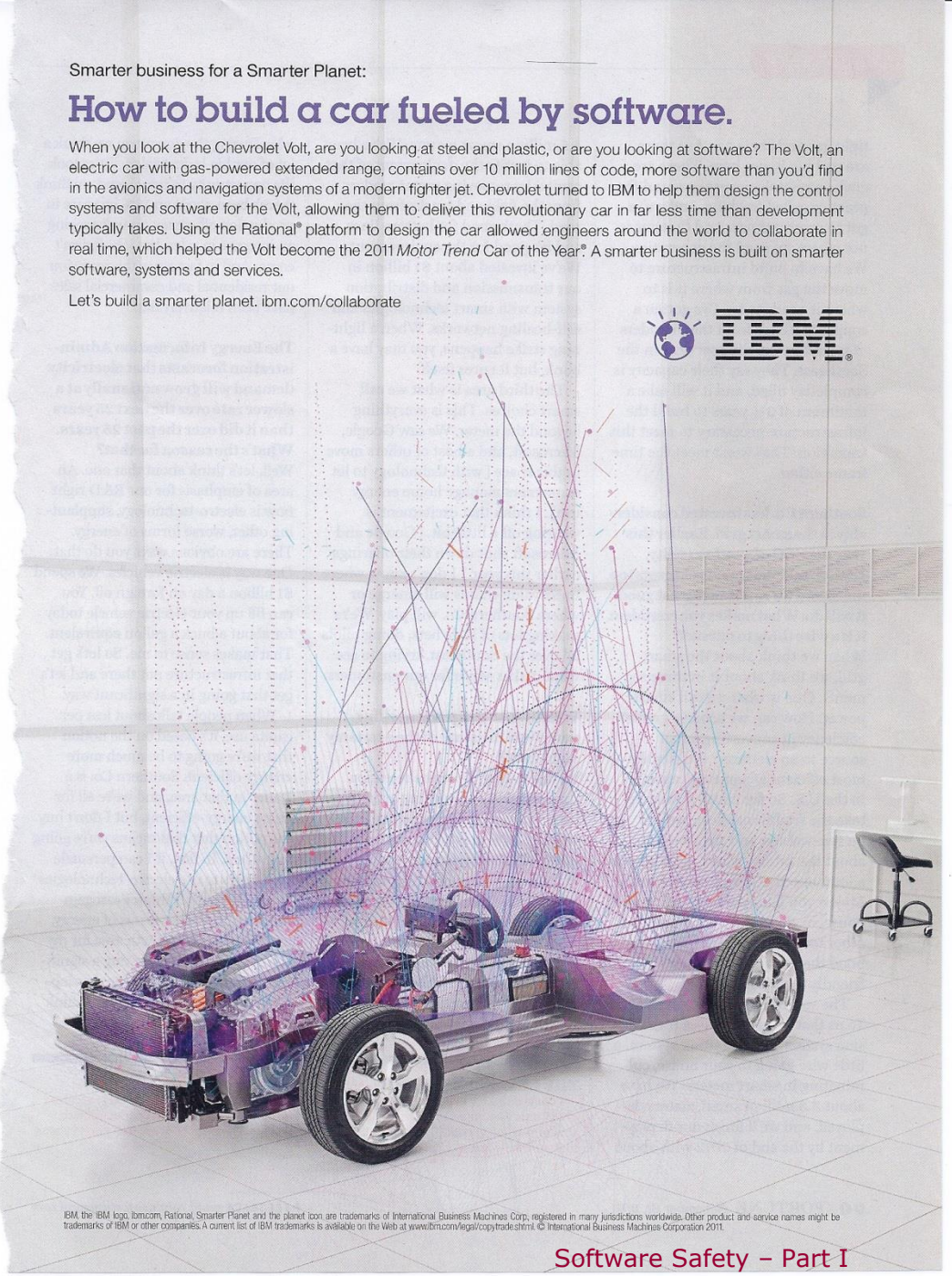
Some Software Functions are Very Sophisticated

Appropriate Center of Gravity control in fuel system can increase fatigue life of wings

Software Involvement in Modern Automobiles



About 25% of the capital cost of a new car is electronics (software, microprocessors, etc.)


Smarter business for a Smarter Planet:
How to build a car fueled by software.
When you look at the Chevrolet Volt, are you looking at steel and plastic, or are you looking at software? The Volt, an electric car with gas-powered extended range, contains over 10 million lines of code, more software than you'd find in the avionics and navigation systems of a modern fighter jet. Chevrolet turned to IBM to help them design the control systems and software for the Volt, allowing them to deliver this revolutionary car in far less time than development typically takes. Using the Rational® platform to design the car allowed engineers around the world to collaborate in real time, which helped the Volt become the 2011 Motor Trend Car of the Year®. A smarter business is built on smarter software, systems and services.
Let's build a smarter planet. ibm.com/collaborate
IBM
IBM, the IBM logo, ibm.com, Rational, Smarter Planet and the planet icon are trademarks of International Business Machines Corp, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml. © International Business Machines Corporation 2011.

Software Involvement in Modern Medical Equipment

Medical equipment is highly computerized and highly dependent on sophisticated software

Robotics in Medicine - Today

Computer Aided Surgery is expected to become common in many medical facilities due to its:

- low invasiveness,
- lower hand vibration and
- highly accurate positioning

➢ ***Reducing physical burdens on the patient***

da Vinci Surgical System

- More than 2700 installations

- ***da Vinci* Surgery is used today for:**
  - ✓Cardiac Surgery
  - ✓Colorectal Surgery
  - ✓General Surgery
  - ✓Gynecologic Surgery
  - ✓Head & Neck Surgery
  - ✓Thoracic Surgery
  - ✓Urologic Surgery

Robotics in Medicine - Future

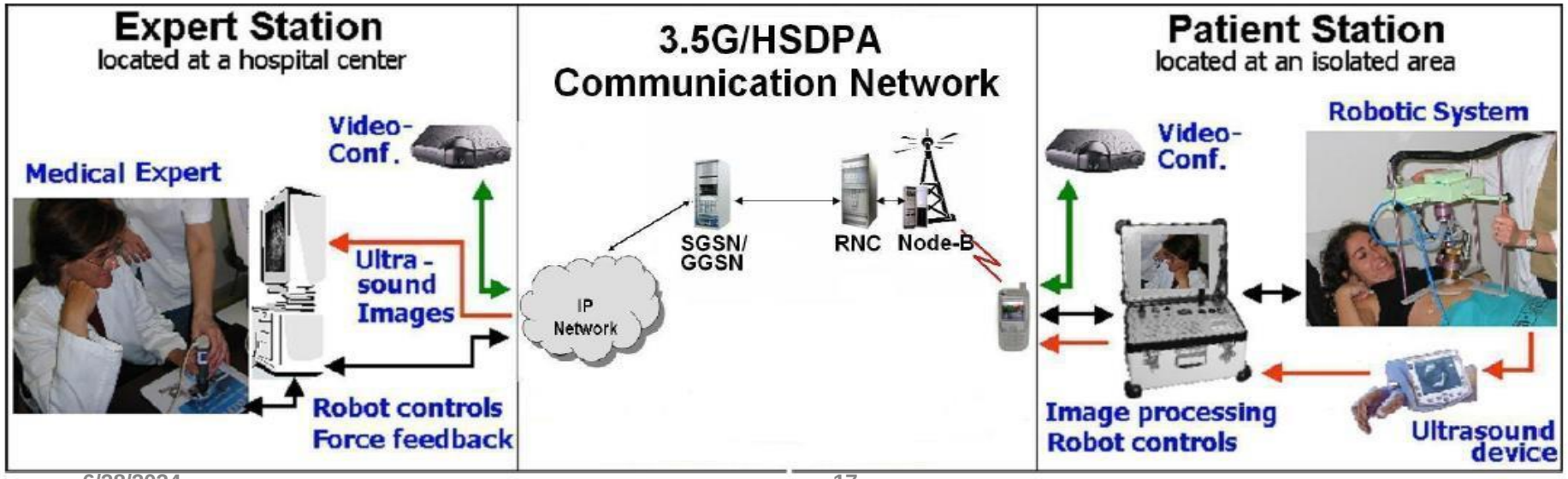
Robotics will have a great potential to allow patients, elderly, and pregnant to undertake

Tele-[checkup, diagnosis, therapy]

- **beyond hospital, region, country, and even continent**

Tele-echography robot, will be applicable to

critical care in long-travel ships and aircraft



Who Develops All of This Software?

