Engineering Safety- and Security-Related Requirements for Software-Intensive Systems

ICCBSS'2007 Conference Tutorial

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Donald Firesmith
27 February 2007

Tutorial Goals

Familiarize Members of:

- ***Safety*** **and Security Teams with:**
  - **Foundations of *Requirements Engineering***
  - ***Common Concepts and Techniques from Both Disciplines***
- ***Requirements*** **Teams with the Foundations of:**
  - ***Safety Engineering***
  - ***Security Engineering***

Familiarize Members of all three Disciplines with:

- **Different Types of *Safety- and Security-related Requirements***
- **Common *Process* for Engineering these Requirements**

Contents

Challenges:

Combining Requirements, Safety, and Security Engineering

Challenges$_1$

Requirements Engineering, Safety Engineering, and Security Engineering:

- Different *Communities*
- Different *Disciplines* with different Training, Books, Journals, and Conferences
- Different *Professions* with different Job Titles
- Different fundamental underlying *Concepts* and Terminologies
- Different *Tasks, Techniques,* and *Tools*

Safety and Security Engineering are:

- Typically treated as Specialty Engineering Disciplines
- Performed separately and largely independently of the primary Engineering Workflow (Requirements, Architecture, Design, Implementation, Integration, Testing.

Challenges$_2$

Current separate Processes for Requirements, Safety, and Security are Inefficient and Ineffective.

Separation of Requirements Engineering, Safety Engineering, and Security Engineering:

- Causes *poor* Safety- and Security-related Requirements.
  - Goals rather than Requirements
  - Vague, unverifiable, unfeasible, architectural and design constraints
- Inadequate and too late to drive architecture and testing
- Difficult to achieve Certification and Accreditation

Challenges$_3$

Poor requirements are a primary cause of more than half of all project failures (defined in terms of):

- Major Cost Overruns
- Major Schedule Overruns
- Major Functionality not delivered
- Cancelled Projects
- Delivered Systems that are never used

Poor Requirements are a major Root Cause of many (or most) Accidents involving Software-Intensive Systems.

Security ‘Requirements’ often mandated:

- Industry Best Practices
- Security Functions

Challenges$_4$

How Safe and Secure is Safe and Secure *enough*?

Situation Cries out for Process Improvement:

- Better Consistency between Safety and Security Engineering
  - More consistent Concepts and Terminology
  - Reuse of Techniques
  - Less Unnecessary Overlap and Avoidance of Redundant Work
- Better Collaboration:
  - Between Safety and Security Engineering
  - With Requirements Engineering
- Better Safety- and Security-related Requirements

Three Related Disciplines

Safety Engineering

> the engineering discipline within systems engineering concerned with lowering the risk of *unintentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to such harm, mishaps (i.e., accidents and incidents), hazards, and safety risks

Security Engineering

> the engineering discipline within systems engineering concerned with lowering the risk of *intentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to such harm, misuses (i.e., attacks and incidents), threats, and security risks

Requirements Engineering

> the engineering discipline within systems/software engineering concerned with identifying, analyzing, reusing, specifying, managing, verifying, and validating goals and requirements (including safety- and security-related requirements)

Common Example:

An Automated People Mover System

Desired Characteristics

Common Ongoing Example throughout the Tutorial

Should Not Need Special Domain Knowledge

Example System should be:

- Safety-Critical
- Realistic
- SW-Intensive
- Understandable in terms of:
  - Requirements
  - Technology
  - Hazards

Example Overview

Very Large New Zoo

Zoo Automated Taxi System (ZATS)

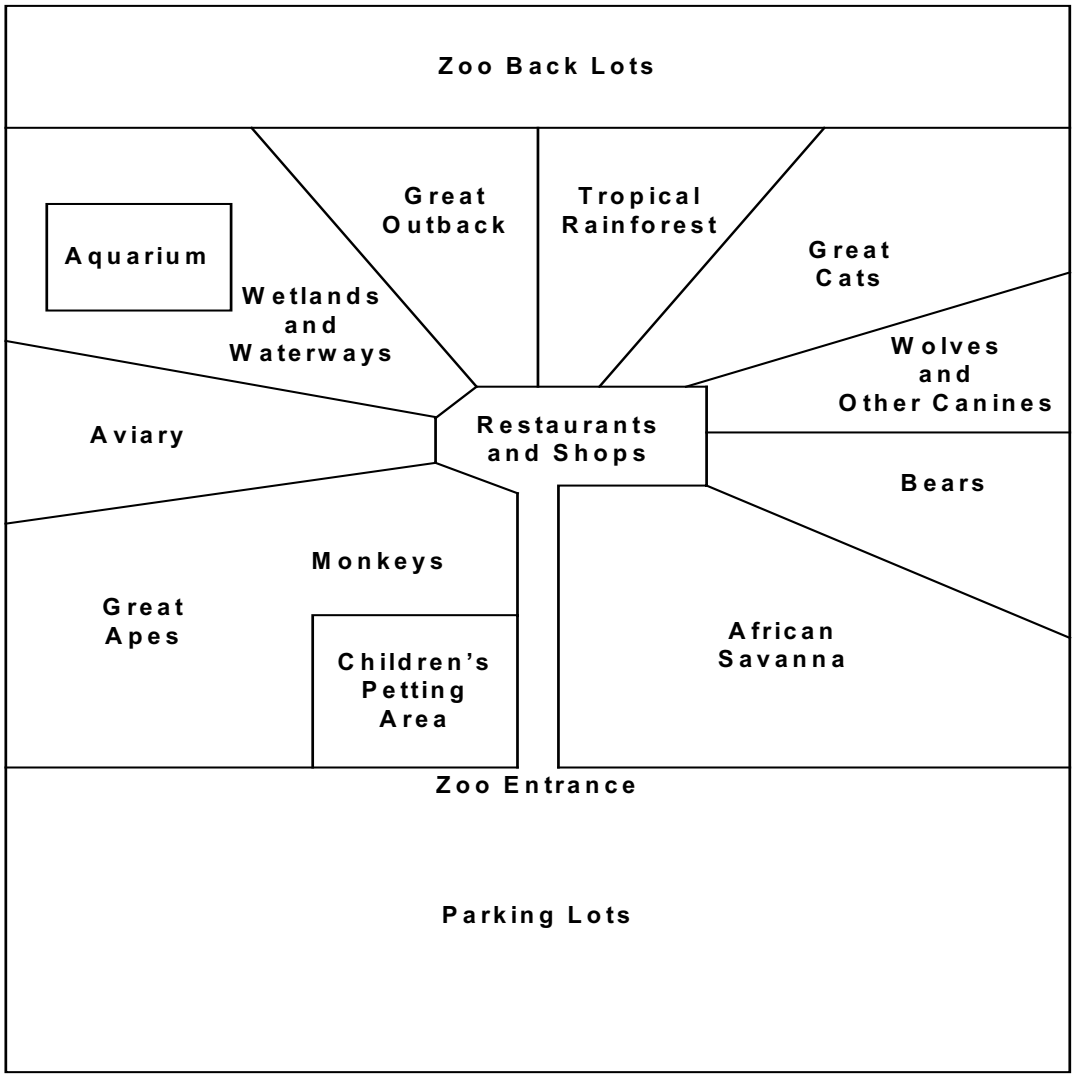
Example Zoo Habitat Guideway Layout

ZATS Context Diagram

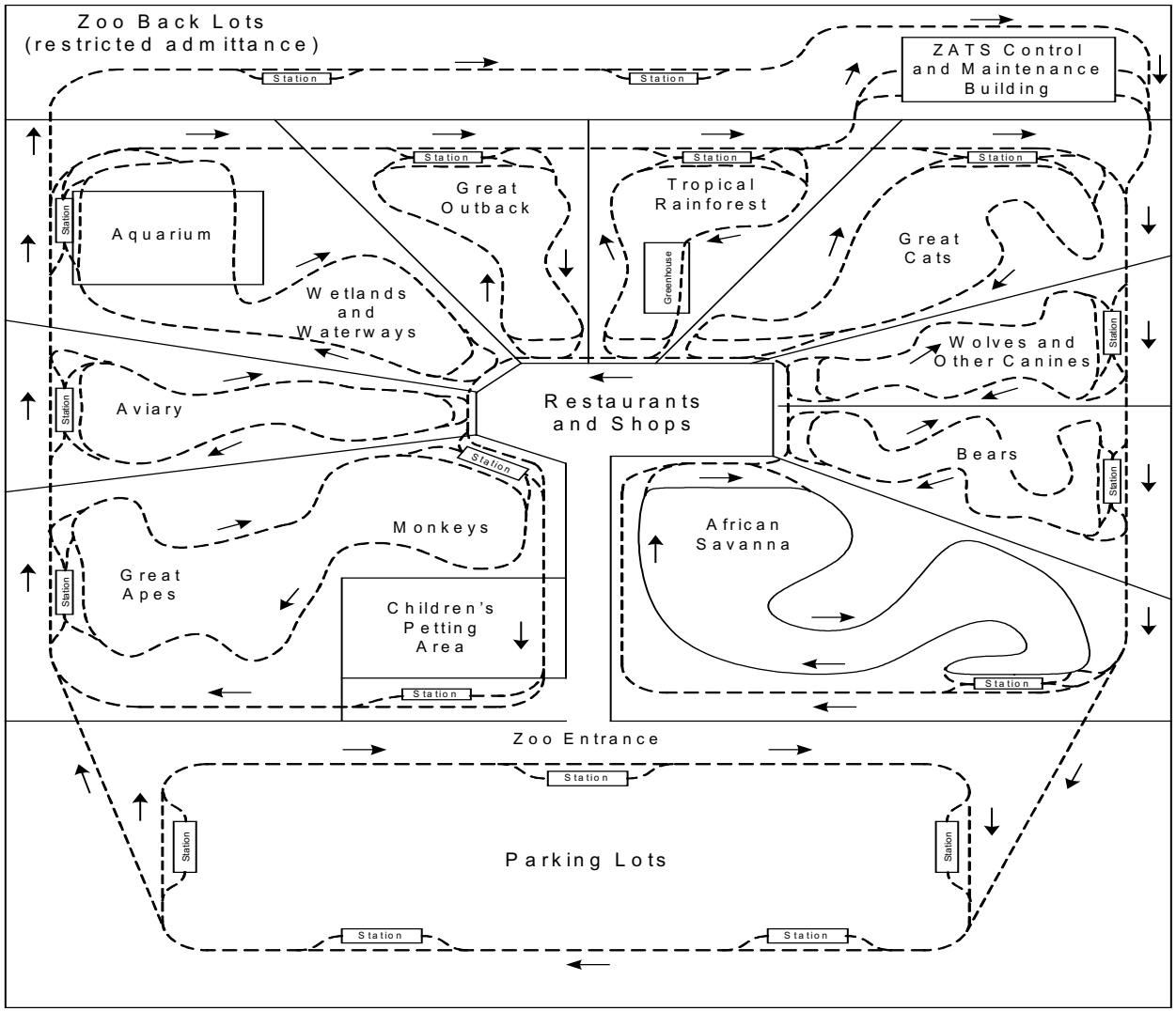
Proposed ZATS:

- Taxis
- Elevated Concrete Guideway
- Taxi Stations

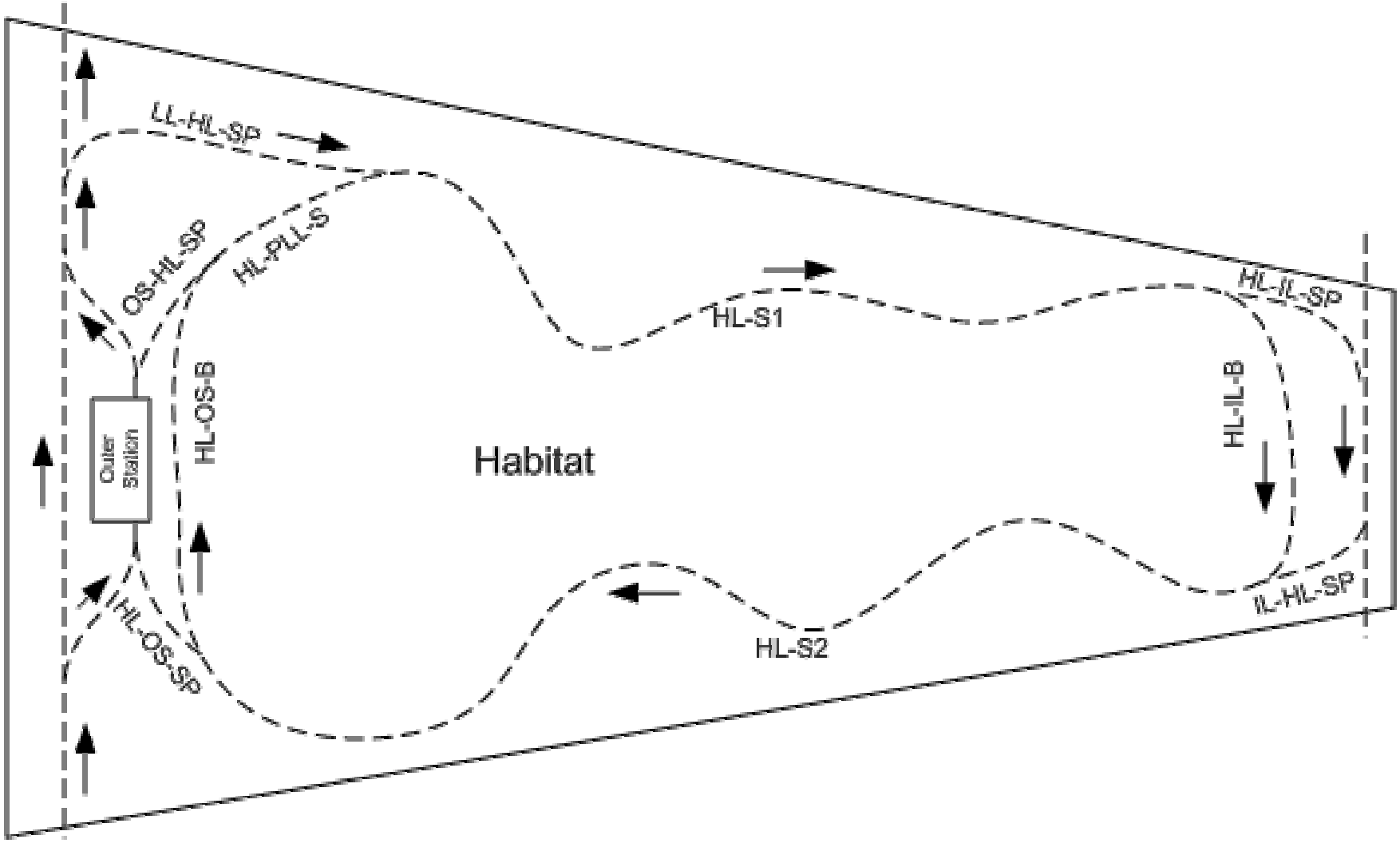
Very Large New Zoo



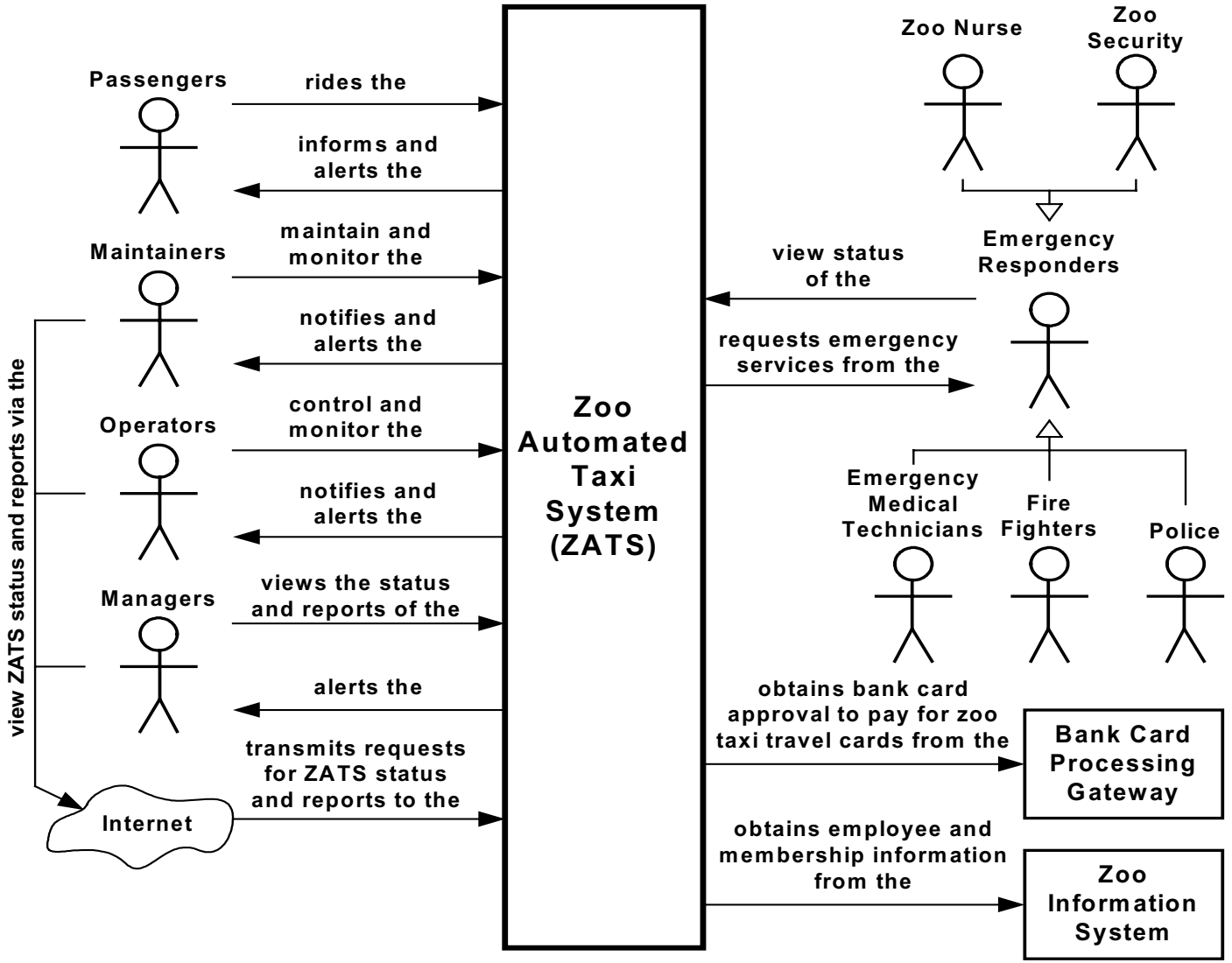
Zoo Automated Taxi System (ZATS)



