Class4: Safety Engineering
Chapter 12

Adeeb Noor, PhD
IT Department
Faculty of Computing and Information Technology
King Abdulaziz University
Fall 2025

Today lecture topic

Lecture 4

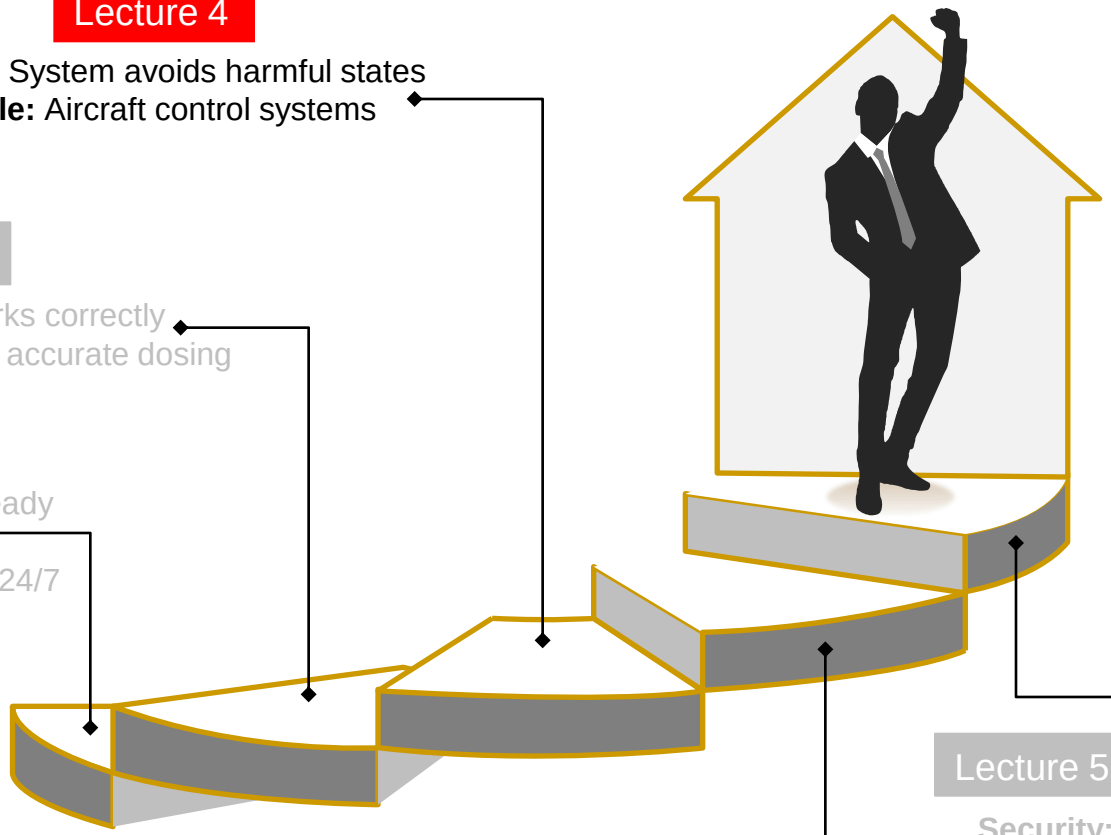
Safety: System avoids harmful states
Example: Aircraft control systems

Lecture 3

Reliability: System works correctly
Example: Insulin pump accurate dosing

Lecture 2

Availability: System is ready when needed
Example: ATM available 24/7



Lecture 6

Resilience: System recovers from failures
Example: Backup power systems

Lecture 5

Security: System resists attacks
Example: Secure against fraud

The Big Picture

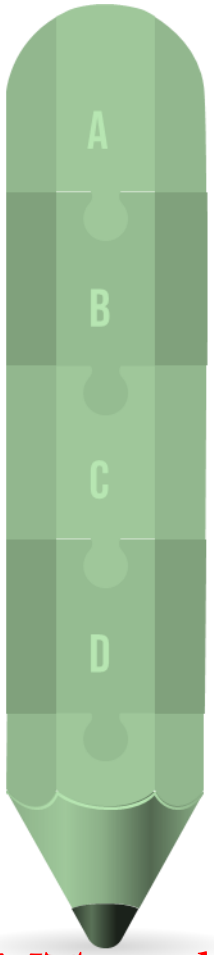
BASICSAFE

"Safety is not an intellectual exercise to keep us in work. It is a matter of life and death. It is the sum of our contributions to safety management that determines whether the people we work with live or die."