Software Engineers!

What's Different from Normal Software Development?

- It's very important for the software to operate correctly
- Or, at the very least, in a way that doesn't result in harm
  - All the Time!
  - Under Usual and Unexpected Circumstances
  - Under Conditions of Operator Error
  - Under Conditions of Hardware Error

So What's Different about Safety Critical Software Development?

Much more rigorous development practices

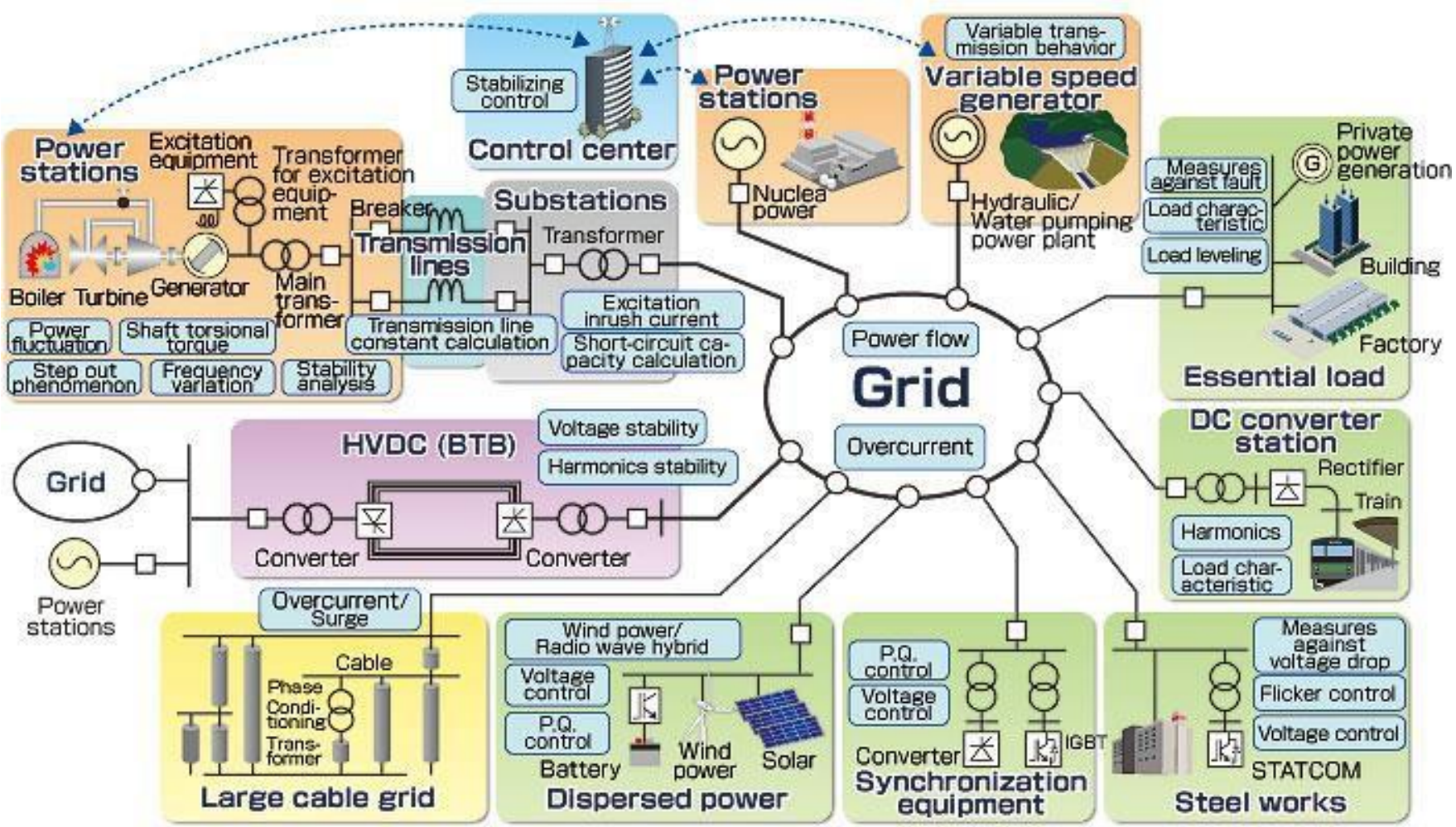
Characteristics of Safety Critical Projects

- Intense focus on identifying ways the system can fail and how to prevent that
- Rigorous development process, from requirements through all phases of product development
- Independent audits of each development step to assure that nothing was done incorrectly
- Rigorous safety standards that must be met

But That's Not Enough

You Also Need to Think Differently About What You Are Doing

Thinking at the System Level instead of at the module level



Software Often Replaces Hardware and People

Many software tasks were once done in hardware.

- **And some of those were once done by people**

In many cases, software today performs functions that are too dangerous, too quick, or too complex for humans to perform.

Therac 25 - a radiation therapy machine produced by Atomic Energy of Canada Limited (AECL) in 1982

Therac-25 Safety Problems

courtesy of Wikipedia

[Therac-25] was involved in at least six accidents between 1985 and 1987

- in which patients were given **massive overdoses of radiation**

Because of **concurrent programming errors**
it sometimes gave its patients
radiation doses that were
hundreds of times greater than normal,
resulting in **death or serious injury**

Diagnosis of Therac-25 Failure[1]

A commission concluded that the primary reason was the bad software design and development practices

- **and not explicitly to several coding errors that were found.**

In particular, the software was designed so that it was realistically impossible to test it in a clean, automated way.

[1] *Leveson, Nancy, University of Washington (1995). "Medical Devices: The Therac-25 Accidents" (PDF). Safeware: System Safety, and Computers (Update of the 1993 IEEE Computer article ed.). Addison-Wesley.*

Part 1 Agenda

- Background, Introduction
- **How Software Contributes to Safety and why we need to be concerned about it.**
- Exercise 1
- The Safety Process
- Exercise 2 (homework)

Ways that Software can Harm Someone

- **It can Control the Behavior of Dangerous Devices**
  - Robots
  - Weapons
  - Security Doors at Building entry
  - Medical Devices
  - Chemical Experiments
  - Factory Manufacturing Lines
  - …
- **It can Send Information to People who do Potentially Dangerous Things**
  - Location of Airplanes for Air Traffic Control
  - Identification of Intruders
  - …
- **It can Deceive**
  - Internet scams
- **And on and on …**

What is "Safe" Software?