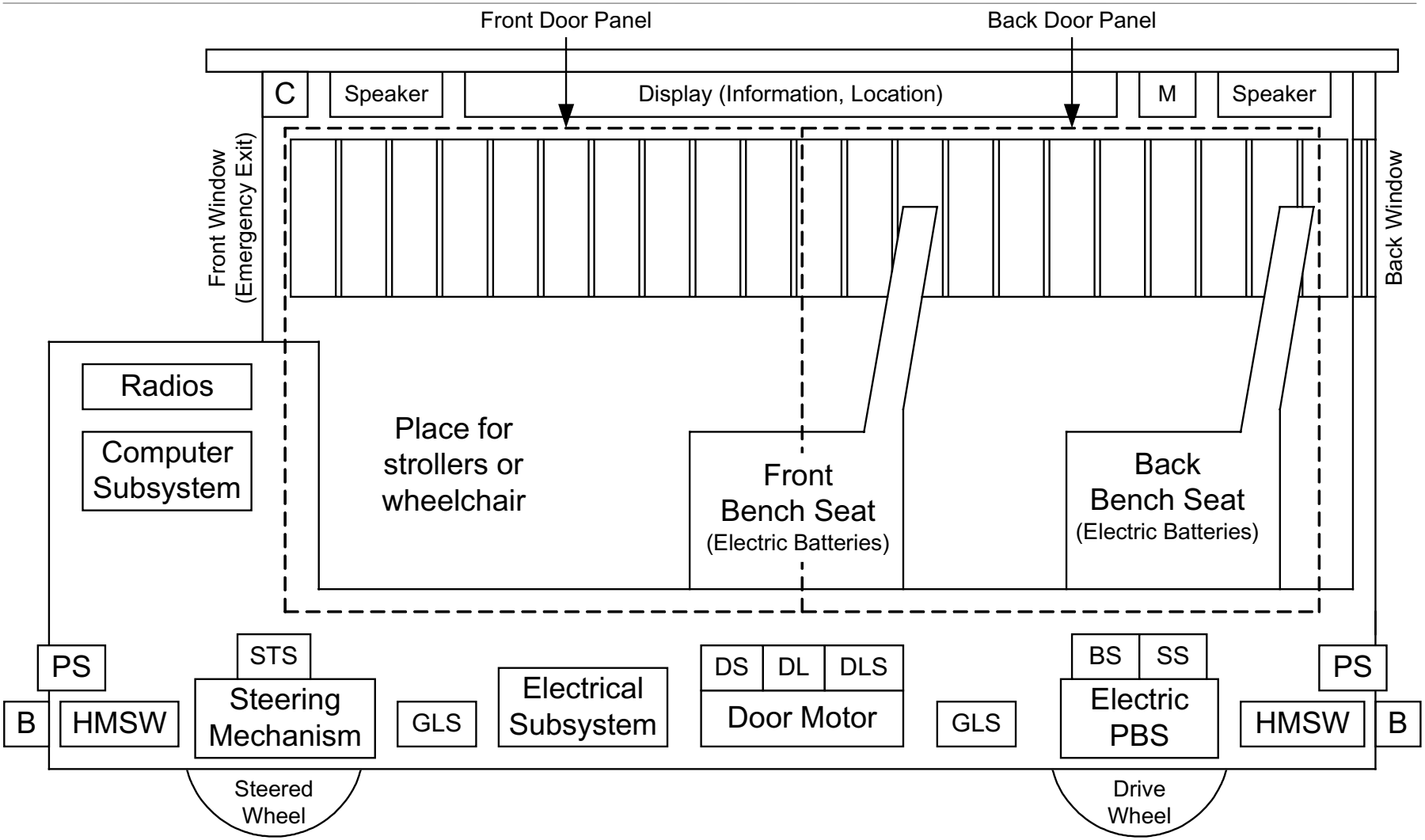
Example Habitat Layout



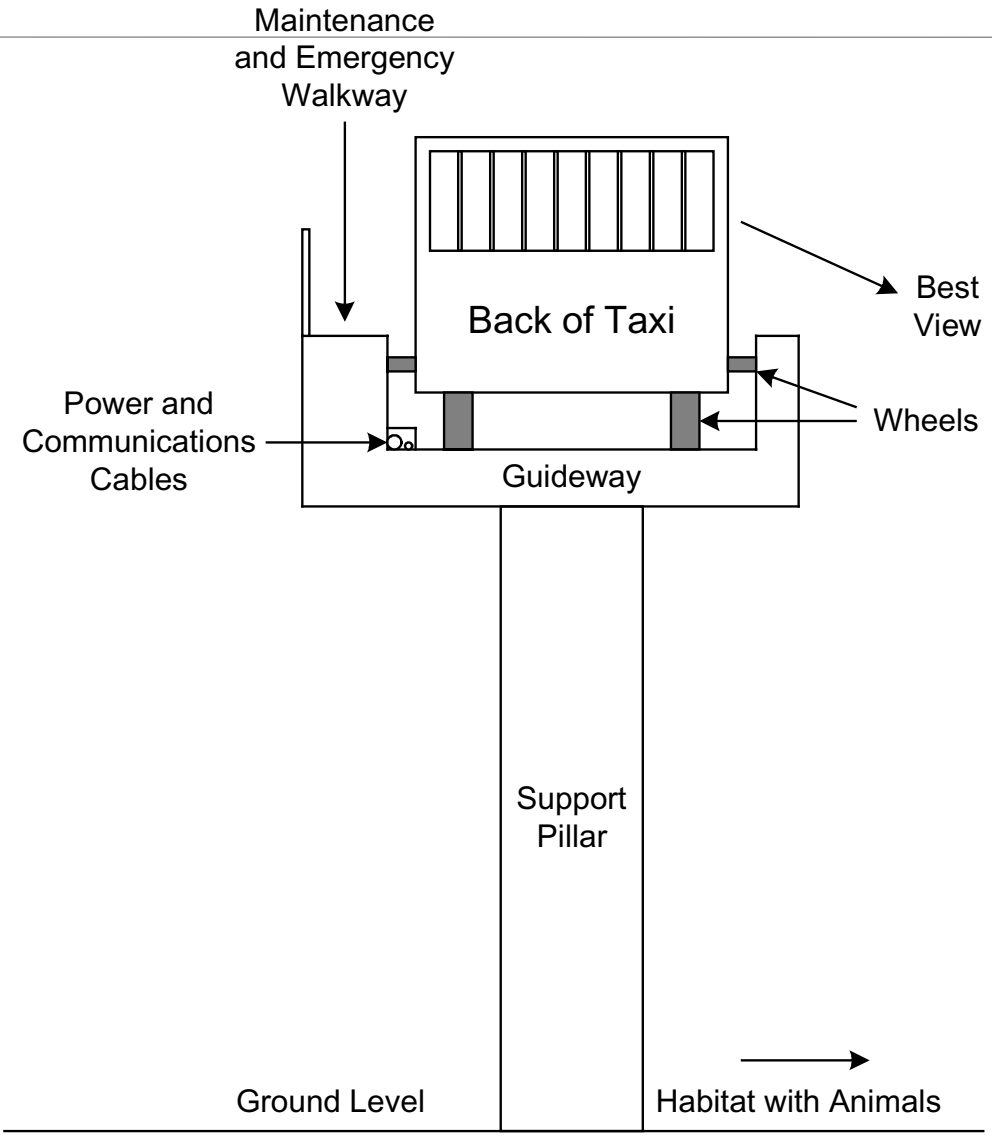
ZATS Context Diagram



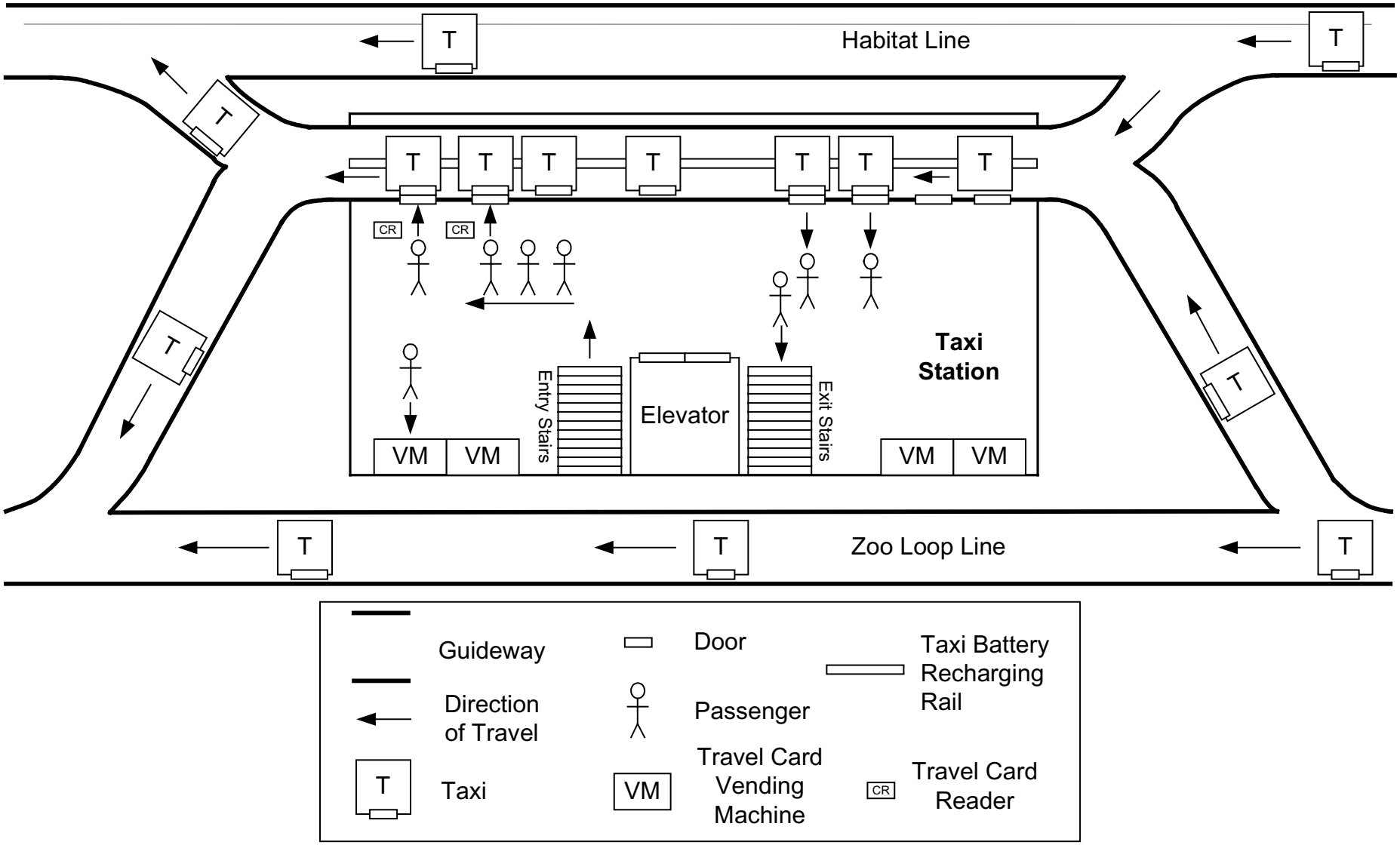
Proposed Taxi Architecture



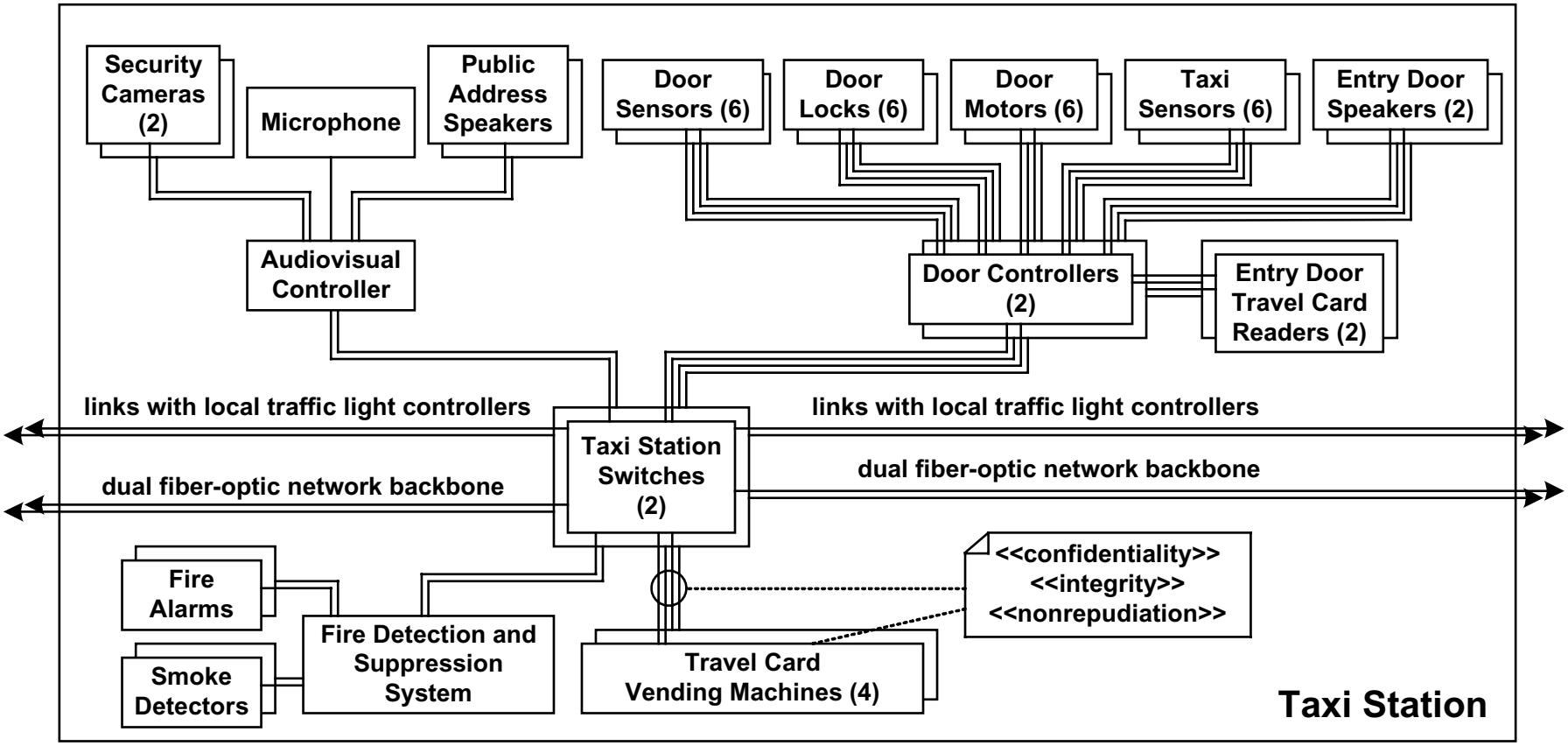
Automated Taxis On Elevated Guideways



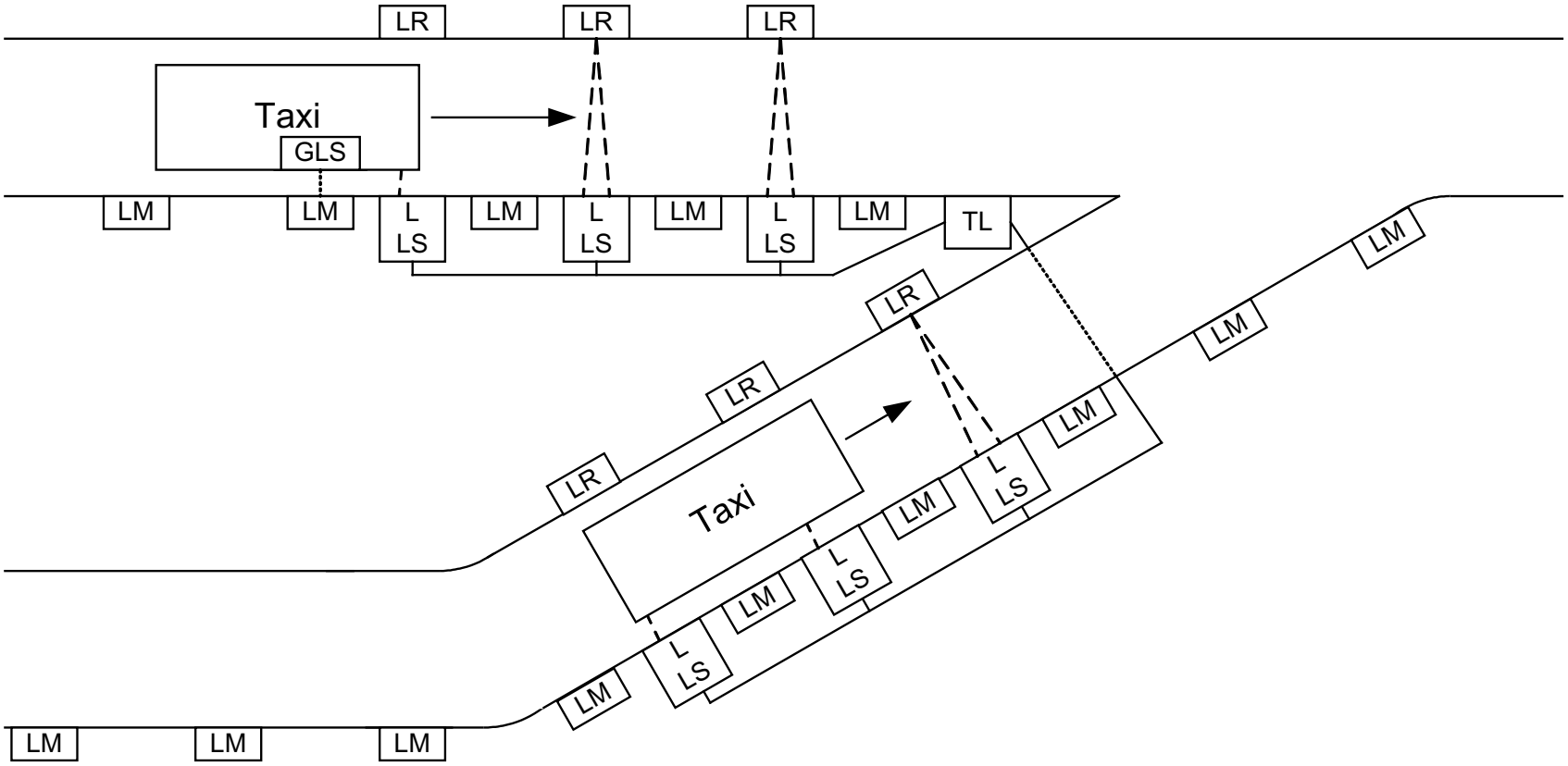
Proposed Taxi Station



Proposed Taxi Station Network Diagram



Example Collision Hazard



Requirements Engineering:
An Overview

Requirements Engineering Topics

Definition of Requirements Engineering

Requirements Engineering:

- Tasks
- Work Products

Importance and Difficulty of Requirements Engineering

Goals vs. Scenarios vs. Requirements

Types of Requirements

Characteristics of Good Requirements

Requirements Engineering

Definition

> the engineering discipline within systems/software engineering concerned with identifying, analyzing, reusing, specifying, managing, verifying, and validating goals and requirements (including safety- and security-related requirements)
>
> the cohesive collection of all *tasks* that are primarily performed to produce the *requirements* and *other related requirements work products* for an *endeavor*

Today, these RE tasks are typically performed in an *iterative*, *incremental*, *parallel,* and *time-boxed* manner rather than according to the traditional Waterfall development cycle, whereby parallel means with the:

- Primary work flow disciplines such as architecting, design, and testing
- Specialty engineering disciplines such as safety and security engineering

RE Tasks and Work Products

Business Analysis (i.e., Customer, Competitor, Market, Technology, and User Analysis as well as Stakeholder Identification and Profiling)

Visioning

Requirements Identification (a.k.a., Elicitation)

Requirements Reuse

Requirements Prototyping

Requirements Analysis

Requirements Specification

Requirements Management

Requirements Validation

Scope Management (Management)

Change Control (Configuration Management)

Quality Control (Quality Engineering)

Requirements Engineering Work Products

Business Analyses

Stakeholder Profiles

Vision Statement

- **Goals**

Operational Concept Document (OCD)

- **Usage Scenarios**

Requirements Repository and published Specifications

- **Requirements**

Requirements Prototypes

Domain Model

Glossary

Importance and Difficulty of Requirements Eng.