-Sir Brian Appleton, safety assessor

Take-home points

- The Key CLO: Design safety in safety critical systems
- Safety-critical systems are those that can cause injury, death, or environmental damage if they fail
- Reliability alone is not enough
- Safety requirements define constraints the system must respect to maintain safety
- Safety engineering processes aim to identify and eliminate hazards through techniques like hazard analysis, formal verification, and static analysis
- The safety case is a structured argument demonstrating that a system is acceptably safe to deploy



CIMT model

Software safety: why matters

12.1 Safety-critical systems

Concept

- What makes a system "safety-critical"? It's one where fail include to:
  - Loss of human life or serious injury
  - Severe environmental damage
  - Major property or financial lose
- Some common examples of safety-critical systems include: Aircraft flight controls, Nuclear power plant control, etc
- Reliable systems can be unsafe
- The implementation of software control systems is essential to enhance system safety

Safety terminologies

Term	Definition
Accident (or mishap)	An unplanned event or sequence of events which results in human death or injury, damage to property, or to the environment. An overdose of insulin is an example of an accident.
Hazard	A condition with the potential for causing or contributing to an accident. A failure of the sensor that measures blood glucose is an example of a hazard.
Damage	A measure of the loss resulting from a mishap. Damage can range from many people being killed as a result of an accident to minor injury or property damage. Damage resulting from an overdose of insulin could be serious injury or the death of the user of the insulin pump.
Hazard severity	An assessment of the worst possible damage that could result from a particular hazard. Hazard severity can range from catastrophic, where many people are killed, to minor, where only minor damage results. When an individual death is a possibility, a reasonable assessment of hazard severity is 'very high'.
Hazard probability	The probability of the events occurring which create a hazard. Probability values tend to be arbitrary but range from 'probable' (say 1/100 chance of a hazard occurring) to 'implausible' (no conceivable situations are likely in which the hazard could occur). The probability of a sensor failure in the insulin pump that results in an overdose is probably low.
Risk	This is a measure of the probability that the system will cause an accident. The risk is assessed by considering the hazard probability, the hazard severity, and the probability that the hazard will lead to an accident. The risk of an insulin overdose is probably medium to low.

12.2 Safety requirements

Implementation

- The goal of safety requirements engineering is to identify protection requirements that ensure that system failures do not cause injury or death or environmental damage

Define "shall not" define situations and events that should never occur:

- Recovery & attack protections
- "Doors shall not open while train is in motion“

Hazard analysis:

- Identification - what hazards could occur?
- Assessment - which are most dangerous/likely?
- Analysis - what events can lead to the hazard?
- Reduction - specify requirements to avoid/handle hazards

Fault-tree analysis:

- What: top-down deduction
- How: root cause analysis
- Goal: reduce single cause failures
- Root causes of software hazards: algorithm & arithmetic errors

Examples of safety requirements

Implementation

SR1: The system shall not deliver a single dose of insulin that is greater than a specified maximum dose for a system user.

SR2: The system shall not deliver a daily cumulative dose of insulin that is greater than a specified maximum daily dose for a system user

SR3: The system shall include a hardware diagnostic facility that shall be executed at least four times per hour

SR4: The system shall include an exception handler for all of the exceptions that are identified in Table 3

SR5: The audible alarm shall be sounded when any hardware or software anomaly is discovered and a diagnostic message, as defined in Table 4, shall be displayed

SR6: In the event of an alarm, insulin delivery shall be suspended until the user has reset the system and cleared the alarm