Software is "safe" if …

- It has features and procedures that ensure it performs predictably under normal and abnormal conditions
- The likelihood of an undesirable event occurring in that software is minimized
- If an undesirable event does occur, the consequences are controlled and contained

Per D. Herrman, *Software Safety and Reliability*

Key Concepts

- **No software-based, safety critical system is 100% "safe"**
  - It is not possible to test all possible conditions or prove that it will never fail under any circumstances
  - Instead, safety experts ask two questions:
    - "**How likely is the software to fail**?"
    - "**If it fails, how severe will the consequences be**?"

  } **Goal: very low**

- **Software safety can only be evaluated within the context of the system the software is part of**
  - Including the people who interact with the system

Some Existing Software Safety Standards

Aerospace Systems – SAE AS9006A

Aircraft – RTCA DO-178C

Automobiles – BS ISO 26262; SAE JA 1002

Information Technology Equipment – IEC 60950-1

Medical Devices - ANSI/AAMI/ISO 14971; ANSI/AAMI/IEC 62366; FDA 21 CFR 820.70; IEC 62304; ISO 13485; MDD (European Council Directive 93/42/EEC)

NASA – STD-8719.13C

Nuclear Power – ANSI/IEEE 7-4.3.2; IEC 60880

Programmable Electronic Systems – IEC 61508; ANSI/UL 1998

Railway Systems - BS EN 50128

US Military Applications – MIL-STD-882E

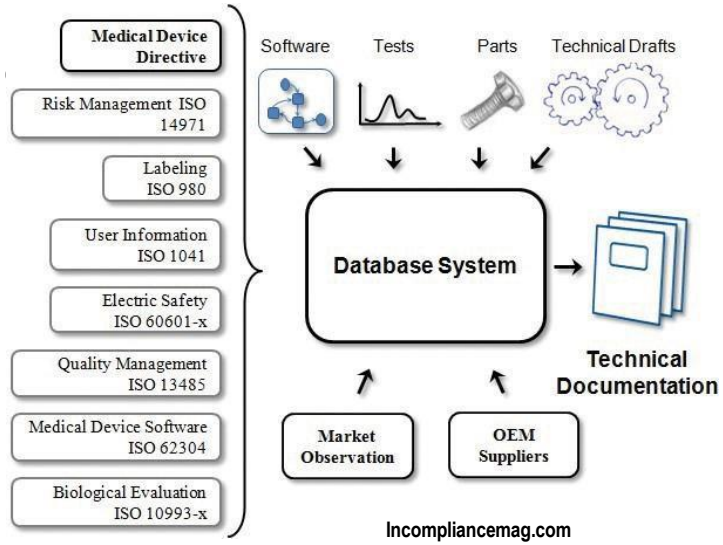
What These Standards Typically Specify

- **Processes and procedures to be followed when developing safety-critical software**
  - Development Practices
  - Documentation
  - Verification and Validation
  - Etc.
- **Risk Management processes and procedures**
  - Including risk control, analysis, mitigation and abatement
  - Both top down (functional) and bottom up (code/design)
  - Risk classification classes or categories
- **Processes and procedures for other situations**
  - Often specific to the application

Coordination Among Standards

- **There is some coordination among these standards**
  - But applications differ so much that there are often major differences
  - And even within an application domain, there may be many standards that are not necessarily well coordinated

- **Sometimes it is difficult to navigate through all the applicable standards**
  - Example: a medical data base for use in t h
    - International safety standards
    - International medical safety standards
    - European safety standards
    - European medical device standards
    - US safety standards



Software Safety Starts with System Safety

- **Software is always part of a system**
  - A data base
  - A network
  - A vehicle
  - …

- **If the system can harm someone, then the software may be a factor in whether the system harms someone**
- **So we have to start by analyzing the safety issues of the system**

Basic Terminology (Military; Nuclear Power)

Mil-STD 882E

- **Hazard** – Any **real** or **potential condition** that can cause **injury, illness, or death** [to a person] **or damage to … or loss of** … [property or the environment]
- **Mishap** – An **unplanned event** or series of events resulting in **death, injury, …**

➢ **Safety – Freedom from … conditions** that can cause **injury, illness, …**

I.e., get rid of the hazards so mishaps don't result in harm

In Other Words …

A Hazard is what might result in something going wrong

A Mishap is an accident that might *cause* it to go wrong

A **Hazard** is an accident waiting to happen

www.boredpanda.com

A **Mishap** is the accident

Hazards and Mishaps

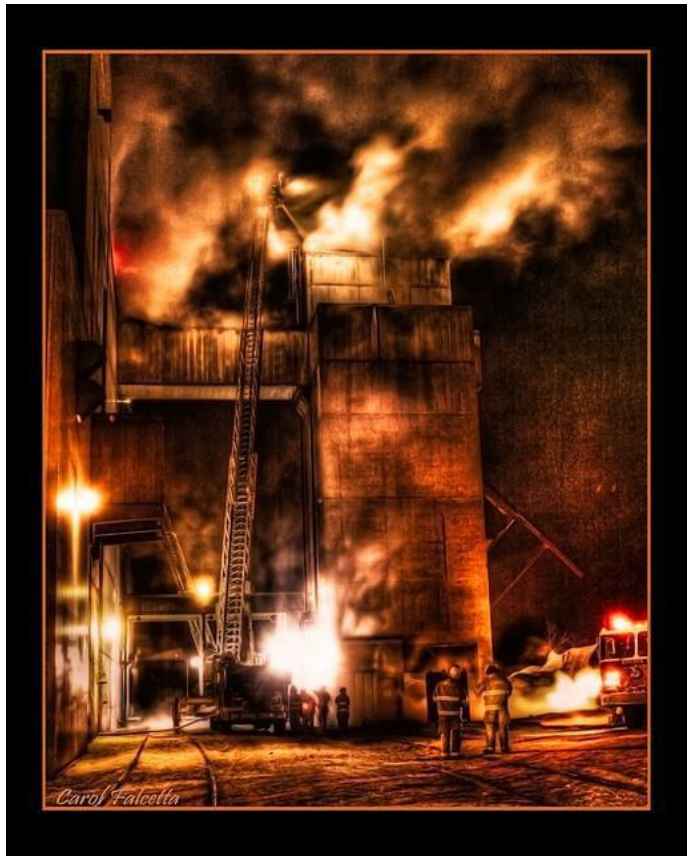
Hazard	Mishap
Parent leaves medicine on counter	Child sees medicine and ingests it
Shrubbery hides stop sign	Car fails to stop and hits another car
Smart phones with texting capability in a car	Driver texts and gets into a serious accident
Water spilled on floor and not cleaned up	Someone slips on the water and breaks an arm

Notes about Hazards and Mishaps

- **Nobody intended to cause injury or harm**
- **The mishap was normal behavior under some circumstances**
  - **The hazard was what turned normal behavior into a harmful event**

More Complex Example

- **Hazard**: the power plant might catch fire because of unsafe design or operation
- **Mishap**:
  - an incorrect temperature reading may cause
  - a heater to go on, resulting in
  - overheating of a chemical mixture, leading to
  - a chemical reaction, producing
  - excessive pressure, causing
  - an explosion in the fuel storage area, producing
  - a major fire in the power plant



Basic Terminology (Aircraft Safety)

DO-178C

Failure – The inability of a system … to perform a required function within specified limits.