Poor requirements are a primary cause of more than half of all:

- Project failures (defined in terms of):
  - Major cost overruns
  - Major schedule overruns
  - Major functionality not delivered
  - Cancelled projects
  - Delivered systems that are never used
- Hazards and associated Mishaps (Accidents and Safety Incidents)
- Vulnerabilities

Difficulty of Requirements Engineering

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

F. Brooks, *No Silver Bullet*, IEEE Computer, 1987

Goals

A **goal** is an *informally documented perceived need* of a *legitimate stakeholder*.

- Goals are typically documented in a vision statement.
- Goals drive the analysis and formal specification of the requirements.
- Examples:
  - The system shall support user activity X.
  - The system shall be efficient.
  - The system shall be easy to use.
  - The system shall be safe to use.
- Goals are typically not verifiable.
- Goals may not be feasible.

Example ZATS Goals

Functional Goals:

- ZATS must rapidly transport patrons between the parking lots and the zoo.
- ZATS must rapidly transport patrons between habitats within the zoo.
- ZATS must allow patrons to take leisurely tours of the habitats.

Data Goal:

- ZATS must record and report appropriate system usage statistics.

Capacity Goal:

- ZATS must include sufficient taxis so that patrons need not wait long for a free taxi.

Usability Goal:

- ZATS must be very easy and intuitive for patrons to use, including those who are not very good with technology.

Usage Scenarios

A **usage scenario** is a specific functionally cohesive sequence of interactions between user(s), the system, and potentially other actors that provides value to a stakeholder.

Usage scenarios:

- Are instances of use cases.
- Can be either “sunny day” or “rainy day” scenarios.
- Have preconditions, triggers, and postconditions.
- Are typically documented in an Operational Concept Document (OCD).
- Drive the analysis and formal specification of the [primarily functional] requirements.
- Often include potential design information.
- Can be written in either list or paragraph form.

Example ZATS Scenario

Ride Zoo Loop Line To Restaurants for Lunch:

After the family enters a waiting taxi, Mr. Smith looks at the zoo map on its ceiling. A light representing their taxi is glowing at the Tropical Rainforest Habitat outer taxi station. He uses the control panel to select the inner taxi station at the habitat, which is the central taxi station near the restaurants and shops as a destination. He then swipes his zoo taxi debit card, and the display shows the remaining balance of $9.00 on the card. The taxi warns them to set down and thirty seconds later, the station and taxi exit doors close. Their taxi accelerates out of the taxi station and turns to the left onto the Zoo Loop Line.

Shortly after leaving the taxi station, they see a spur the angles off to their left towards a large building containing the taxi control center and maintenance facility. They continue around the outside of the zoo, passing other the Great Cats, the Wolves and Other Dogs, and the Bears habitats. Just before they reach the outer African Savanna taxi station, the guideway makes a sweeping turn to the right and they can see the parking lot on their left. Everyone looks to see if they can see the family van, but the parking lot is too big and they can only see the parking lot taxi station near where it is parked.

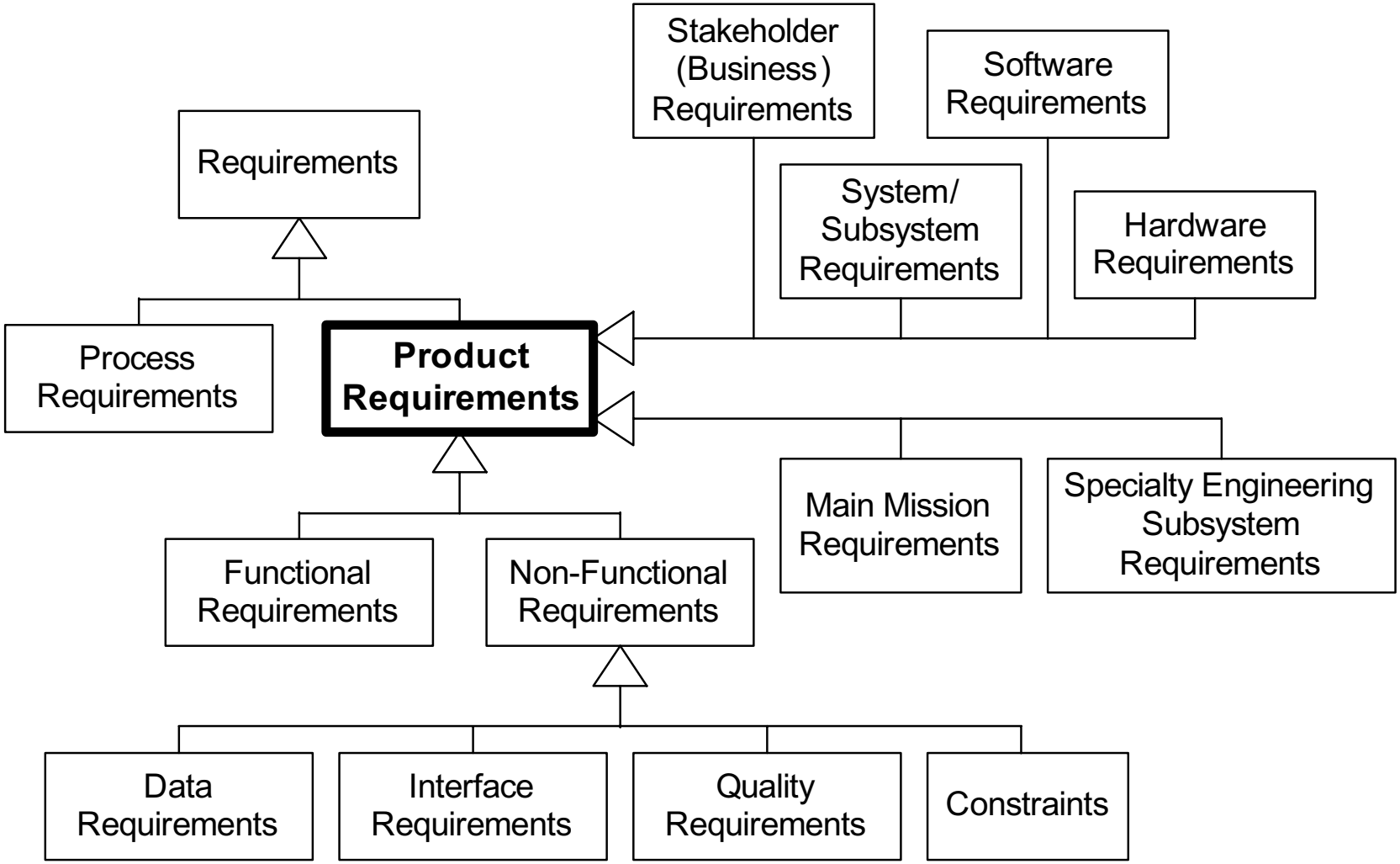
Soon, they pass the zoo entrance on their left and turn right to follow the main street to where the main restaurants and shops are. Their taxi passes the inner African Savanna taxi station on their right, circles around the central area, and soon pulls off the Zoo Loop Line to enter the inner Great Apes and Monkeys taxi station. Exiting the taxi when the doors open, they head down the elevator and outside for an early lunch at one of the many restaurants.

Requirements

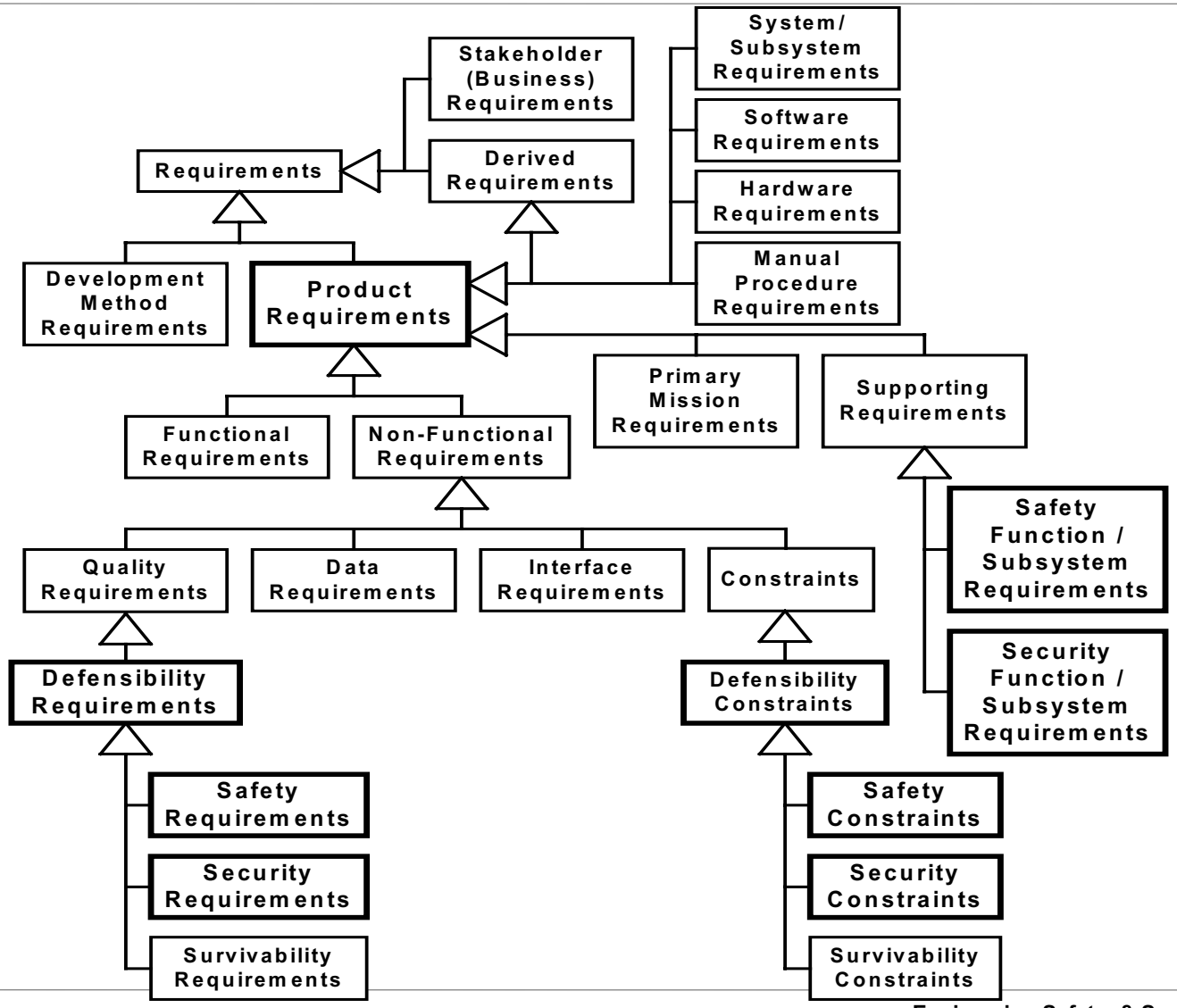
A (product) **requirement** is a *mandatory* characteristic (behavior or attribute) of a product (e.g., system, subsystem, software application, or component).

- Requirements are documented in requirements specifications.
- Requirements are driven by goals.
- Example: “At each taxi station while under normal operating conditions, ZATS shall provide a taxi to passengers within an average of 5 minutes of the passengers’ request.”
- Requirements must have certain characteristics (e.g., verifiable and feasible).

Types of Requirements



Types of Requirements



Characteristics of Good Requirements

Mandatory

Correct

Cohesive

Feasible

Relevant

Unique

Unambiguous

Validatable

Verifiable

What or How Well, not How

Complete

Consistent

Usable by Stakeholders

Uniquely Identified

Traced

Externally Observable

Stakeholder-Centric

Properly Specified

Prioritized

Scheduled

Managed

Controlled

http://www.jot.fm/issues/issue_2003_07/column7

Safety and Security Engineering:

An Overview

Similar Definitions

Safety Engineering

> the engineering discipline within systems engineering concerned with lowering the risk of *unintentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to such harm, mishaps (i.e., accidents and incidents), hazards, and safety risks

Security Engineering

> the engineering discipline within systems engineering concerned with lowering the risk of *intentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to such harm, misuses (i.e., attacks and incidents), threats, and security risks

Fundamental Concepts:

A Foundation for Understanding

Fundamental Concepts

Quality Model

Safety and Security as a Quality Factors with associated Quality Subfactors

Systems responsible for Valuable Assets

Stakeholders

Accidental and Malicious Harm to Valuable Assets

Defensibility Occurrences (Accidents, Attacks, and Incidents)

Agents (External and Internal, Malicious and Non-malicious)

Vulnerabilities (system-internal sources of dangers)

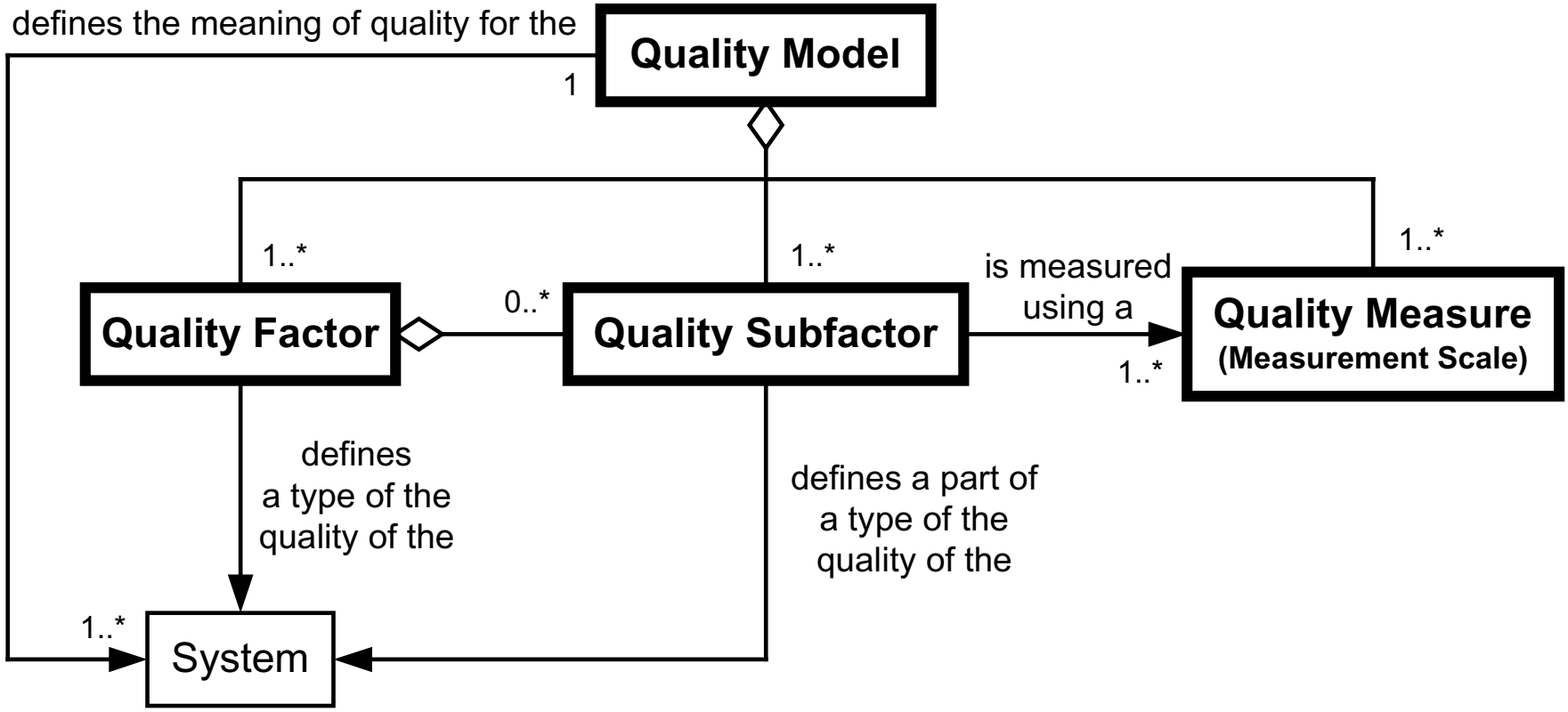
Dangers (Hazards and Threats)

Defensibility Risks (Safety and Security)