12.3 Safety engineering processes

Concept

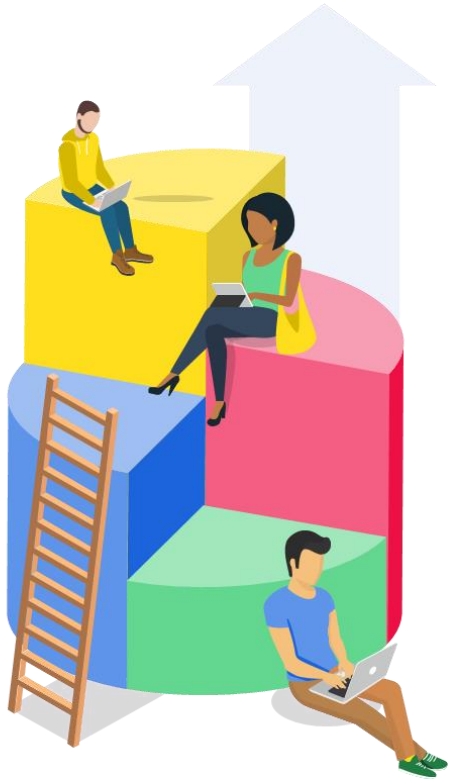
- Safety doesn't happen by chance - it must be designed in from the start through a systematic engineering process

Static program analysis

What: Examining source code for errors without executing it
How: Parsing code to detect common fault patterns
Why: Cheaply find faults that may be missed by tests
When: Implementation stage, as supplement to reviews and testing
Ex: Detecting null pointer risks in medical device code

Formal verification

What: Mathematically proving specs and code meet properties
How: Transforming formal specs to program code, verifying consistency
Why: Uncover specification errors, guarantee code matches spec
When: Requirements and design stages
Ex: Proving aircraft landing gear deploys before landing



Model checking

What: Exhaustively exploring system model for property violations
How: Defining model and properties, using automated checker
Why: Find subtle concurrency errors hard to test
When: Design stage, for small to medium critical systems
Ex: Ensuring trains never routed onto same track

Safety assurance processes

What: Activities to check system will operate safely
How: Hazard analysis & monitoring, safety reviews, safety certification
Why: Ensure safety considered throughout development
When: At all stages, from requirements to deployment
Ex: Analysing insulin pump risks, reviewing software, certifying before use

Safety assurance processes

Measurement

- Activities that check system will operate safely
  - Hazard analysis & monitoring
  - Safety reviews
  - Safety certification
- Performed at all stages, from requirements to deployment
- Goal is to ensure safety is considered via development
- Generates documentation for traceability & regulation

12.4 Safety case

Concept

- Safety and dependability cases are structured documents that set out detailed arguments and evidence that a required level of safety or dependability has been achieved

- Regulators and developers work together and negotiate what needs to be included in a system safety/dependability case

- A safety case is:
  - A documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment

Key elements of safety case

Concept

Description of system, operating context, assumptions

Hazard analysis results showing risks identified and controlled

Details of processes followed, and standards complied with

A software safety case is usually part of a wider system safety case that takes hardware and operational issues into account