- Catastrophic – … would prevent continued safe flight and landing
- Hazardous - … would reduce the capability of an aircraft or the ability of the crew to cope with adverse operating conditions … potentially fatal injuries
- Major - … significant discomfort or injuries
- Minor - … inconvenience

Basic Safety Concepts
NASA

NASA-STD-8719.13C

- **Process safety requirements**
  - The way you develop software
  - The inspections and evaluations and tests you perform along the way
- **Technical safety requirements**
  - Specific techniques for software design and development when the software is safety critical

Part 1 Agenda

- Background, Introduction
- How Software Contributes to Safety and why we need to be concerned about it.
- **Exercise 1**
- The Safety Process
- Exercise 2 (homework)

Exercise 1

- **Each student is assigned a scenario**
  - These scenarios all relate to the Covid-19 virus
- **Identify up to three ways in which software might contribute to a safety hazard. For each, identify:**
  - The hazard and how software creates that hazard
  - A mishap that might cause a safety problem
- **Report to the group, after the break**

Time: 20 minutes (10 for break, 10 for exercise)

(When we resume in 20 minutes, each student will present his or her results)

Break / Exercise 1

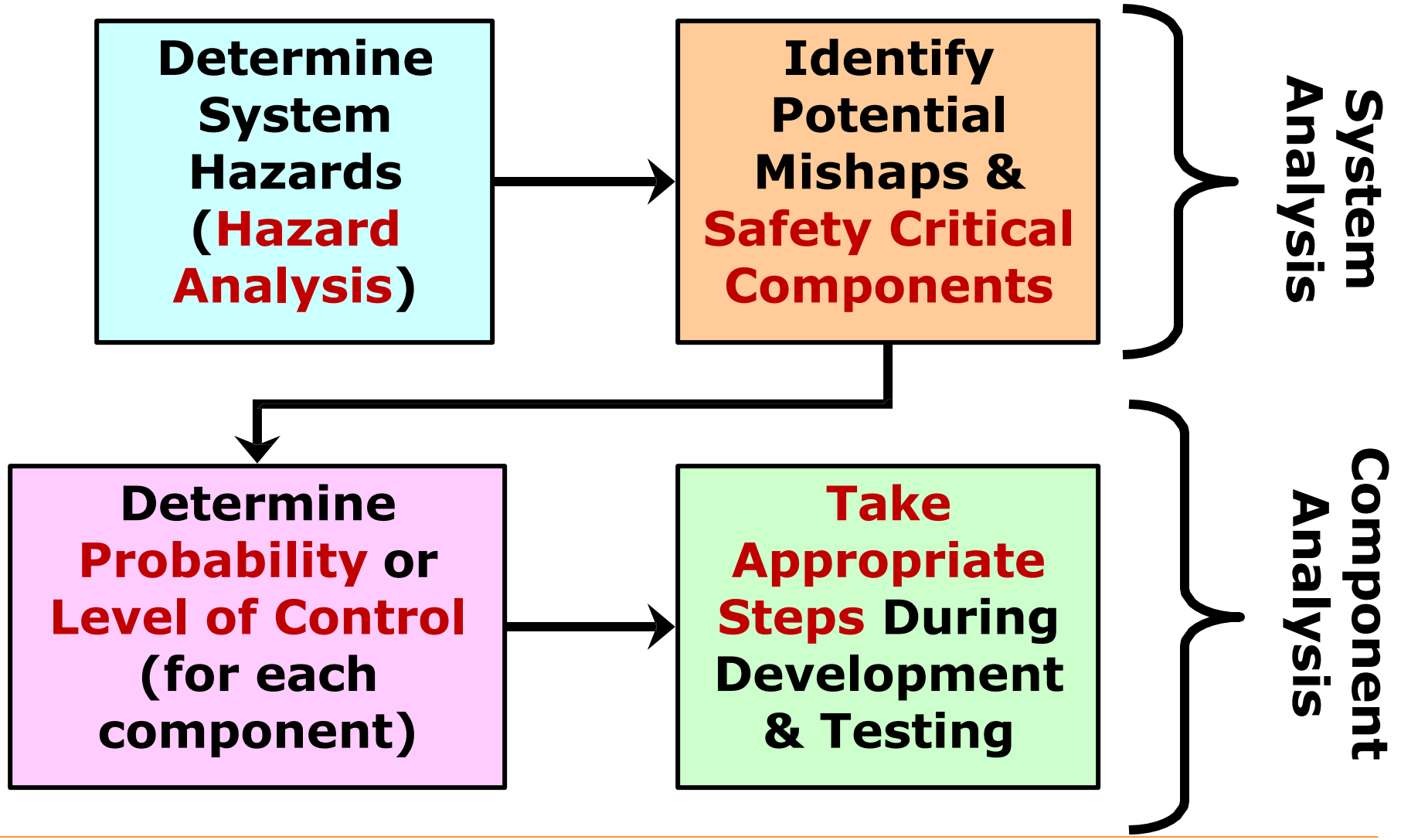
20 minutes

Students Present Exercise Results

Part 1 Agenda

- Background, Introduction
- How Software Contributes to Safety and why we need to be concerned about it.
- Exercise 1
- **The Safety Process**
- Exercise 2 (homework)

The Safety Process
(varies somewhat with different standards)



Basic System Safety Process in More Detail

1. ***Identify*** **the potential hazards**
2. ***Decompose*** **hazard threads (potential mishaps) through subsystem components, including software**
3. ***Link/trace*** **to requirements**
4. ***Generate*** **appropriate mitigation strategy**
5. ***Implement*** **the mitigation**
6. ***Verify*** **that the mitigation is implemented and that it functions as expected**
7. ***Document*** **safety artifacts**

Hazard Analysis Approaches

Traditional - treat it as a reliability issue

- Identify the components that may fail and contribute to a hazard
- Use various approaches to make the failures less likely or less harmful
- Much of the analysis occurs during or after the design step

Emerging - treat it as a system control issue

- Identify the ways the system may fail and contribute to a hazard
- Create system requirements and constraints that forbid these conditions
- The analysis occurs before the design is started