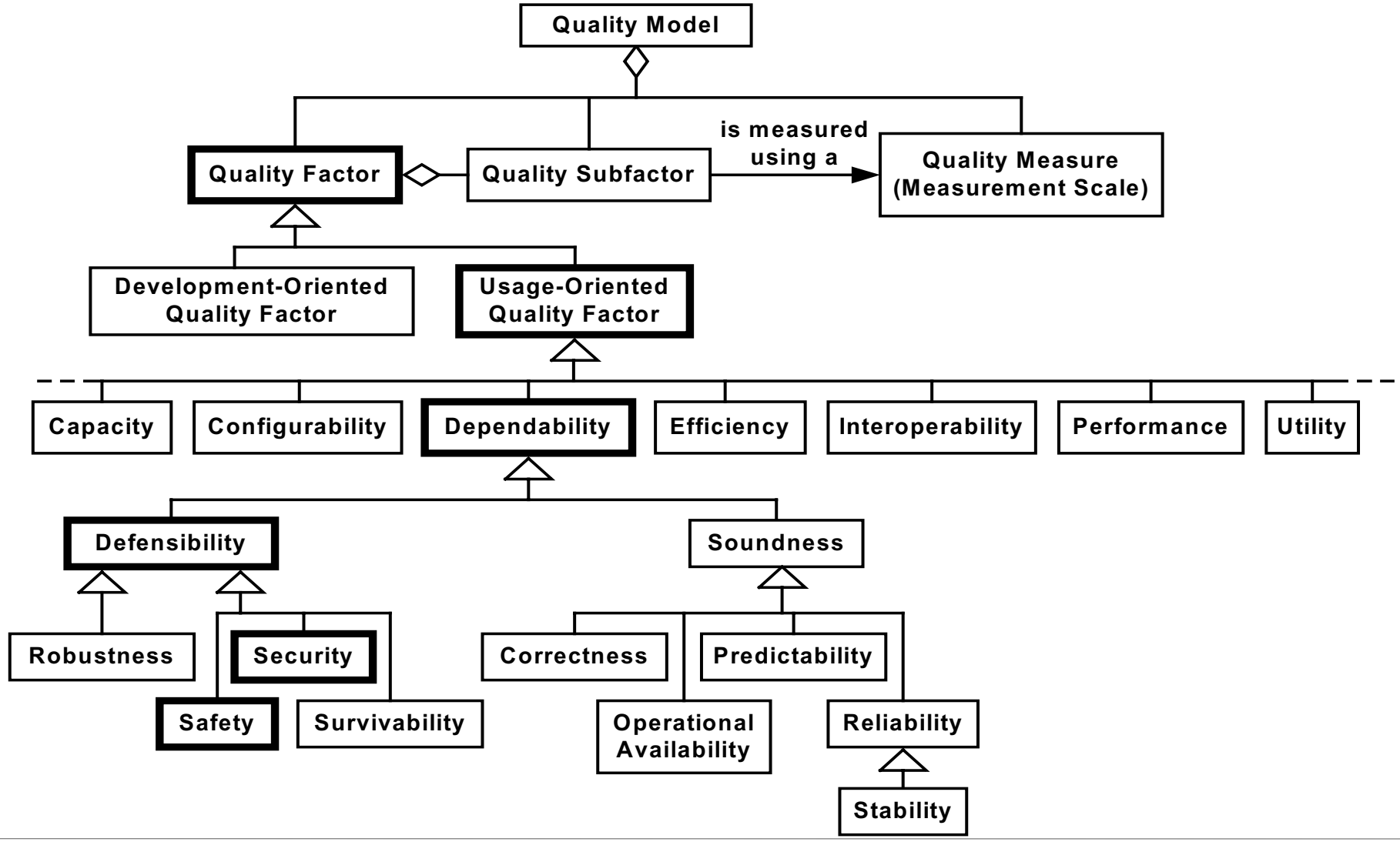
Goals, Policies, and Requirements

Defenses (Safeguards and Counter Measures)

Quality Model



Quality Factors



Safety as a Quality Factor

Safety is the Quality Factor capturing the *Degree* to which:

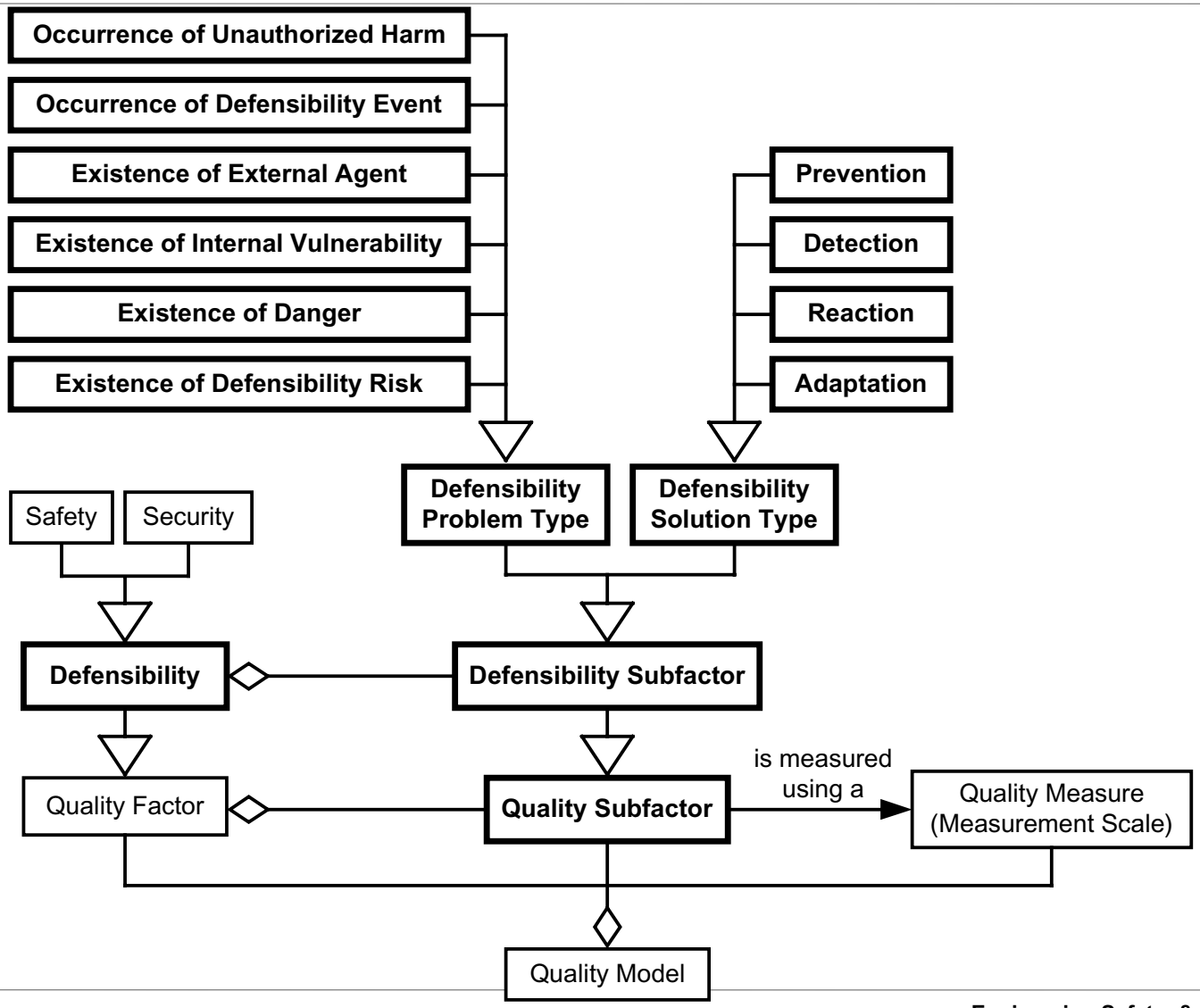
- *Accidental Harm* to Valuable Assets is eliminated or mitigated
- *Safety Occurrences and Events (Accidents, Safety Incidents,* and *Hazardous Events*) are eliminated or their negative consequence mitigated
- *Hazards* (i.e., Hazardous Conditions) are eliminated or mitigated:
  - System Vulnerabilities
  - Non-malicious Agents (humans, systems, and the environment)
- *Safety Risks* are kept acceptably low
- The preceding Problems are *Prevented*, *Detected*, *Reacted to*, and possibly *Adapted to*

Security as a Quality Factor

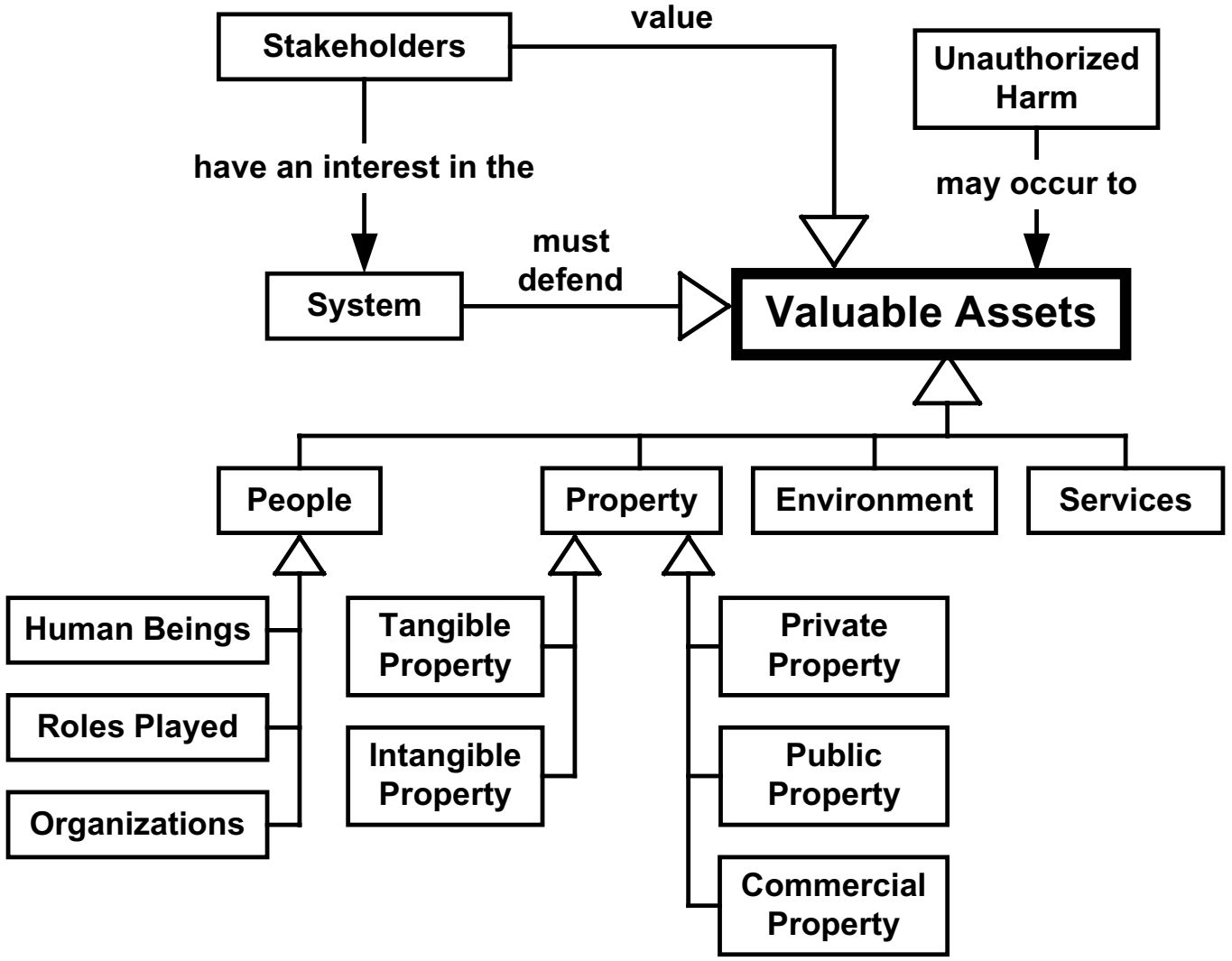
Security is the Quality Factor capturing the *Degree* to which:

- *Malicious Harm* to Valuable Assets is eliminated or mitigated
- *Security Occurrences and Events (Attacks, Security Incidents,* and *Threatening Events*) are eliminated or their negative consequence mitigated
- *Threats* (i.e., Threatening Conditions) are eliminated or mitigated:
  - System Vulnerabilities
  - Malicious Agents (humans, systems, and malware)
- *Security Risks* are kept acceptably low
- The preceding Problems are *Prevented*, *Detected*, *Reacted to*, and possibly *Adapted to*

Defensibility Quality Subfactors



Valuable Assets



Some ZATS Valuable Assets

People:

- Passengers
- Operators
- Maintainers

Property:

- Animals
- Passenger Bank Card Information
- Taxis
- Taxi Stations

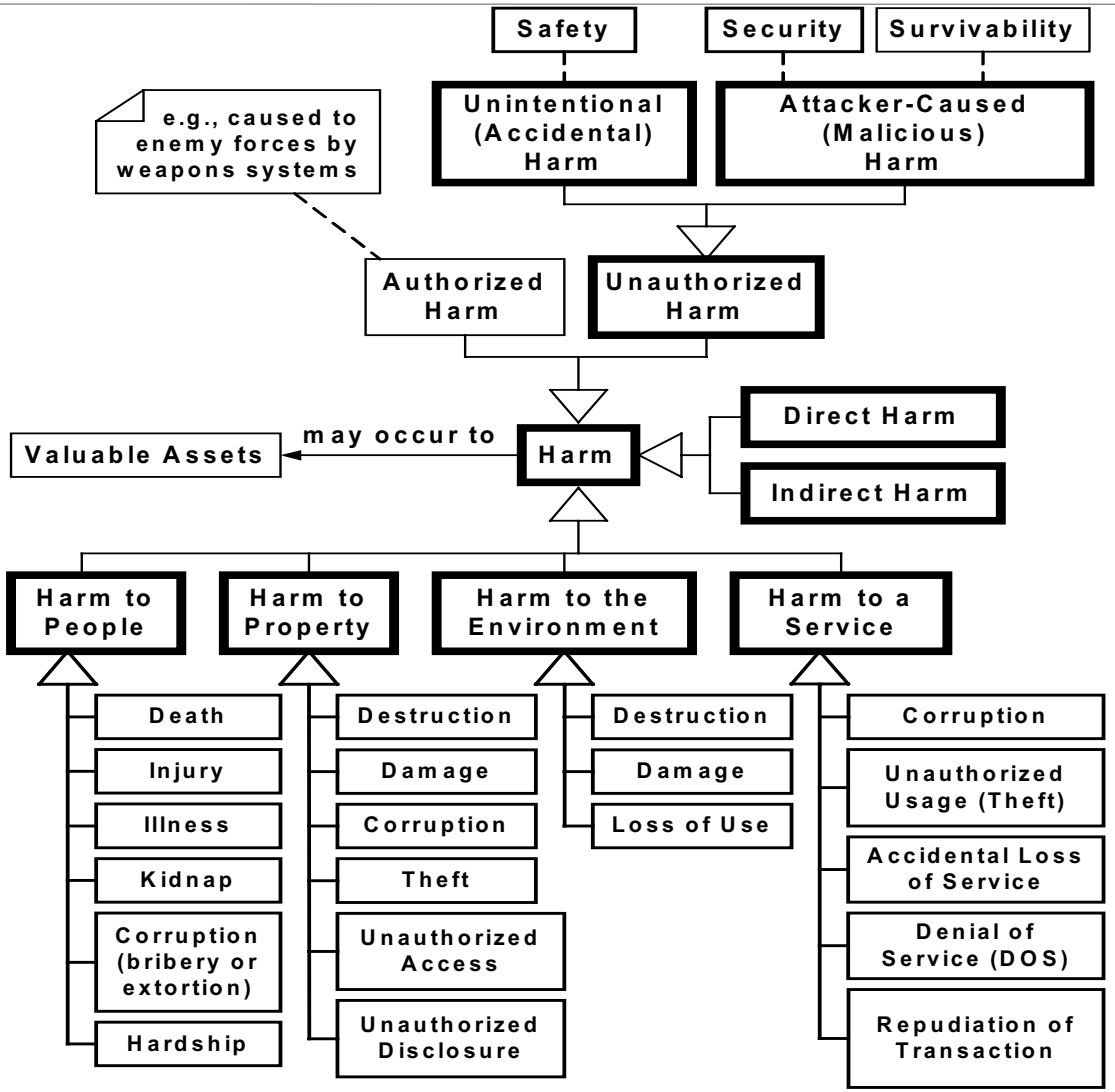
Environment:

- Habitat

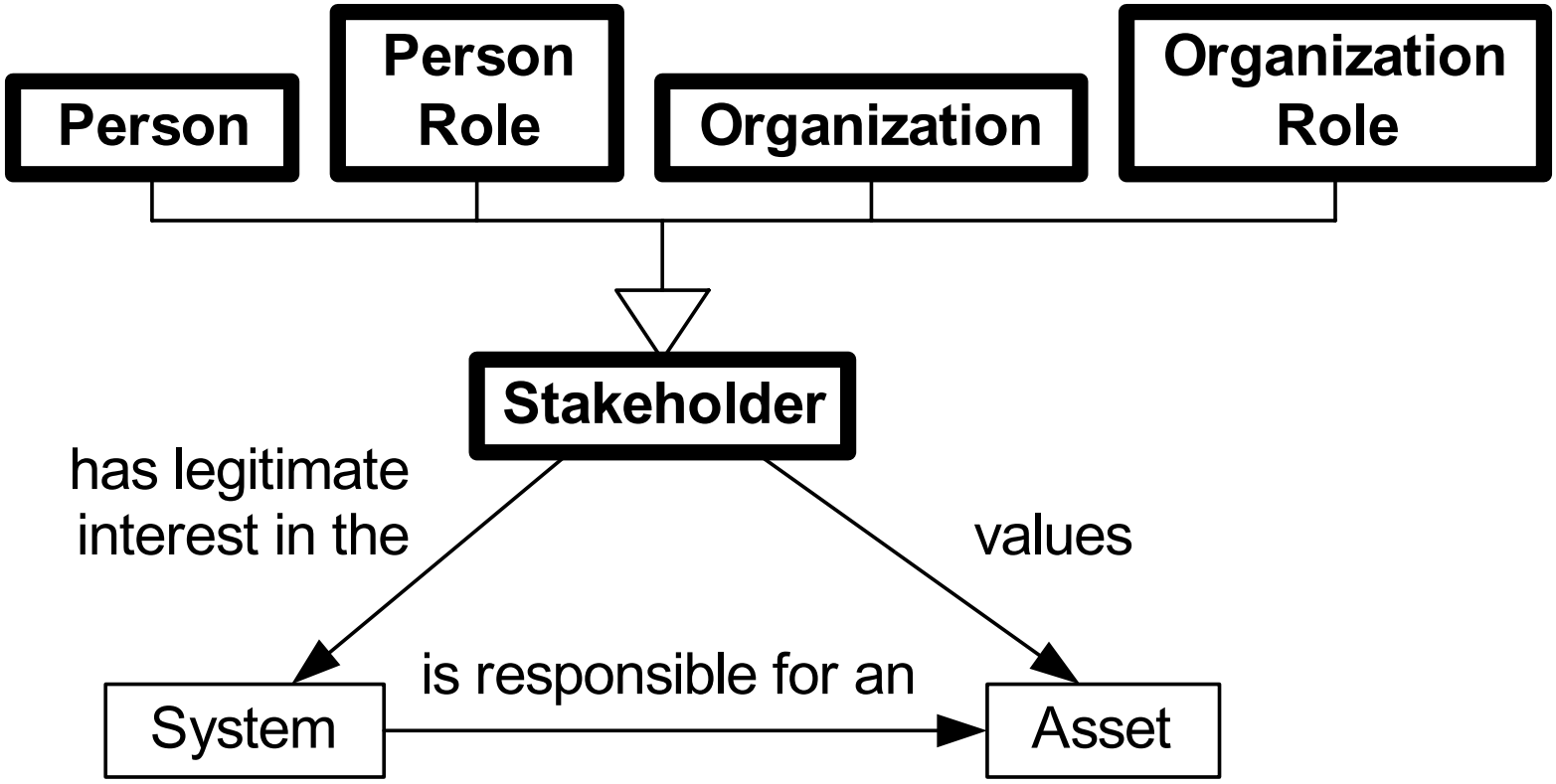
Services:

- Taxi Service

Types of Harm



Stakeholders



Some ZATS Stakeholders

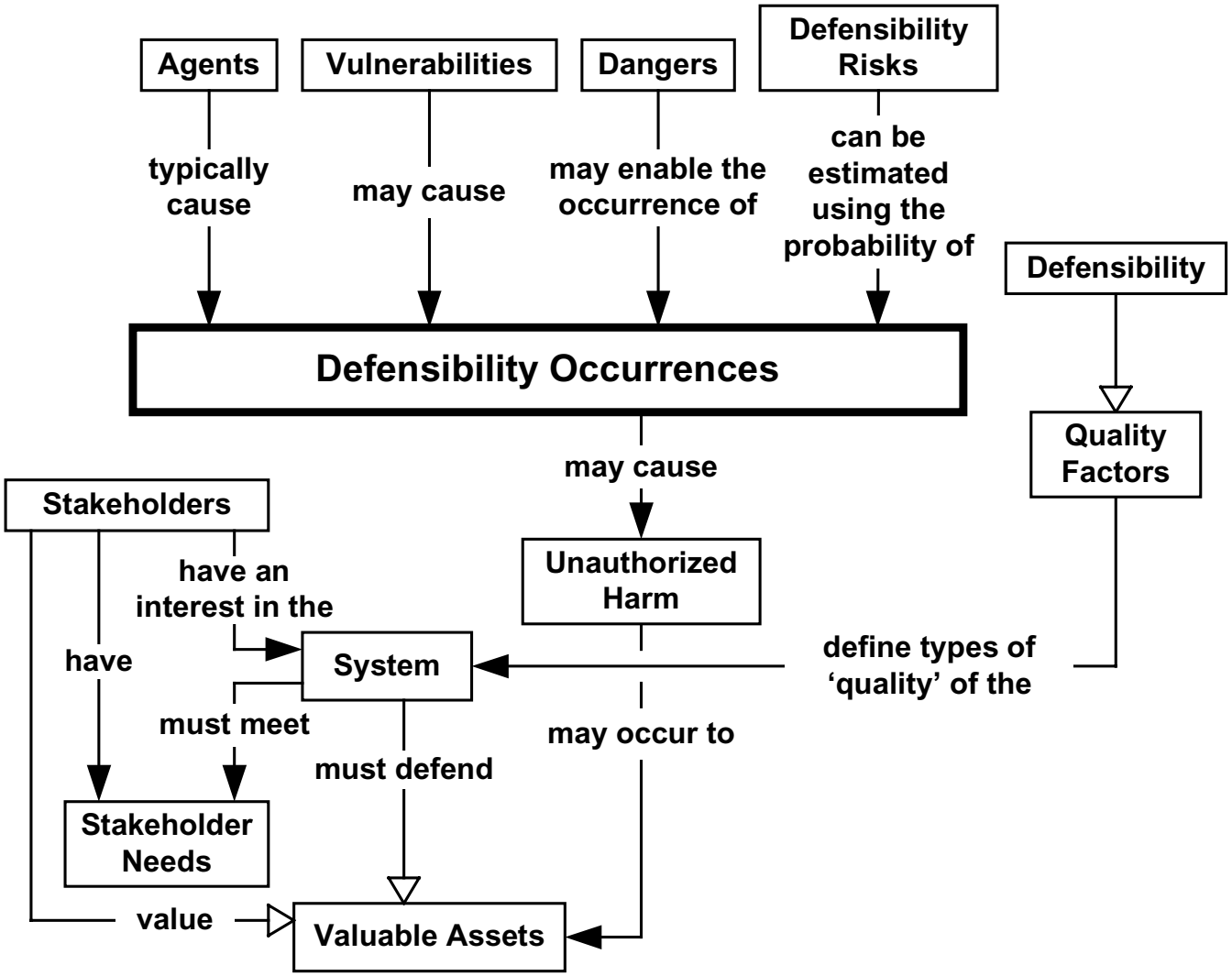
People:

- Emergency Responders
- Passengers
- Operators
- Maintainers
- ZATS Developers
- Zoo Employees
- Zoo Management

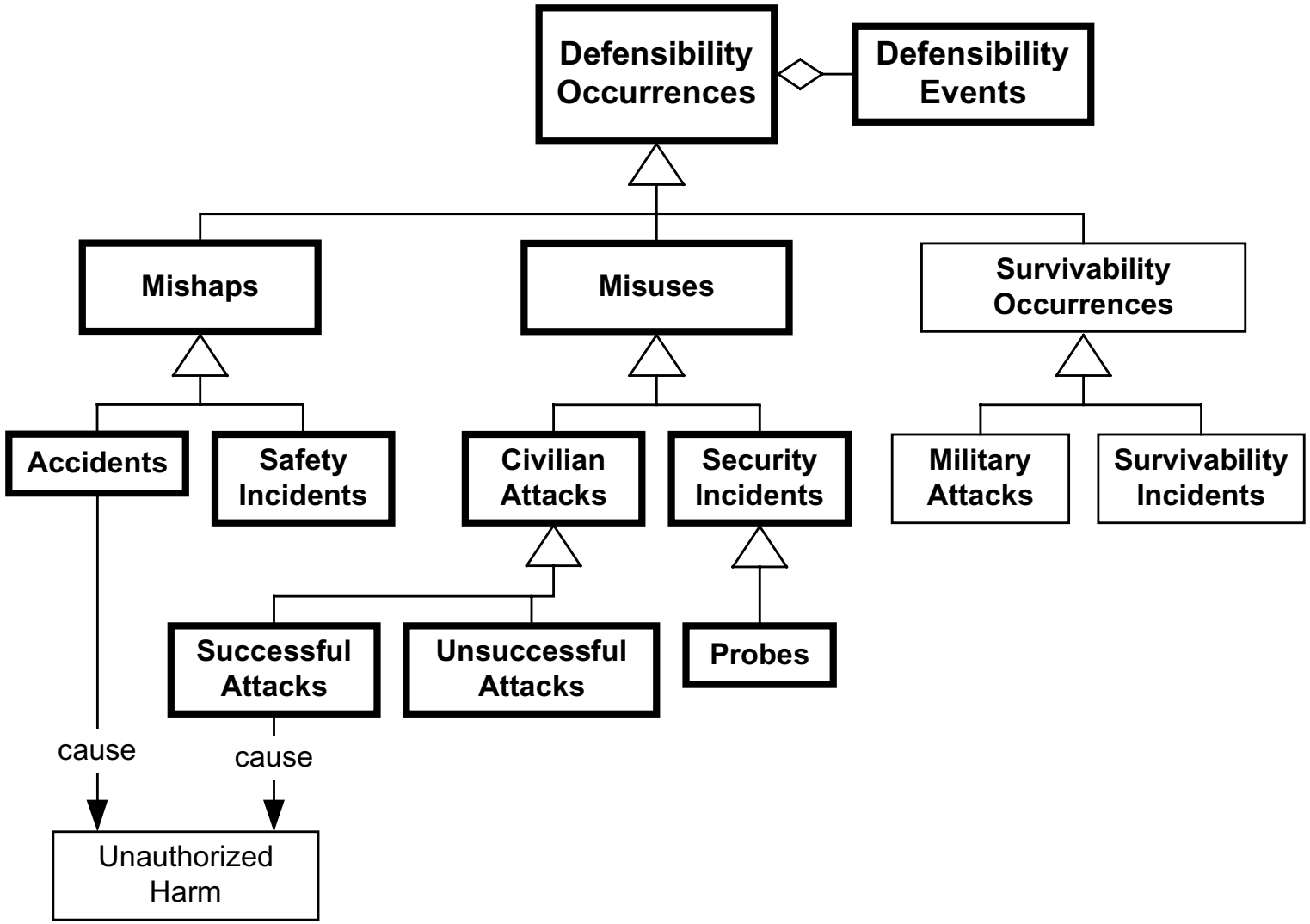
Organizations:

- Bank Card Processing Gateway
- Safety and Security Certification/Accreditation Bodies
- Zoo Regulatory Bodies

Accidents and Attacks



Types of Defense Occurrences



Example ZATS Defensibility Occurrences

Accidents:

- Natural Disasters
- Taxi Accidents
- Taxi Station Accidents

Safety Incidents:

- Inadequate Headway
- Overspeed

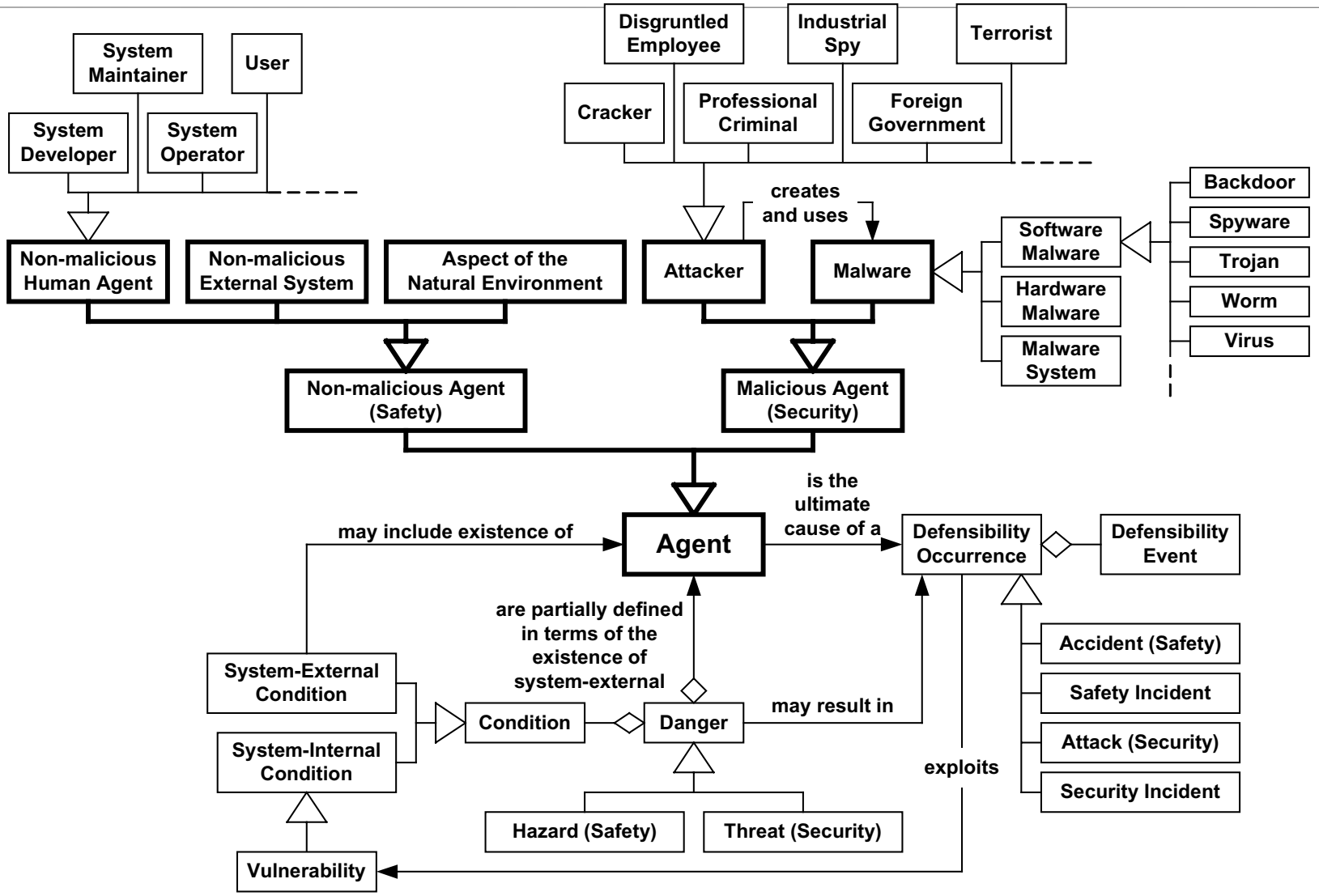
Attacks:

- Arson
- Cyber-attacks

Security Incidents:

- Antivirus Software Works

Agents



Example ZATS Agents

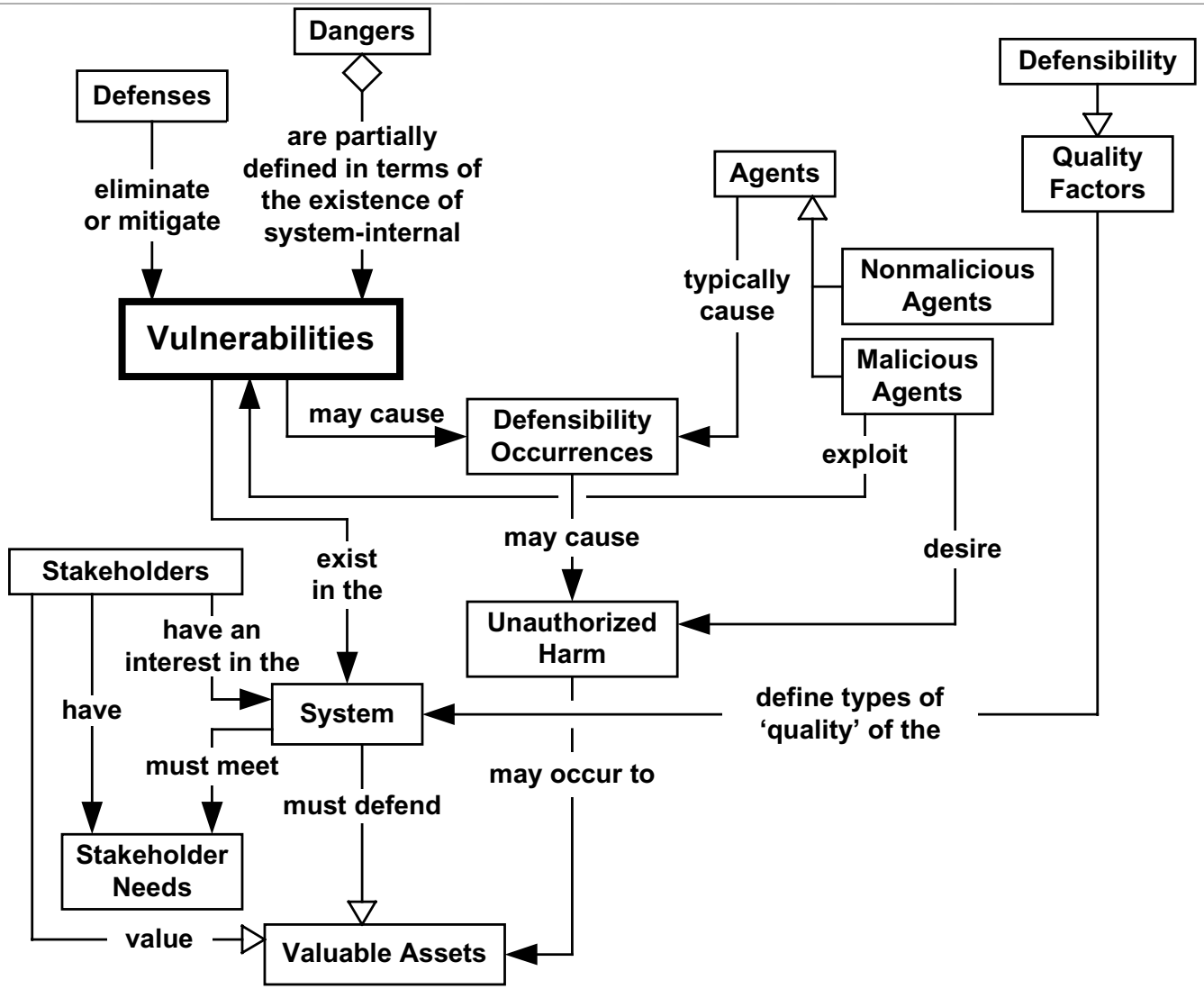
Non-Malicious Agents:

- Human Agents (e.g., Developer, Maintainer, Operator, Passenger)
- External Systems (e.g., Communications Network, Electrical Power Grid)
- Natural Environment (e.g., River or Weather)