Verification and validation results demonstrating safety

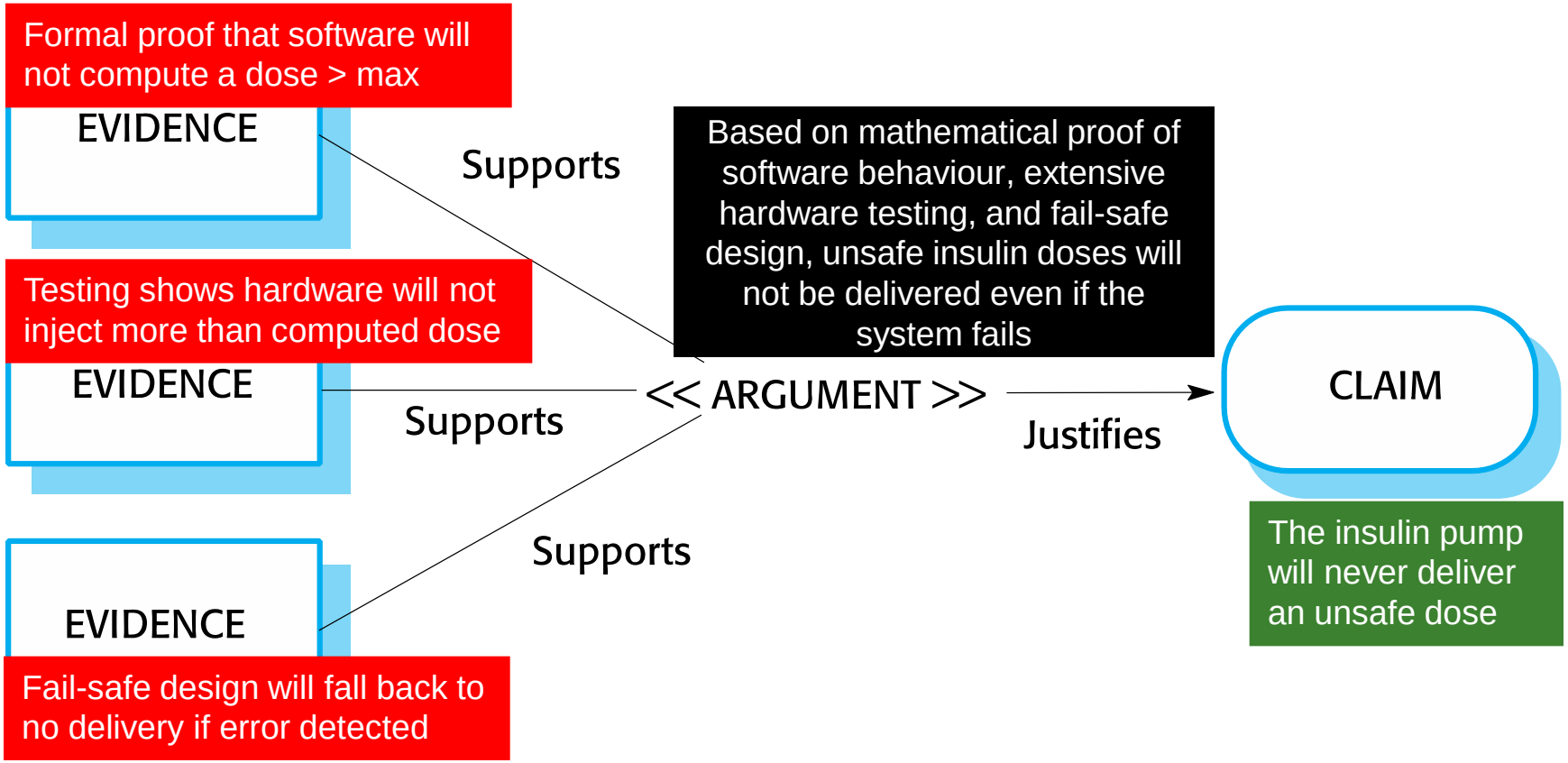
Justification that residual risk is as low as reasonably practicable

Arguments in a safety case can be based on formal proof, design rationale, safety proofs, etc. Process factors may also be included

Structured arguments- example

Implementation

- Structured arguments link evidence to safety claims



Safety implementation code

Implementation

```java
public class SafetyControlSystem {
    private static final double MAX_SAFE_TEMP = 100.0;

    // SR1: Temperature bounds checking
    public void validateTemperature(double temp) {
        if (temp > MAX_SAFE_TEMP) {
            emergencyShutdown();
        }
    }

    // SR2: Redundant sensors
    public double getSafeReading(double[] sensors) {
        return validateReadings(sensors) ?
               getAverageReading(sensors) :
               failSafeValue();
    }

    // SR3: Error recovery
    private void emergencyShutdown() {
        mainPower.shutdown();
        backupSystem.activate();
        notifyOperators();
    }
}
```