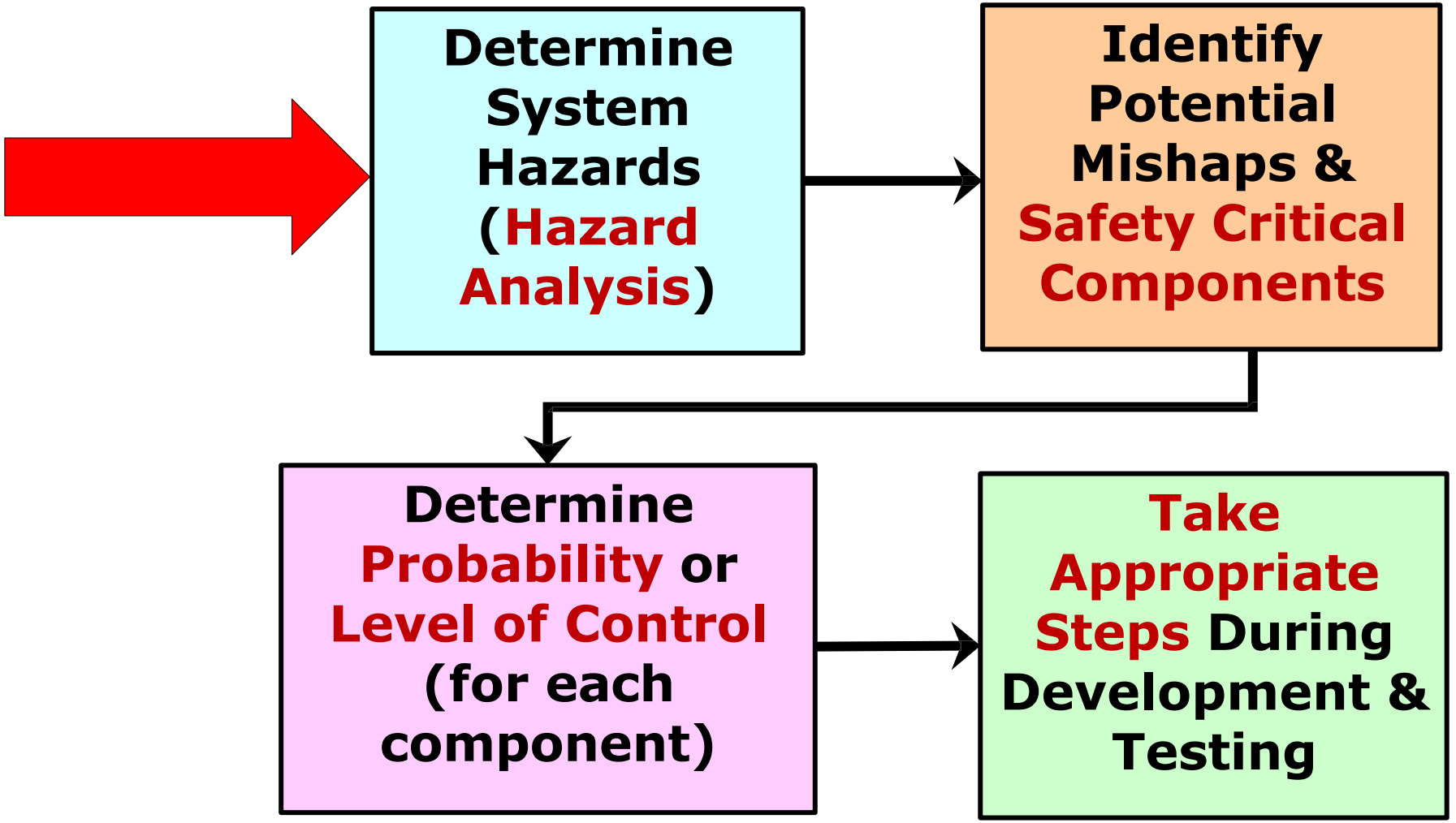
Safety Engineering requires that we do both!

The Traditional Approach

Software is a Component of a System

Make Component Failure Less Likely or Less Harmful

[develop very high quality software]


Determine System Hazards (Hazard Analysis)
Identify Potential Mishaps & Safety Critical Components
Determine Probability or Level of Control (for each component)
Take Appropriate Steps During Development & Testing

System Level Hazard Analysis

Determine System Hazards (Hazard Analysis)

FMEA – Failure Mode and Effects Analysis

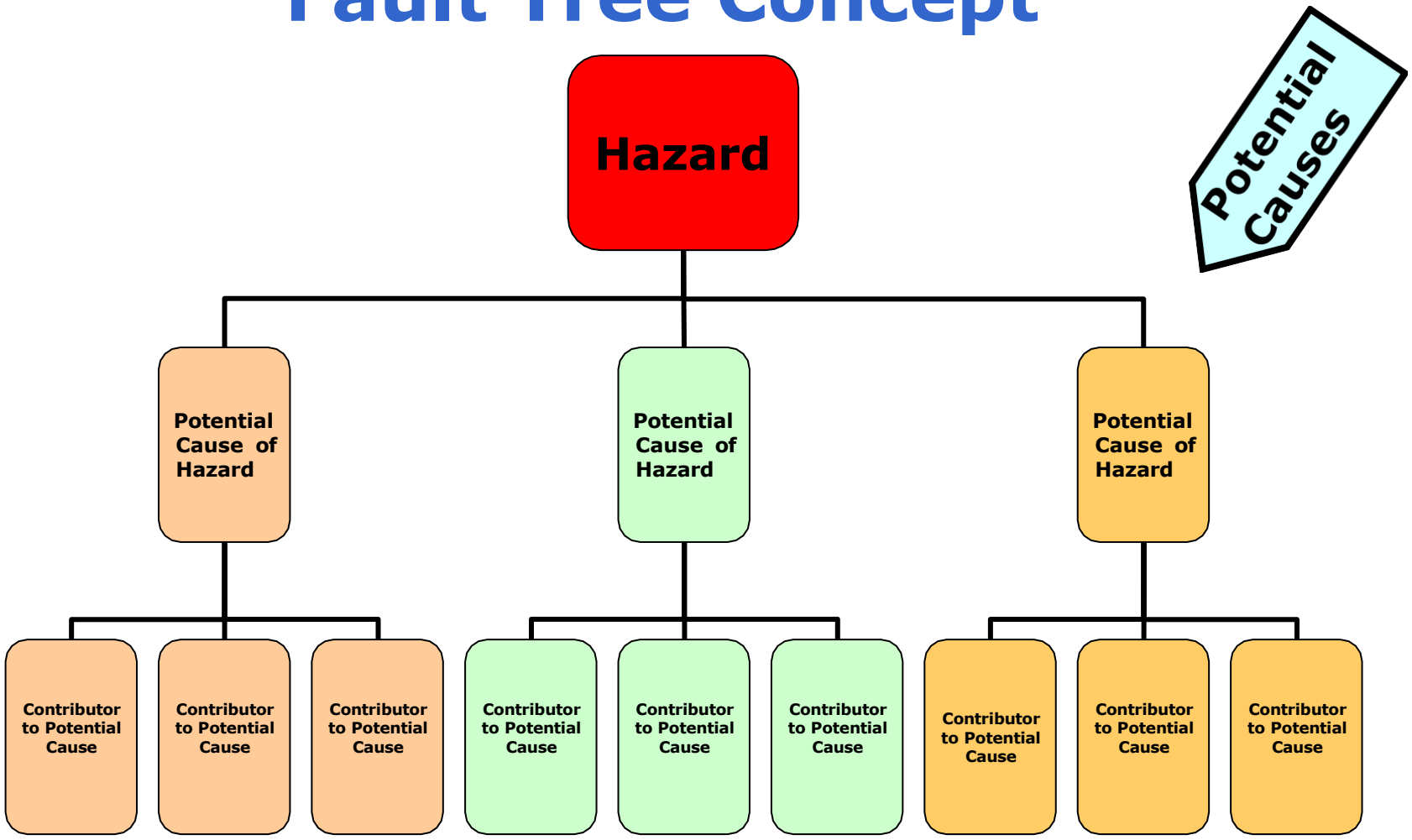
- A reliability engineering technique to identify:
  - Modes of failure
  - Causes of those failures
  - Effects of those failures
- Can be focused on:
  - The functionality of the product
  - The design of the product
  - The process used to develop or produce the product

FMECA – Failure Mode, Effects and Criticality Analysis

- Extends FMEA to include criticality analysis (how severe is the potential failure)