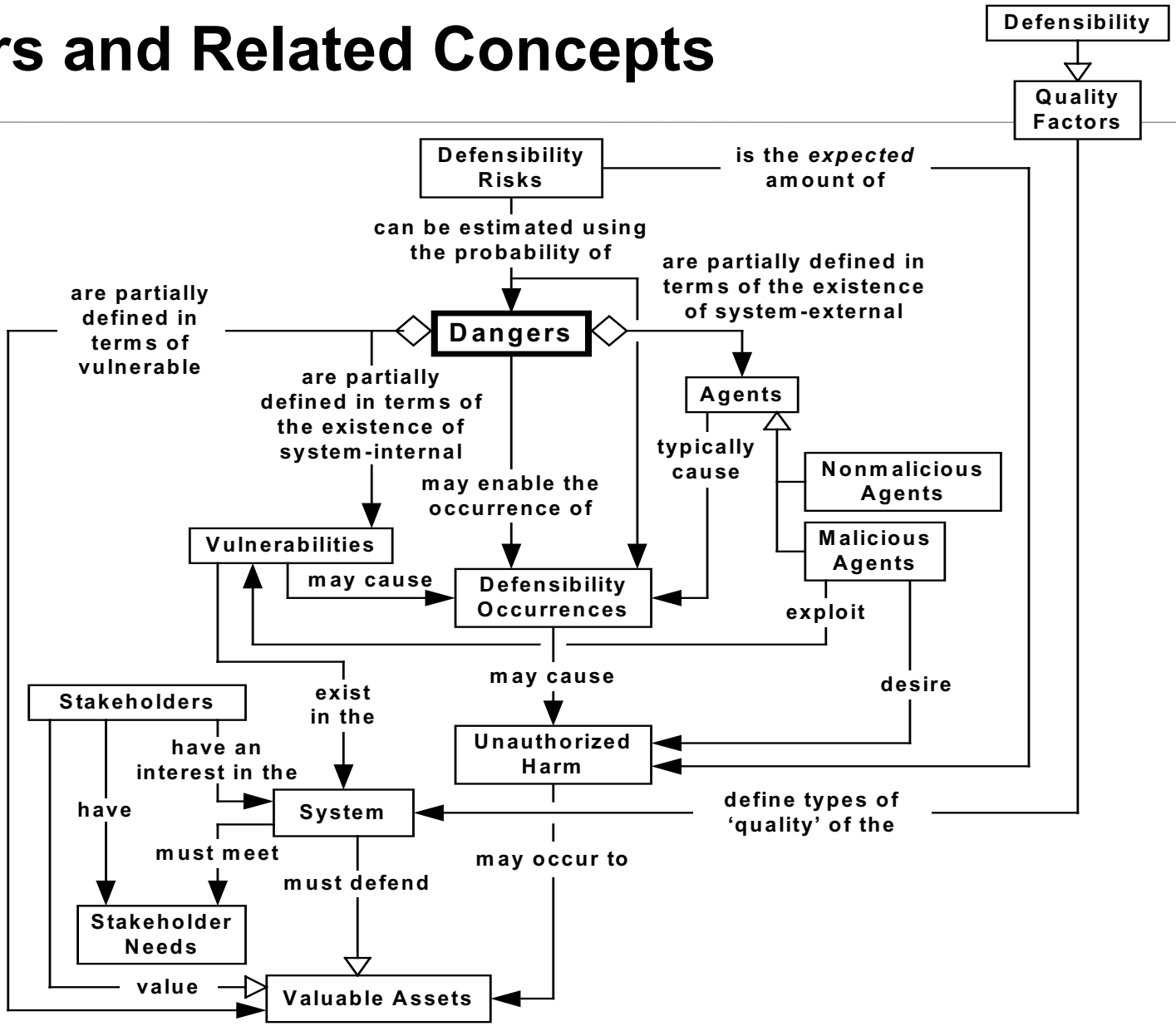
Malicious Agents:

- Attackers (e.g., Arsonists, Crackers, Terrorists, Thieves)
- Malware (e.g., virus, Trojan horse, Worm)

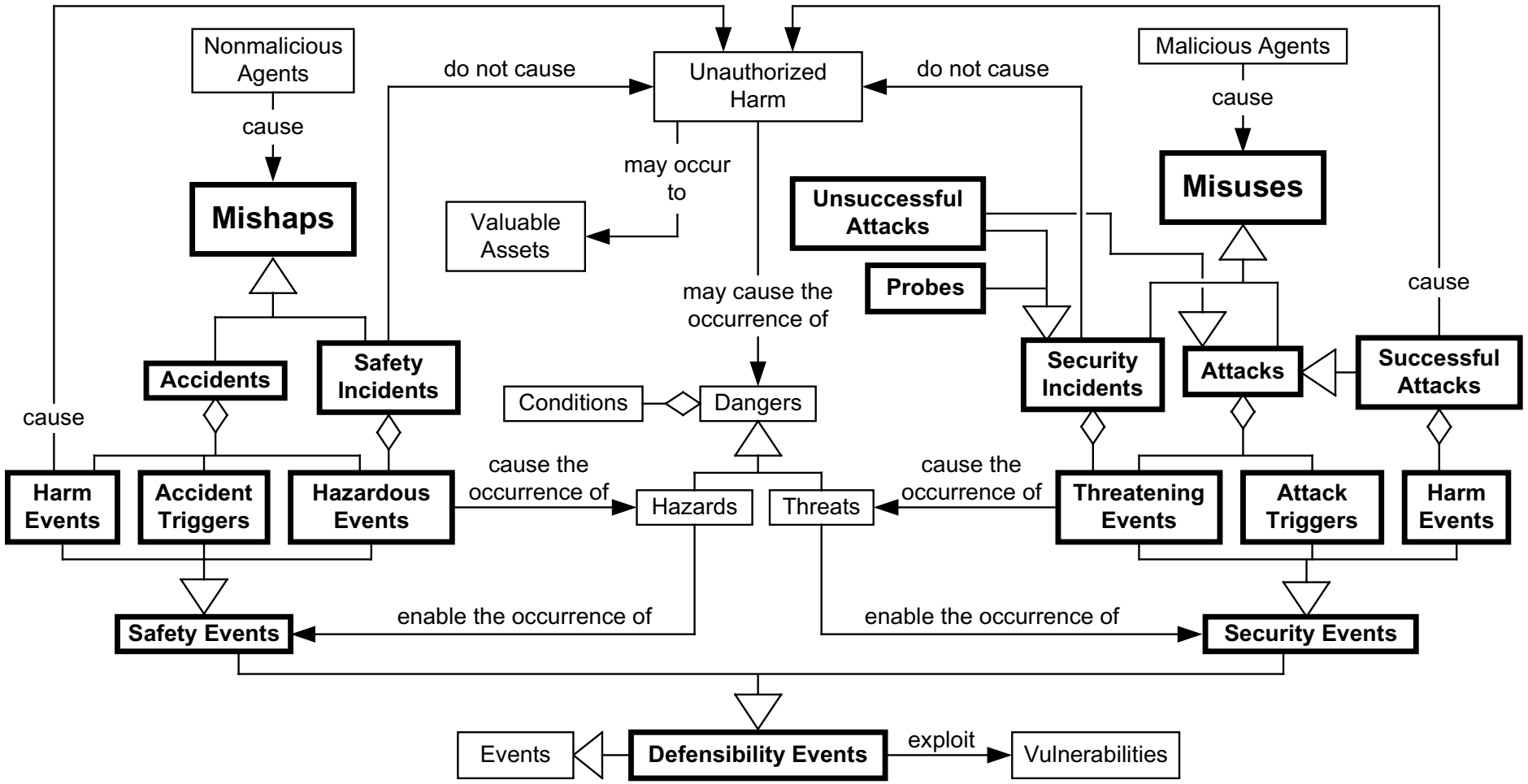
Vulnerabilities



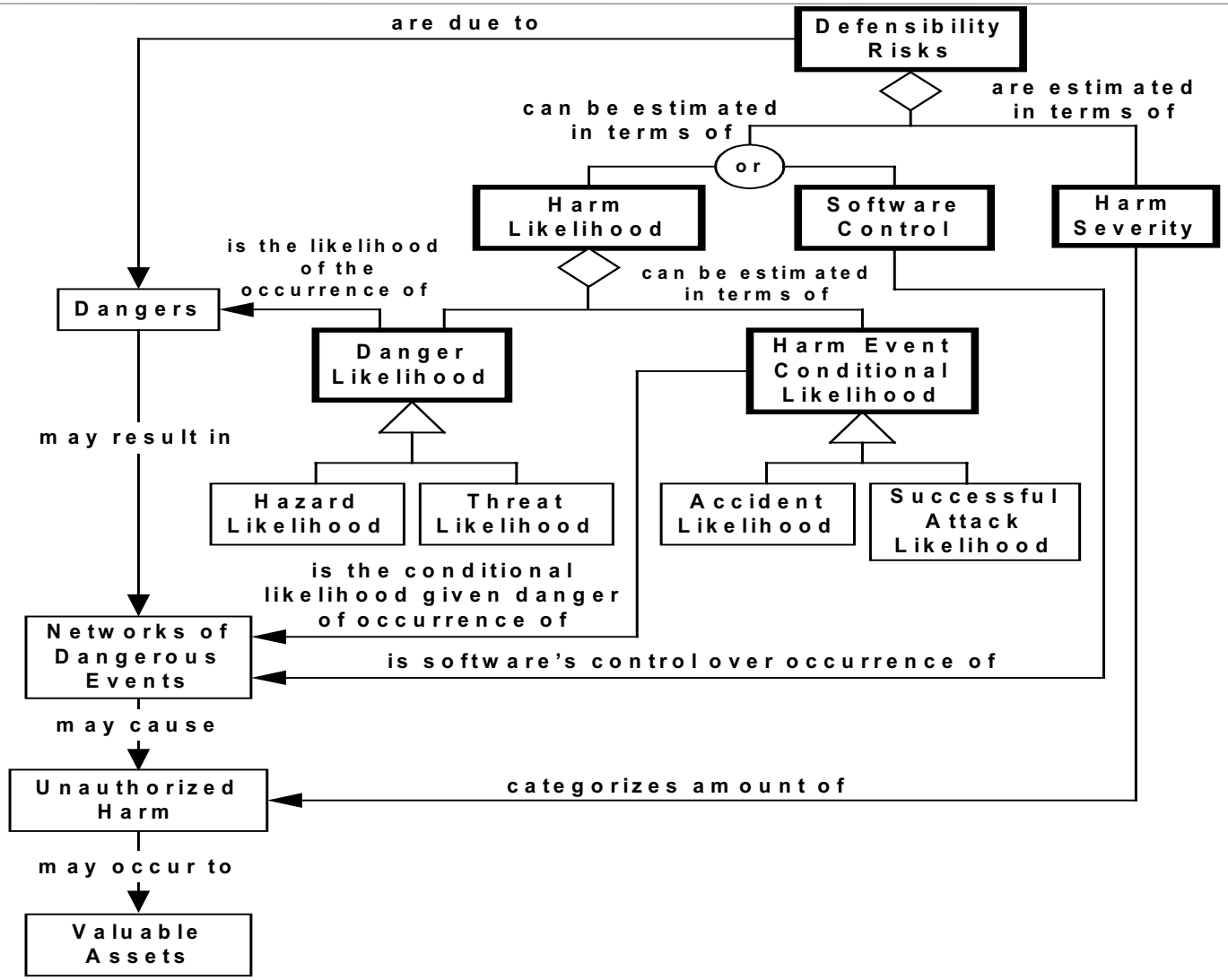
Dangers and Related Concepts



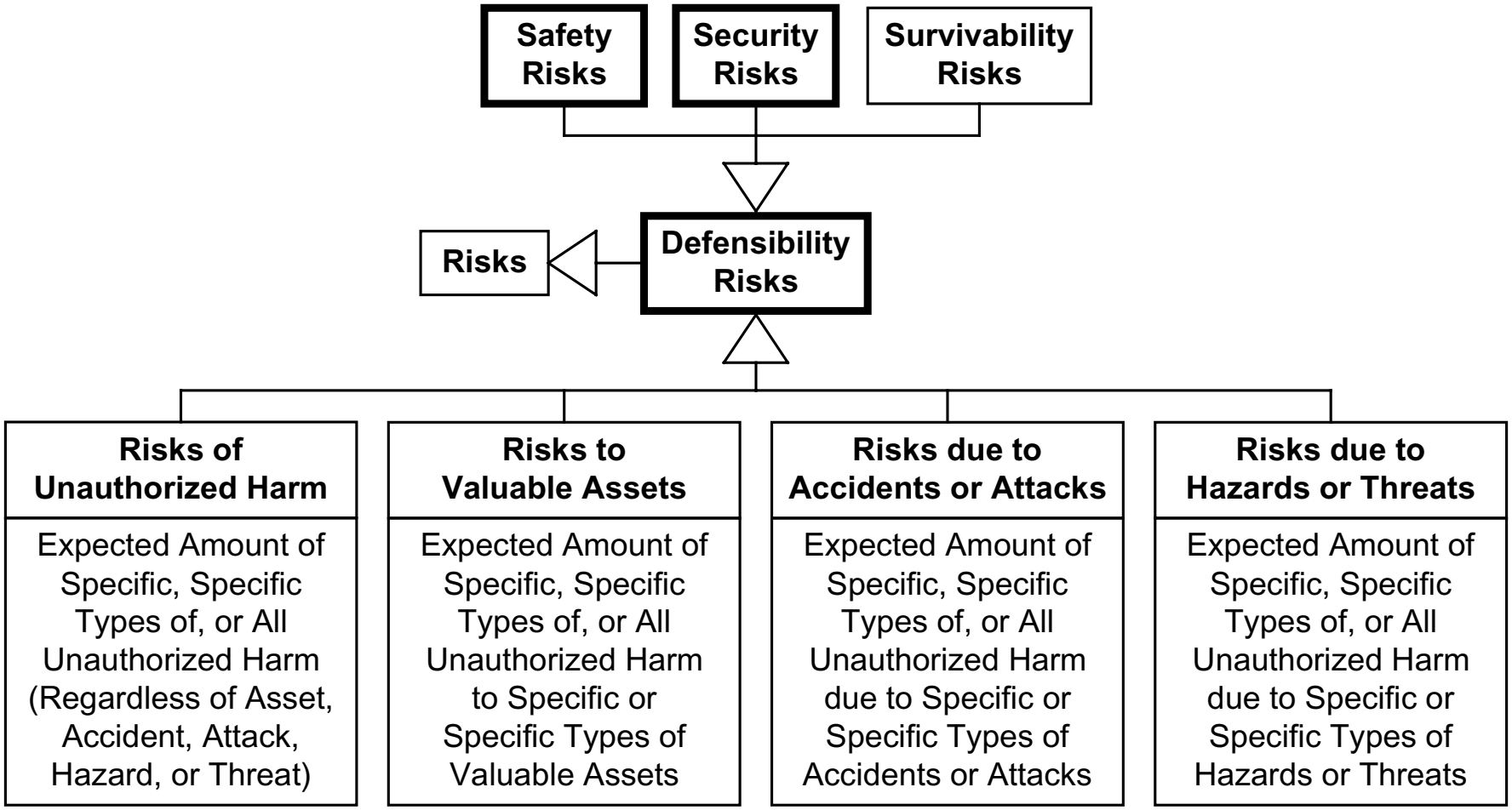
Mishaps and Misuses vs. Hazards and Threats