Safety engineering in AI era

Trend

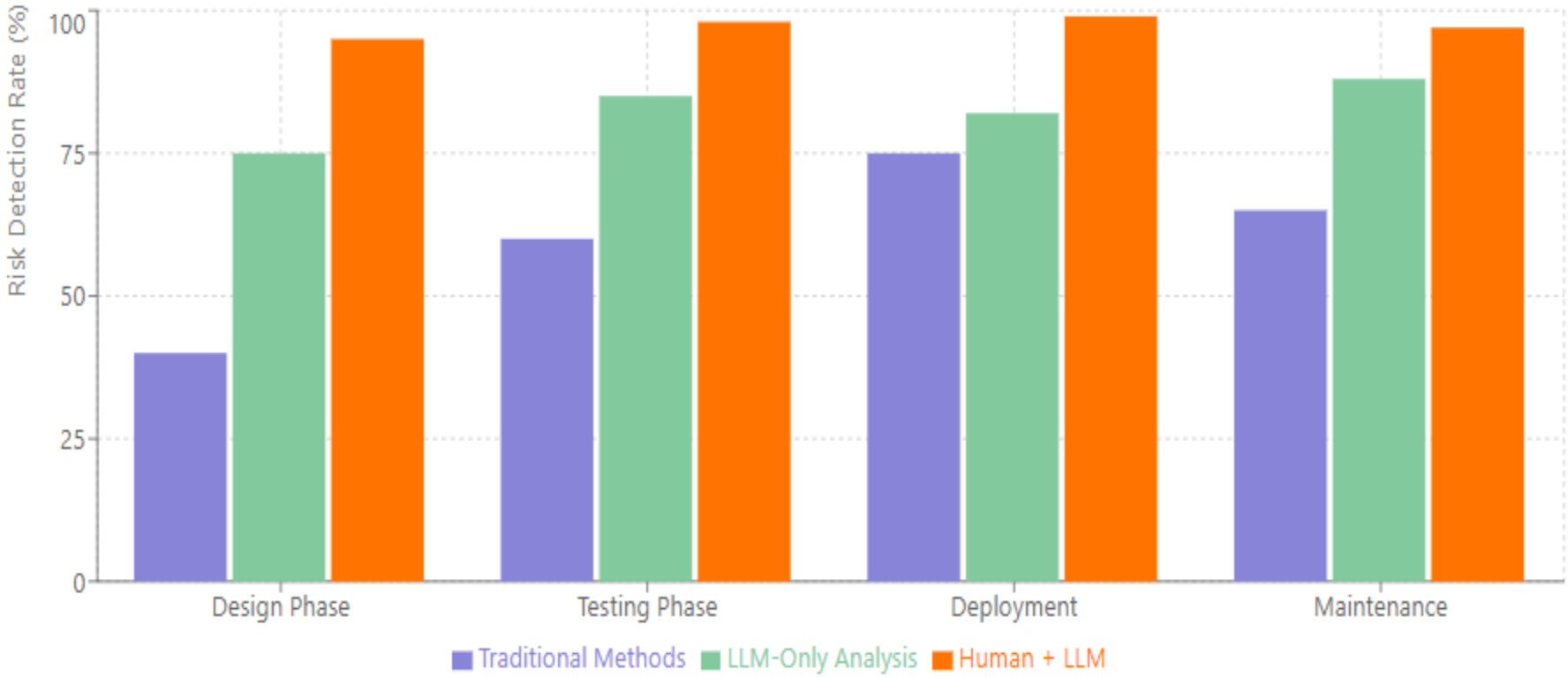
Analysis Enhancement:
- Automated hazard identification
- Pattern recognition in incident data
- Real-time safety monitoring

Documentation Support:
- Safety case generation
- Requirements validation
- Regulatory compliance checking

Risk Management:
- Predictive risk assessment
- Safety verification
- Mitigation strategy suggestion

Ex; AI safety detection rate

Measurement



Comparison of safety risk detection rates across different development phases using traditional methods, LLM-only analysis, and combined human-LLM approach

To master today's lesson

- Assignment 3 is designed to assess your mastery of fourth (CLO): Design safety in safety critical systems
- Total Marks: 5
- Submission Deadline: 8:00 PM every Saturday. No late submissions accepted
- You can use any LLM (ChatGPT, Claude, etc.), but you must follow the IT department's LLM ethics guidelines
- Collaboration allowed but submit individual work