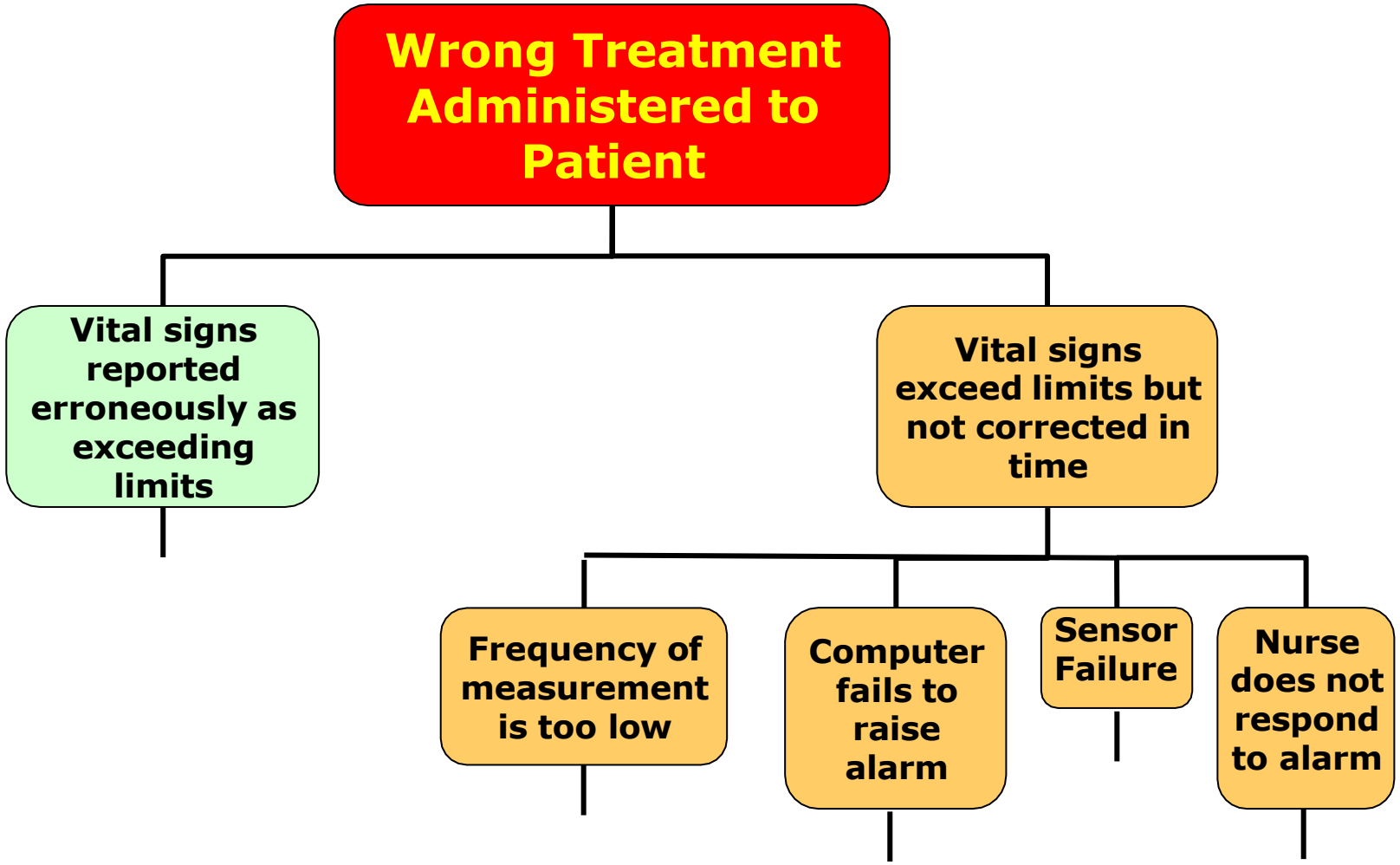
An FMEA Technique

Fault Tree Concept



Keep going down into more detailed causes until you reach a point where you can do something about it.

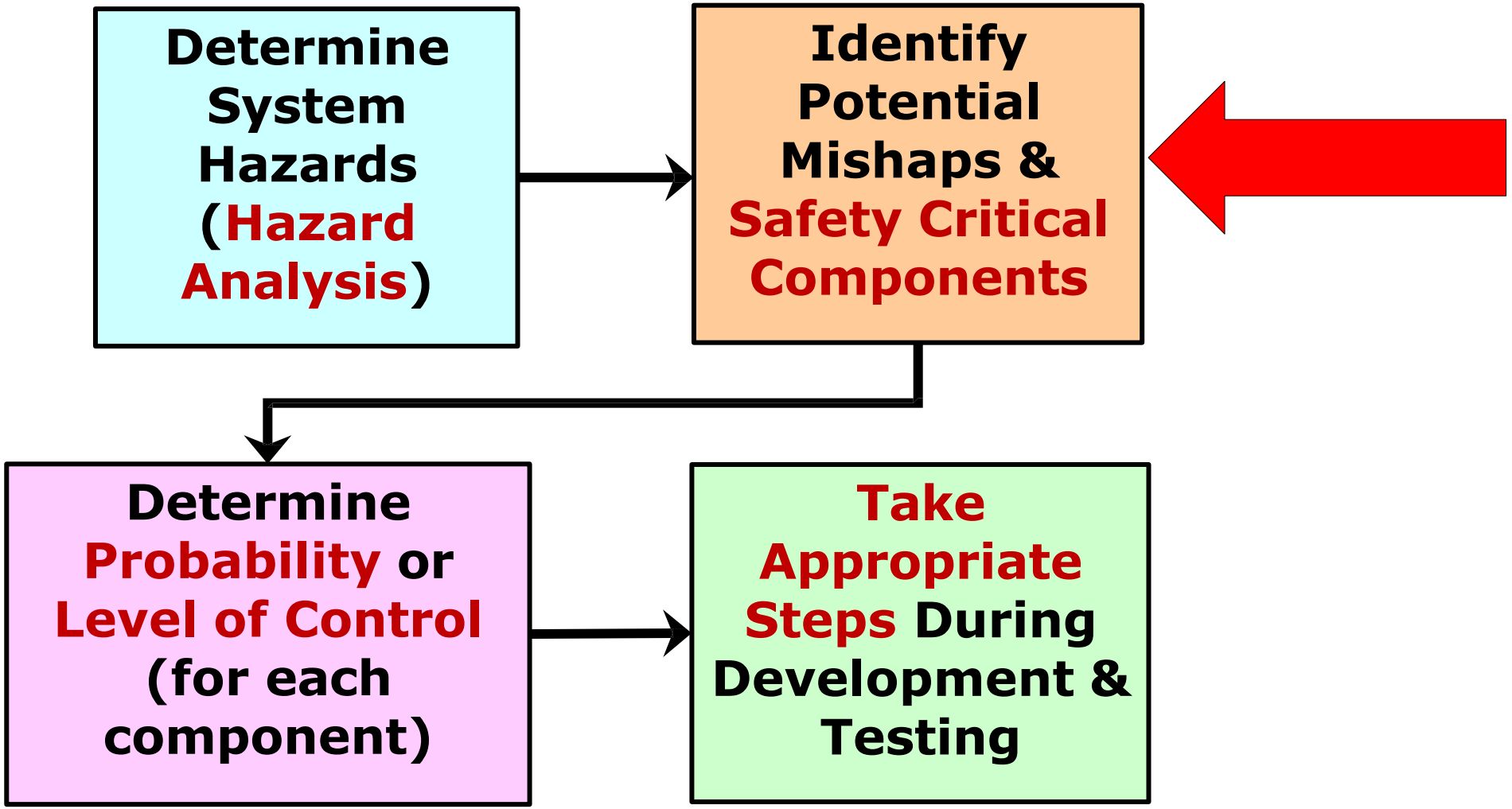
Fault Tree Example



An Example of a System Level Hazard List

Hazard

- Uncontrolled explosion
- Uncontrolled fire
- Injury and/or illness
- Blockage of ingress/egress paths
- Structural failure
- Collision
- Uncontrolled activation of ordinance
- Electromagnetic interference
- Hazardous/reactive materials
- Electrical energy
- Improper engagement control (Fratricide)
- Surface/air contamination
- Corrosion resulting in loss of strength or integrity of exposed surfaces
- Batteries (exposure to toxic material, explosion)
- Radiation (ionizing and non-ionizing)
- Uncontrolled/unsupervised robotic operations