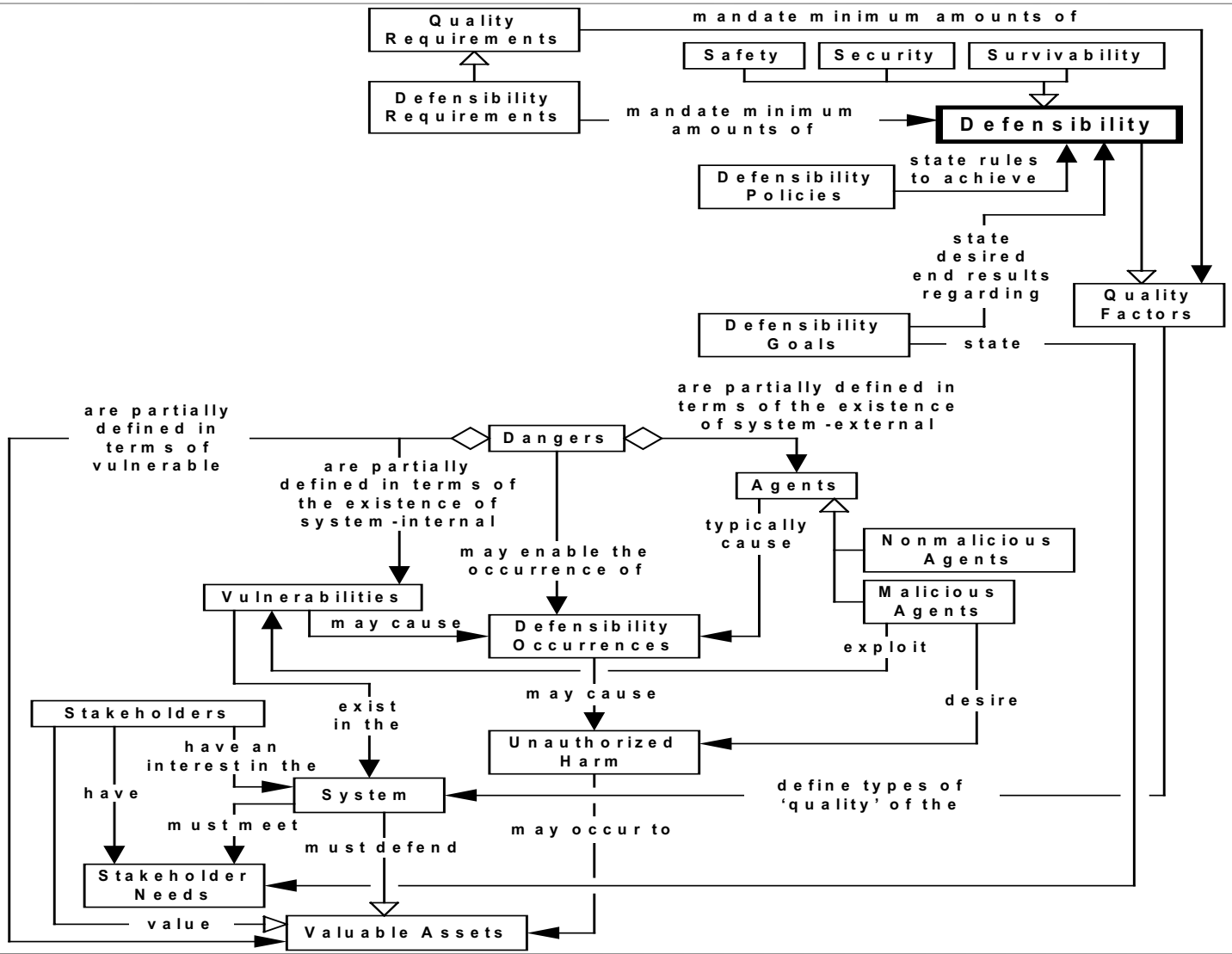
Defensibility Risks



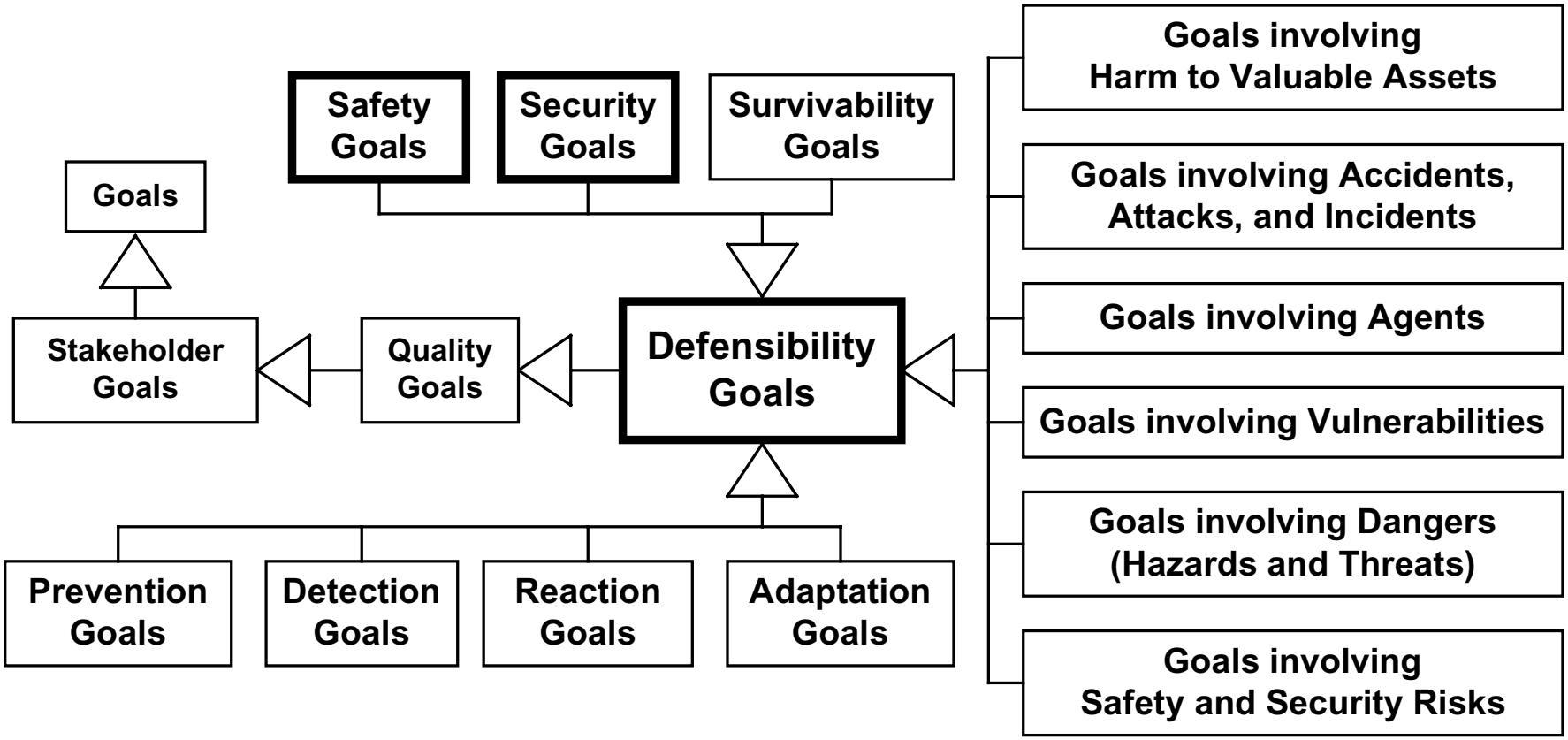
Types of Risks



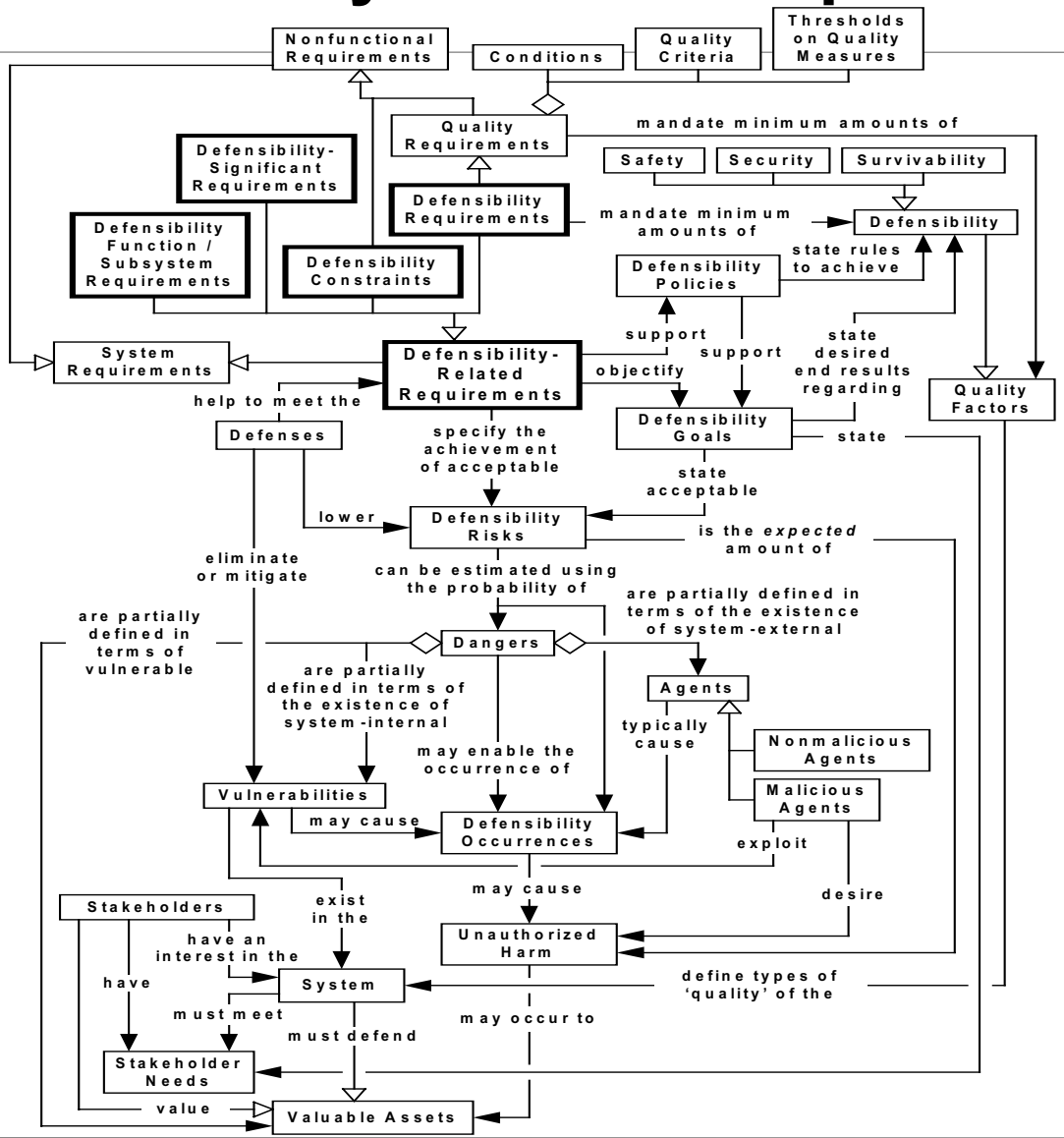
Safety and Security Goals and Policies



Types of Defensibility Goals



Safety- and Security-Related Requirements



Safety- and Security-Related Requirements

Types of Safety- and Security-Related Requirements

Too often only a Single Type of Requirements is considered.

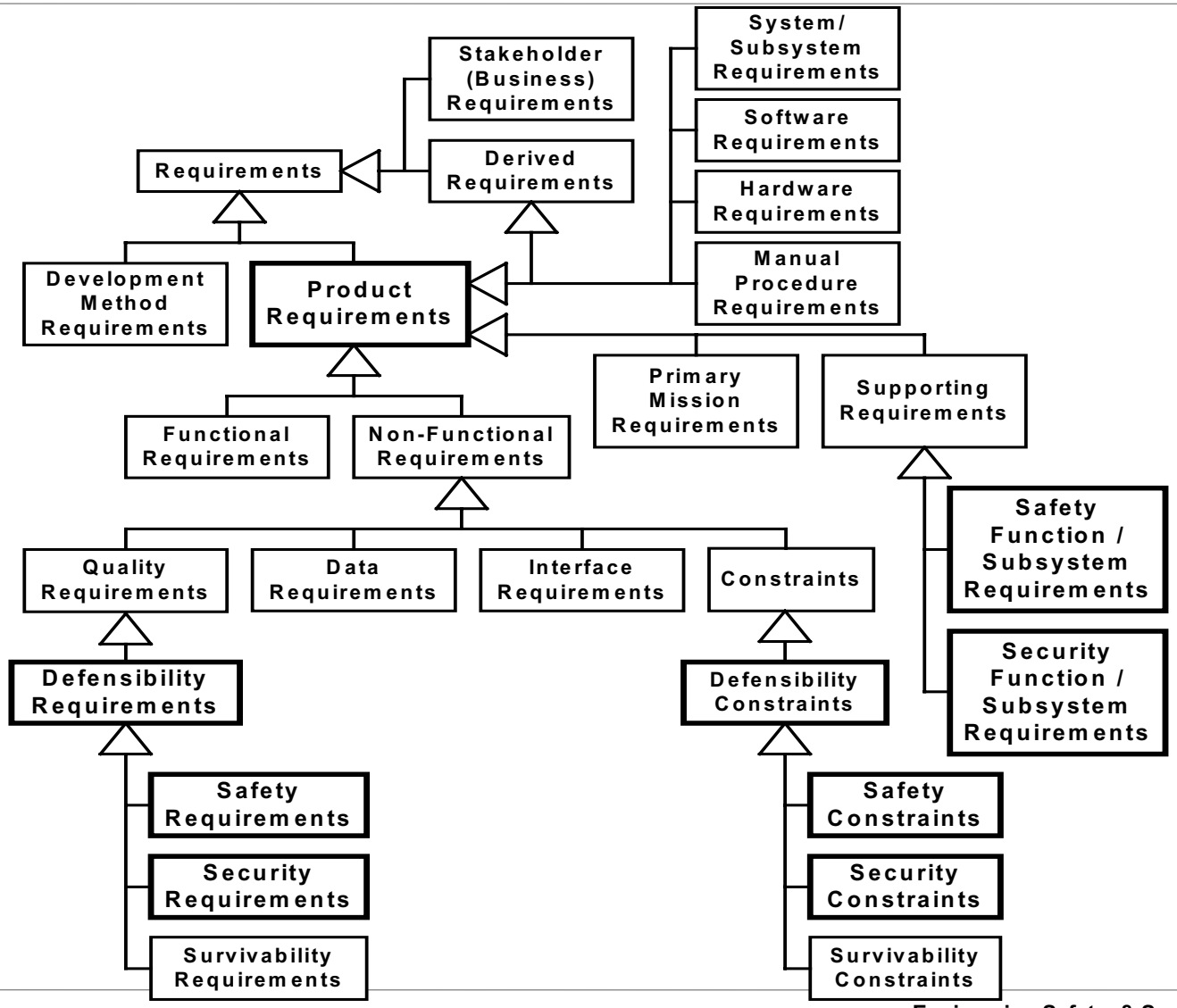
Not just:

- Special Non-Functional Requirements (NFRs):
  - Safety and Security Requirements are Quality Requirements are NFRs
- Safety- and Security-Significant Functional, Data, and Interface Requirements
- Constraints on Functional Requirements
- Architecture and Design Constraints
- Safety and Security Functions/Subsystems
- Software Requirements

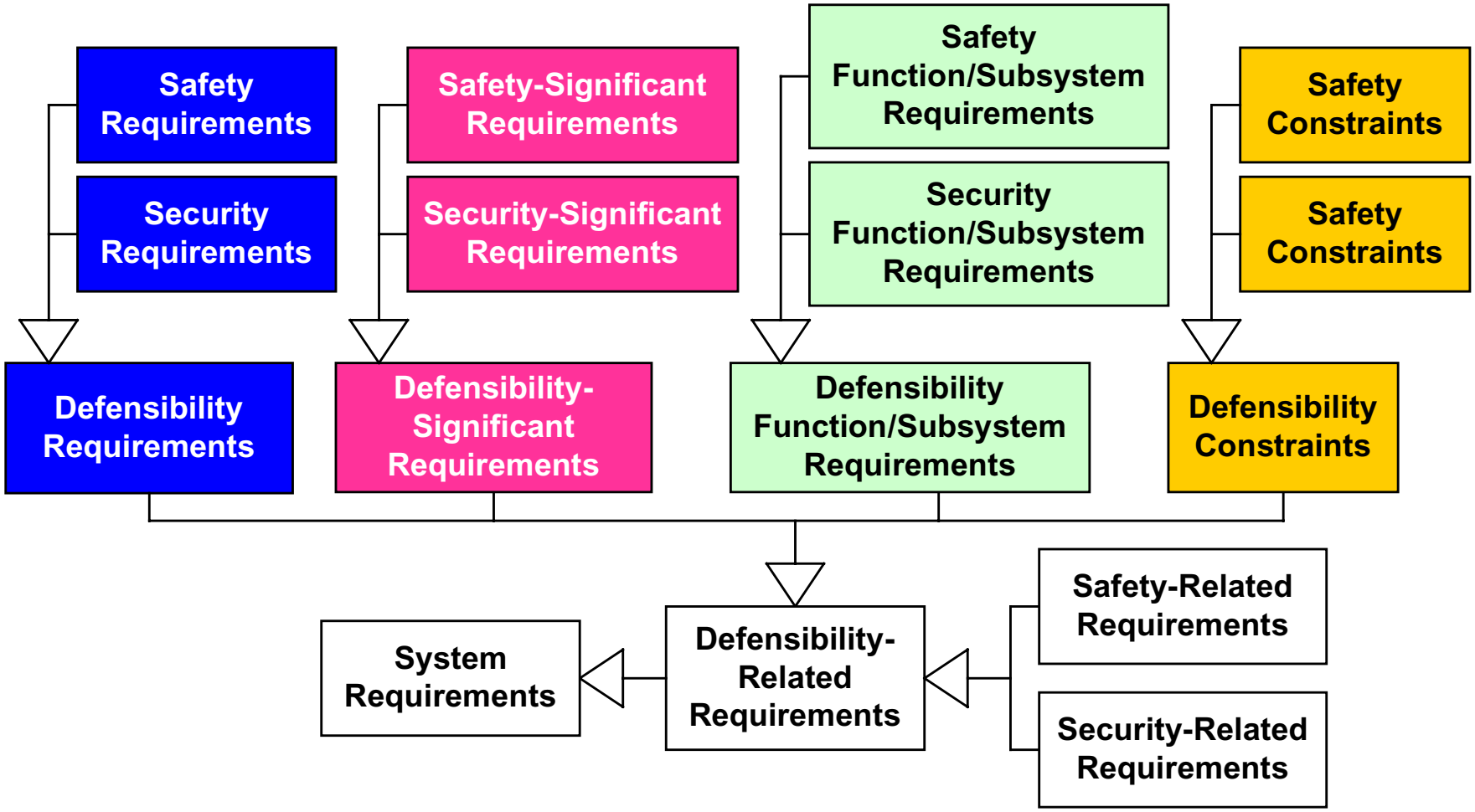
Reason for Presentation Title

Safety- and Security-Related Requirements for Software-Intensive Systems

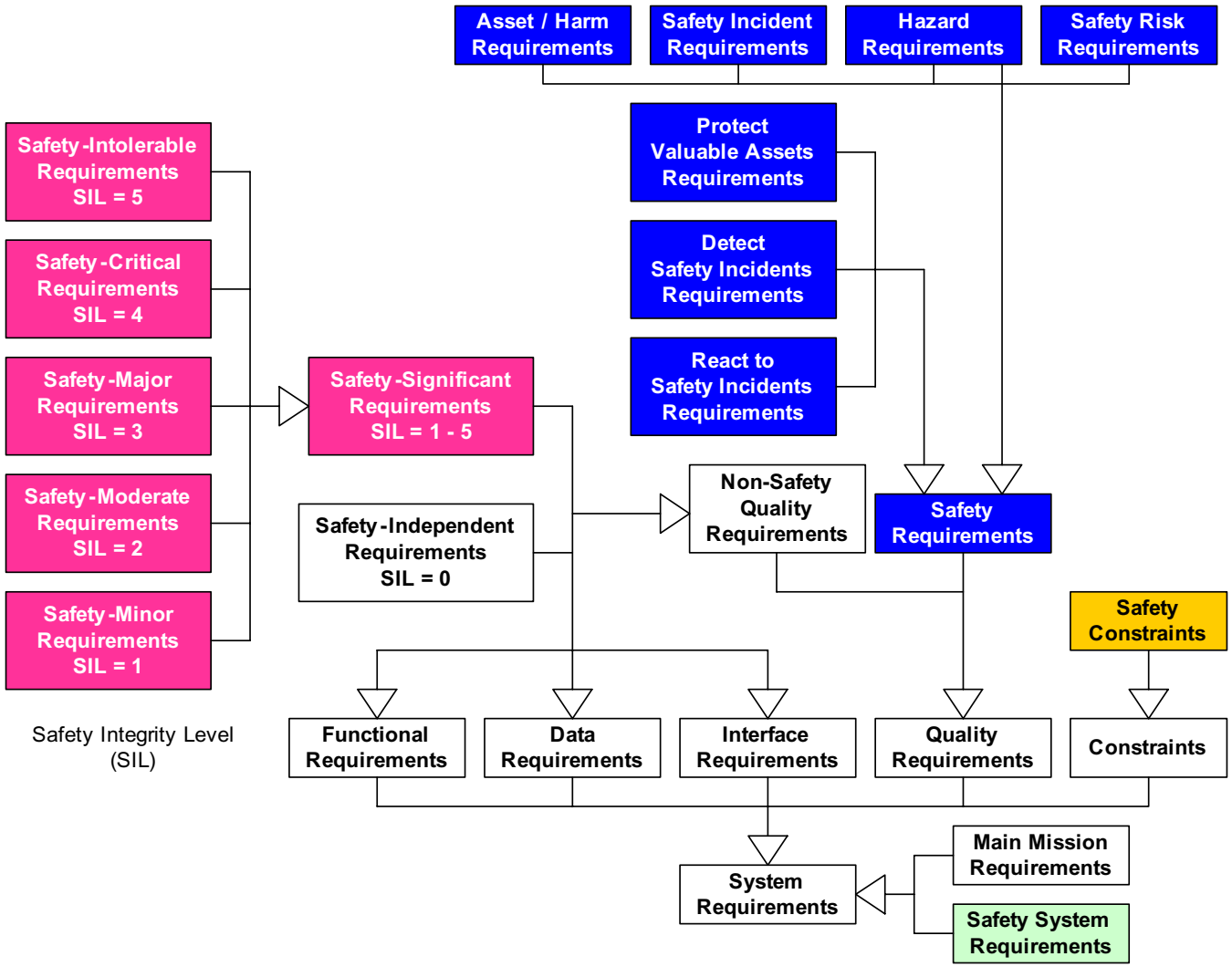
Types of Safety- and Security-Related Requirements