Determine System Hazards (Hazard Analysis)
Identify Potential Mishaps & Safety Critical Components
Determine Probability or Level of Control (for each component)
Take Appropriate Steps During Development & Testing

A Safety Critical Component is …

Identify Safety Critical Components

… a component that might contribute to a hazard if it fails

- **Note that in most hazardous systems there are many components that can contribute to a safety hazard**
- **Some components may be software**

Software is often used to replace mechanical or human controls.

When software controls something it is often potentially safety critical.

But that's not the only way software can be safety critical.

Some Software Functions that May be Safety Critical (1 of 2)

Potential Cause of Hazard

- Assessment of overall system health, such as:
  - power-up and run-time monitors,
  - "heartbeats",
  - program memory CRCs,
  - range checks,
  - CPU health assessments
- Enforcement of critical timing
- Exception trapping and handling including, failure /malfunction detection, isolation, and containment
- Functions that execute the system's response to detected failures/malfunctions

Some Software Functions that May be Safety Critical (2 of 2)

Potential Cause of Hazard

- Functions that enhance the system's survivability (preservation of core functionality)
- Data quarantine/sanitization
- Range and reasonableness (sanity) checking
- Tamper proofing and cyber-attack proofing
- Authentication for lethal actions
- Inhibiting functions and interlocks

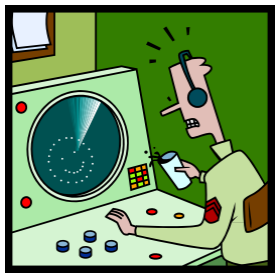
Example: Software Contribution to a Mishap (1/6)

Example SW Contribution	Symptom	Example Potential Cause
Incorrect Safety-Critical Alerts and Warnings.	Safety Critical alerts are incorrect, or are not triggered by Safety Critical Events. Alerts fail to warn the user of an unsafe condition, and or an Unsafe System State.	Software fails to alert the operator to unsafe condition and/or state. Alerts can be audio, visual, or in text format in a log, etc..

False Alarm

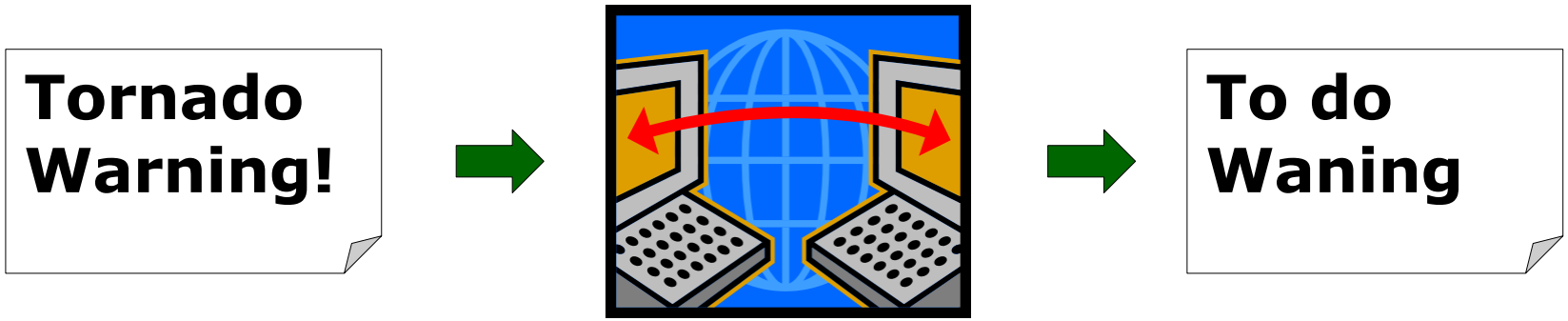
or

No Alarm when Needed



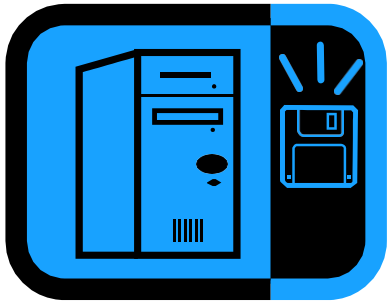
Example: Software Contribution to a Mishap (2/6)

Example SW Contribution	Symptom	Example Potential Cause
Incorrect Data Transfer Messages (transmit and receive).	Data transferred in the wrong format and the Safety Critical Data are interpreted incorrectly.	Failure to validate data transfer with the appropriate parity, check sums to validate Safety Critical data.



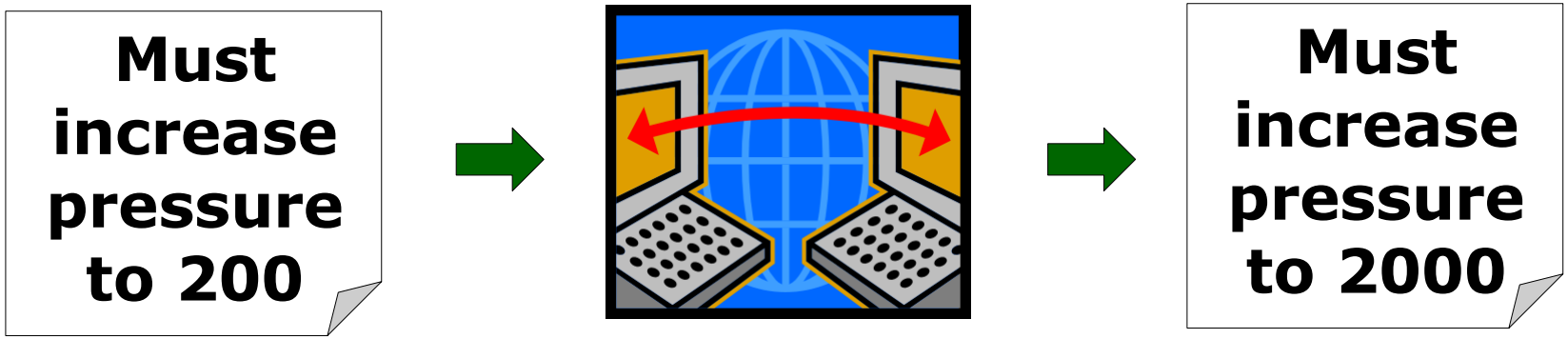
Example: Software Contribution to a Mishap (3/6)

Example SW Contribution	Symptom	Example Potential Cause
Data Storage Failures (Safety Critical Data corrupted and/or lost)	Safety-Critical Displays are confusing, and/or incorrect in presenting Safety Critical Data.	Safety critical data were not properly check-summed; data overwritten by mistake



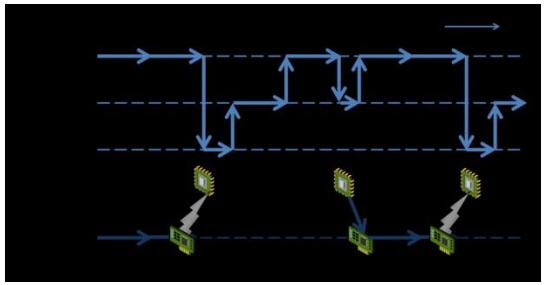
Example: Software Contribution to a Mishap (4/6)

Example SW Contribution	Symptom	Example Potential Cause
Incorrect data transfer between processors	Incorrect message received	Failure to perform verification checks in both processors prior to transferring Safety Critical data.



Example: Software Contribution to a Mishap (5/6)

Example SW Contribution	Description	Example Potential Cause
Timing and Interrupt failures	Interrupts occur at the wrong time and potentially interrupt a Safety Critical process, or introduce a potential hazard.	Interrupts are out of synch with system time, and/or interrupt a Safety Critical Process/Path which introduces a potential hazard.



Example:
Software Contribution to a Mishap (5/6)

Example SW Contribution	Description	Example Potential Cause
Incorrect Modes	Software signals the system to fire a weapon when it should not.	Switch from training to "live" mode is not correctly reflected in the "mode" variable.

The Most Common Source of Software Safety Problems

Requirements not stated or communicated correctly

(approximately 70% of test failures)

- Software is "correct" according to how the software developers understood the requirements
- But the requirements were wrong
- Or they were misunderstood
- Or they changed

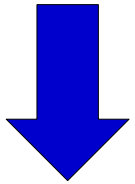
But This is True for Almost All Software!

- **Getting requirements right is difficult**
  - It is hard to understand large, complex systems
  - Some requirements are derived from design decisions
- **Requirements tend to change over time**
- **Errors often occur at organizational boundaries**

ww.tonex.com

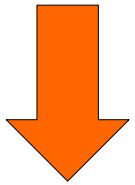
Requirements Flow Across Organizational Boundaries

- **Customer**



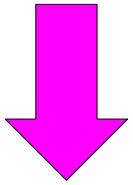
Requirements in Customer's Terminology

- **Systems Analyst or Systems Engineer**



Requirements in Systems Terminology

- **Software Analyst**



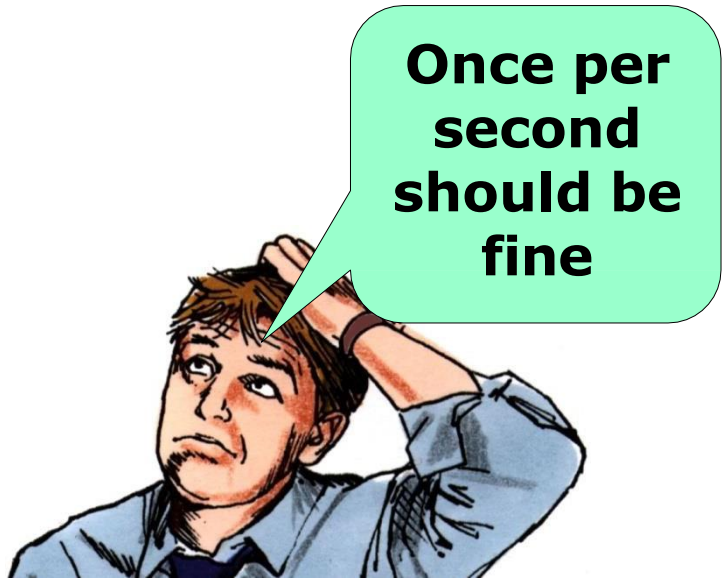
Requirements in Software Terminology

- **Software Engineer/Developer/Coder**

Why Errors Happen at Organizational Boundaries

Different cultures/organizations/vocabularies

- Misinterpretations ("it is obvious that …")



NASA Study of Requirements Errors

85% of requirements errors arose at organizational boundaries

- **Software engineers often did not understand things the system designers thought were obvious**
  - Physics issues
  - Timing issues
  - Complex mathematical and statistical issues
- **Requirements would change**
  - Without suitable assessment of consequences
  - Without communication to software developers

The Second Most Common Sources of Software Safety Problems

Design and coding errors

- Usually resulting from poor code structure that makes it hard to understand
- Sometimes resulting from requirements that are hard to implement in code
  - Example: "no single point of failure"
- Timing errors, syntax errors, algorithm errors, display errors, lack of self-tests, poor error handling, and many others can cause safety hazards
- Bugs don't necessarily map to failures
  - Many bugs have only minor impact or are easily recovered from
  - Some bugs can have catastrophic impact

Underlying Causes of Many Software Safety Problems

- **Change control process**
  - Changes introduce unanticipated errors
  - Change control process not followed correctly
- **Testing process**
  - Safety-related testing is not performed or not performed thoroughly
- **Hardware failures**
  - Hardware fails due to overheating, logic errors, power transients, radiation, magnetic fields, etc.
  - Hardware not designed to inform software of a failure
  - Software not designed to respond properly to hardware failures

Part 1 Agenda

- Background, Introduction
- How Software Contributes to Safety and why we need to be concerned about it.
- Exercise 1
- The Safety Process
- **Exercise 2**

Exercise 2 (safety research)

- **Each person will be assigned a specific safety topic**
- **Research the topic (web / UTD library)**
- **Prepare a brief, introductory presentation**
- **Give the presentation at the next session (10 minutes)**
- **Goal of presentation: give the other students a good idea of some of the software related safety issues associated with your specific safety topic**

EXAMPLE

UT Dallas – Software Safety:

Research, Practice and a Path Forward

Summer, 2024

Child Safety

Your Safety Topic

Sample Student

Your Name

About Me

My name is Sample Student

I'm from Hill and Dale College in Muskegon, Michigan

My college major is computer science

Here is my family

I'm the one in the middle

About Child Safety

There are many potential hazards for children

- Batteries
- Bicycles
- Water/Boating
- Booster Seats
- Burns and Scalds
- Car Seats
- Carbon Monoxide
- Choking and Strangulation
- Driveway Safety
- Falls Apply Falls filter
- Fire Apply Fire filter
- Fireworks
- Getting Ready to Drive
- Guns
- Halloween
- Heatstroke
- Holidays
- In and Around Cars
- Laundry Packets
- Cars
- Toys
- Furniture tip overs
- And many, many more

Software in Child Safety

Software may be found in many child care and child safety devices, such as:

- **Baby monitors**
- **Carbon monoxide detectors**
- **Smoke detectors**
- **Car rear view mirrors**
- **Sports equipment (e.g., concussion monitors)**
- **Cars**
- **Toys**
- **Televisions**

Although software seldom directly controls safety critical functions, it is often used in warnings and must be designed to work correctly, even in the event of device malfunction.

Some Specific Software Issues Encountered in Child Safety Devices

- **Proper response to power failures**
- **Protection from intrusion by network hackers**
  - For devices connected to the internet such as baby monitors
- **Detection of hardware damage (for example, child spills juice onto an electronic monitor in his or her car seat)**
- **Operator error should be very uncommon**
  - Devices should be very easy to use and hard to misuse (because the users are typically not very knowledgeable about computers)
- **Do it right the first time**
  - Parents are unlikely to update software in their baby's toys, monitors, etc.

Any Questions?

End of Part I