Types of Defensibility-Related Requirements



Types of Safety-Related Requirements



Safety and Security Requirements

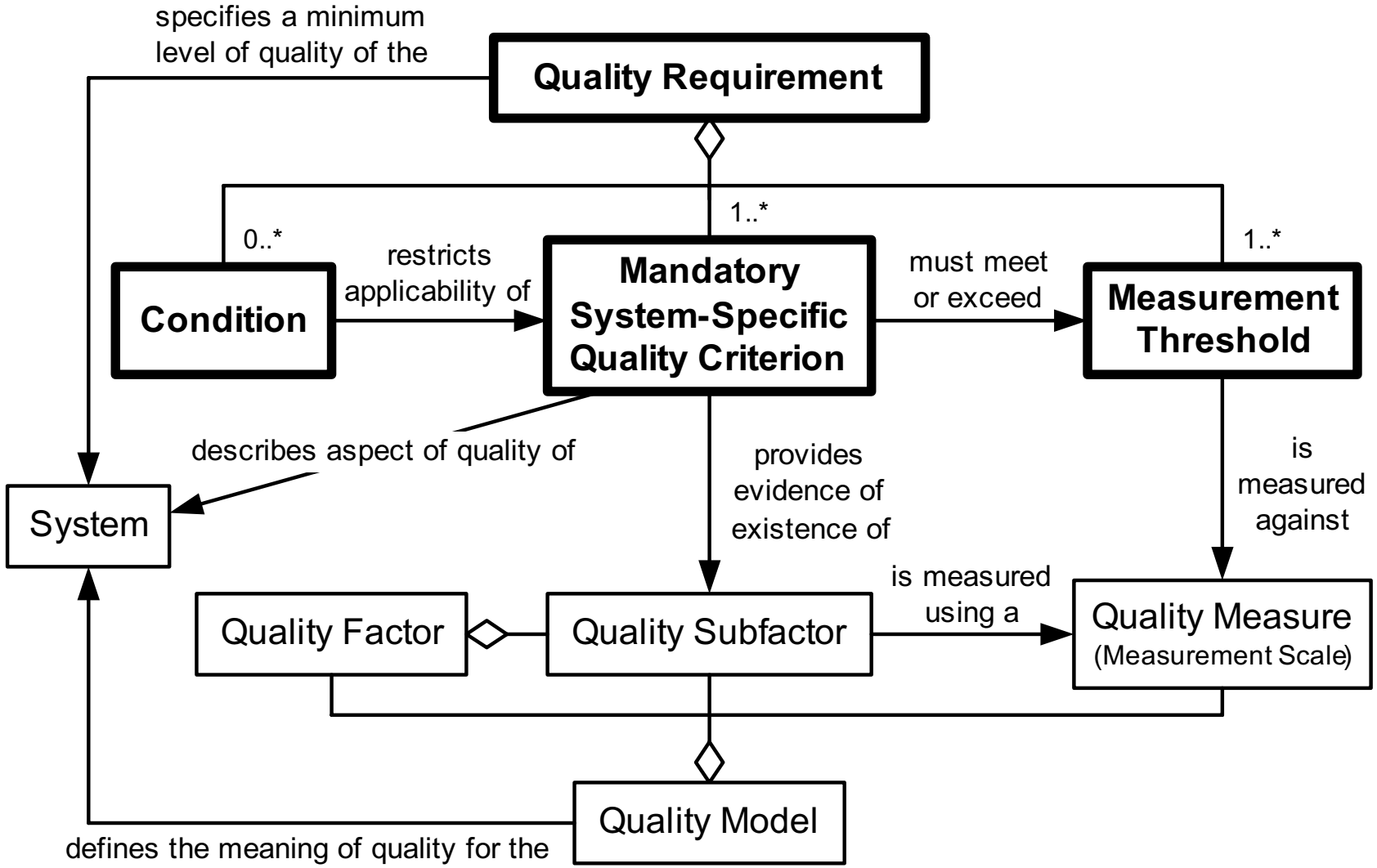
Safety and Security Requirements are Quality Requirements.

Quality Requirements are Product Requirements that specify a mandatory amount of a type of product quality (i.e., quality factor or quality subfactor).

Quality Requirements should be:

- Scalar (How Well or How Much)
- Based on a Quality Model
- Specified in Requirements Specifications
- Critically Important Drivers of the Architecture

Components of a Quality Requirement



Safety- and Security-Significant Requirements

Are identified based on Safety or Security (e.g., hazard or threat) Analysis

Subset of non-Safety and non-Security Requirements:

- Functional Requirements
- Data Requirements
- Interface Requirements
- Other Quality Requirements
- Constraints

Safety/Security Integrity Level (SIL) is not 0:

- May have minor Safety/Security Ramifications
- May be Safety- or Security-Critical
- May have intolerable Safety or Security Risk

SILs and SEALs

Safety/Security Integrity Level (SIL)

> a category of required safety or security for safety- or security-significant requirements.

Safety/Security Evidence Assurance Level (SEAL)

> a category of required evidence needed to assure stakeholders (e.g., safety or security certifiers) that the system is sufficiently safe or security (i.e., that it has achieved its required SIL).

SILs are for *requirements*

SEALs are for *components* that collaborate to fulfill requirements (e.g., *architecture*, design, coding, testing)

Safety and Security Function/Subsystem Rqmts.

Defensibility Function/Subsystem Requirements are requirements for functions or subsystems that exist strictly to improve defensibility (as opposed to support the primary mission requirements).

- **Safety Function or Subsystem Requirements** are requirements for safety functions or subsystems.
- **Security Function or Subsystem Requirements** are requirements for security functions or subsystems.

Safety Function/Subsystem Requirements

Functions or subsystems strictly added for safety:

- Aircraft Safety Subsystems:
  - Collision Avoidance System
  - Engine Fire Detection and Suppression
  - Ground Proximity Warning System (GPWS)
  - Minimum Safe Altitude Warning (MSAW)
  - Wind Shear Alert
- Nuclear Power Plant:
  - Emergency Core Coolant System

All requirements for such functions/subsystems are safety-related.

Example Safety Function/Subsystem Requirements

"Except when the weapons bay doors are open or have been open within the previous 30 seconds, the weapons bay cooling subsystem shall maintain the temperature of the weapons bay below X° C."

"The Fire Detection and Suppression Subsystem (FDSS) shall detect smoke above X ppm in the weapons bay within 2 seconds at least 99.9% of the time."

"The FDSS shall detect temperatures above X° C in the weapons bay within 2 seconds at least 99% of the time."

"Upon detection of smoke or excess temperature, the FDSS shall begin fire suppression within 1 second at least 99.9% of the time."

Security Function/Subsystem Requirements

Functions or subsystems strictly added for security:

- Access Control Function
- Encryption/Decryption Subsystem
- Firewalls
- Intrusion Detection System
- Virus Protection Application

All requirements for such functions/subsystems are security-related.

Look in the Common Criteria for many reusable example security function requirements.

Safety and Security Constraints

A **Constraint** is any Engineering Decision that has been chosen to be mandated as a Requirement. For example:

- Architecture Constraints
- Design Constraints
- Implementation Constraints
  (e.g., coding standards or safe language subset)
- Testing Constraints

A **safety constraint** is any constraint primarily intended to ensure a minimum level of safety (e.g., a mandated safeguard).

Safety and Security Standards often mandate Industry Best Practices as Constraints.

Example ZATS Safety Constraints

"When the vehicle is stopped in a station with the doors open for boarding, the horizontal gap between the station platform and the vehicle door threshold shall be no greater than 25 mm (1.0 in.) and the height of the vehicle floor shall be within plus/minus 12 mm (0.5 in.) of the platform height under all normal static load conditions…"

Automated People Mover Standards – Part 2: Vehicles, Propulsion, and Braking (ASCE 21-98)

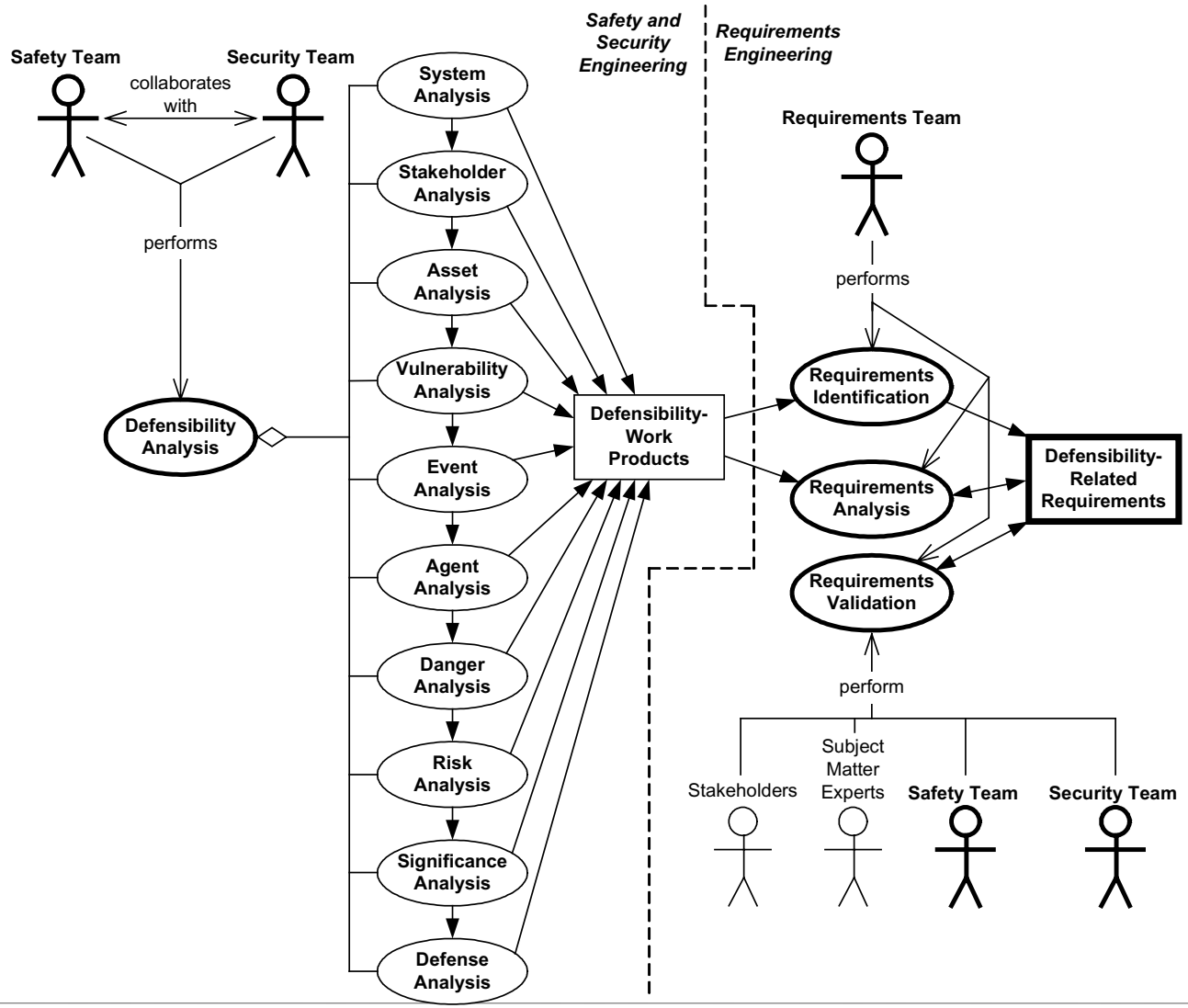
"Oils and hydraulic fluids shall be *flame retardant*, except as required for *normal lubrication*."

Note need to define flame retardant and normal lubrication.

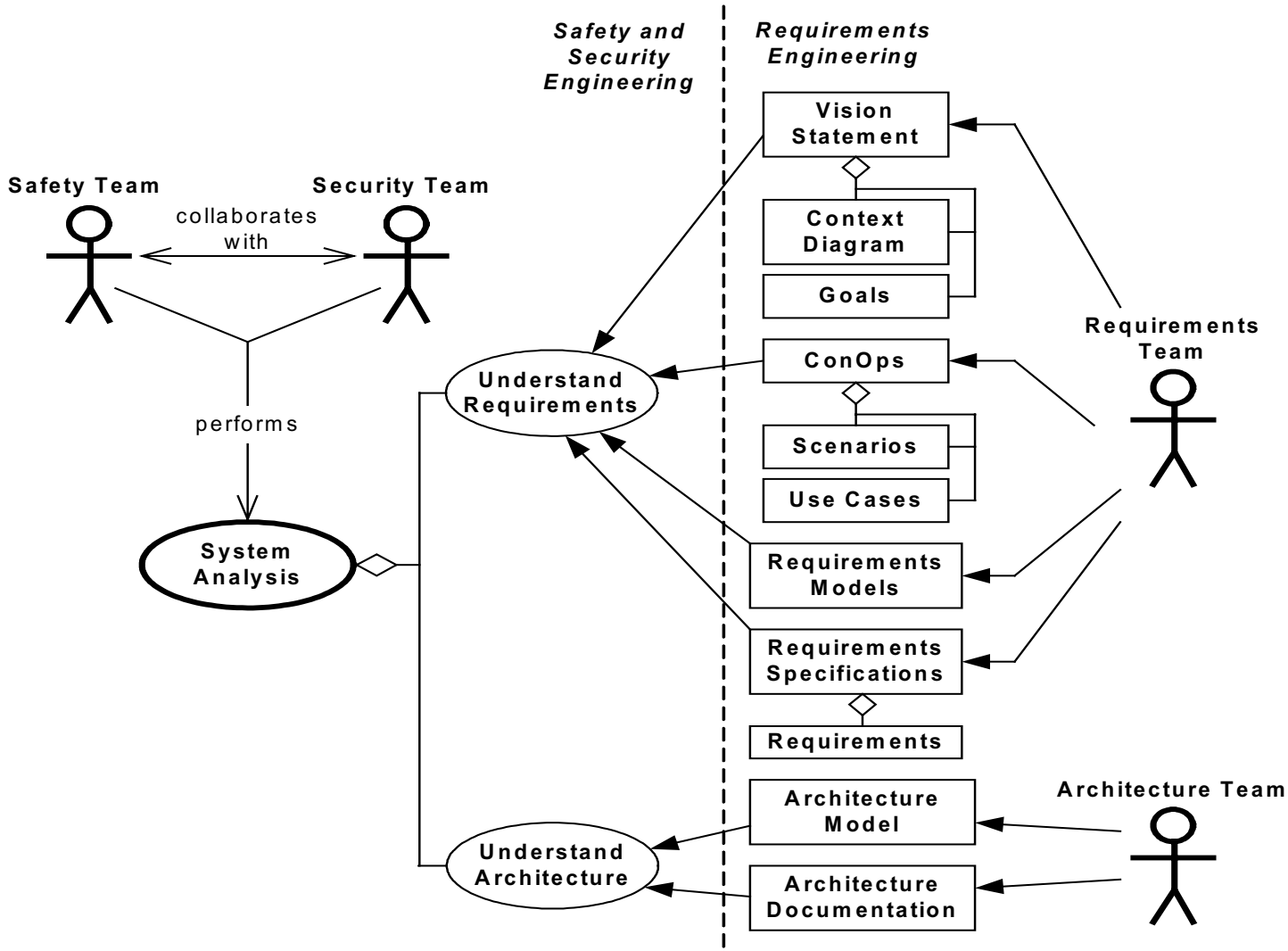
Common Process:

A Basis for Effective Collaboration

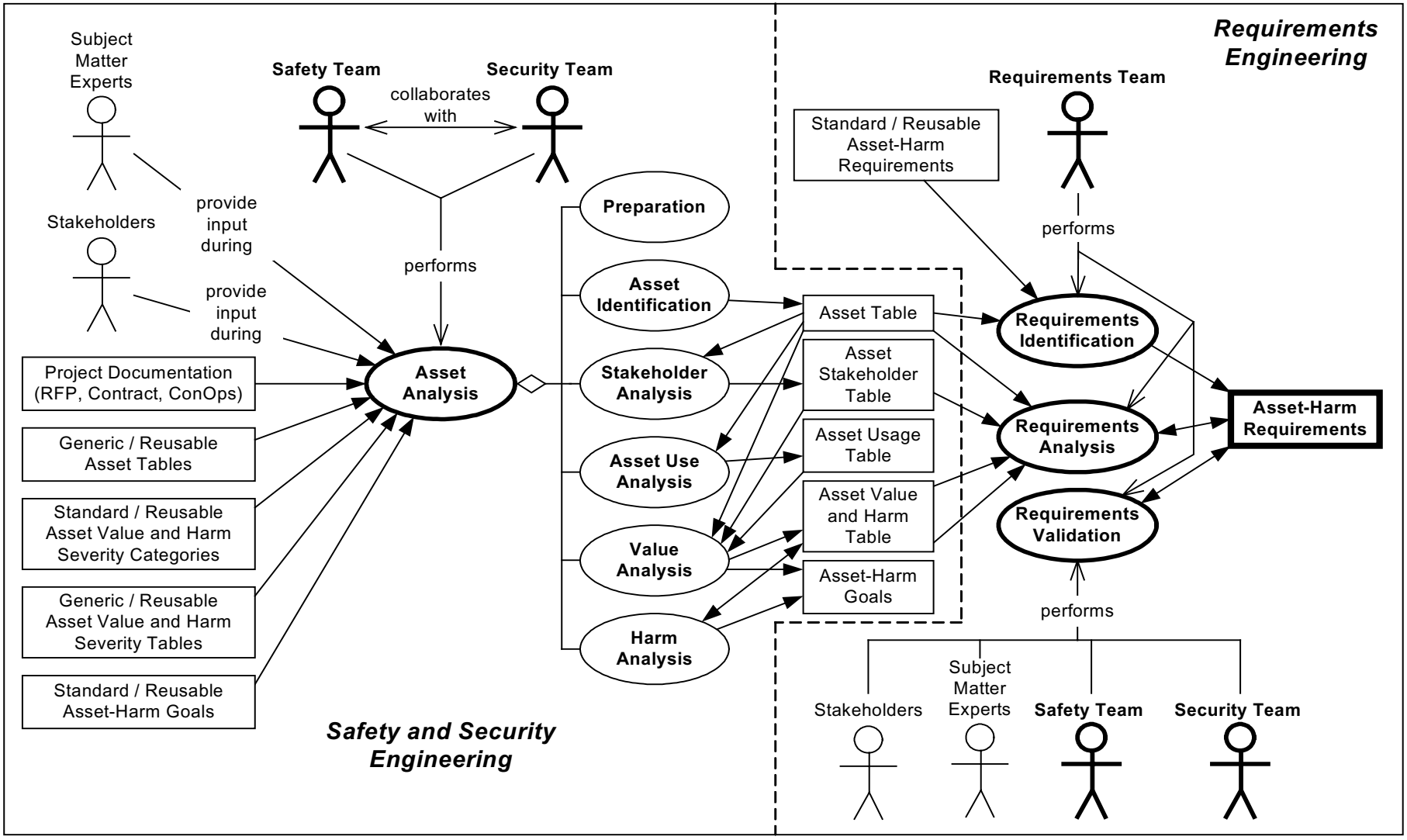
Defensibility & Requirements Engineering



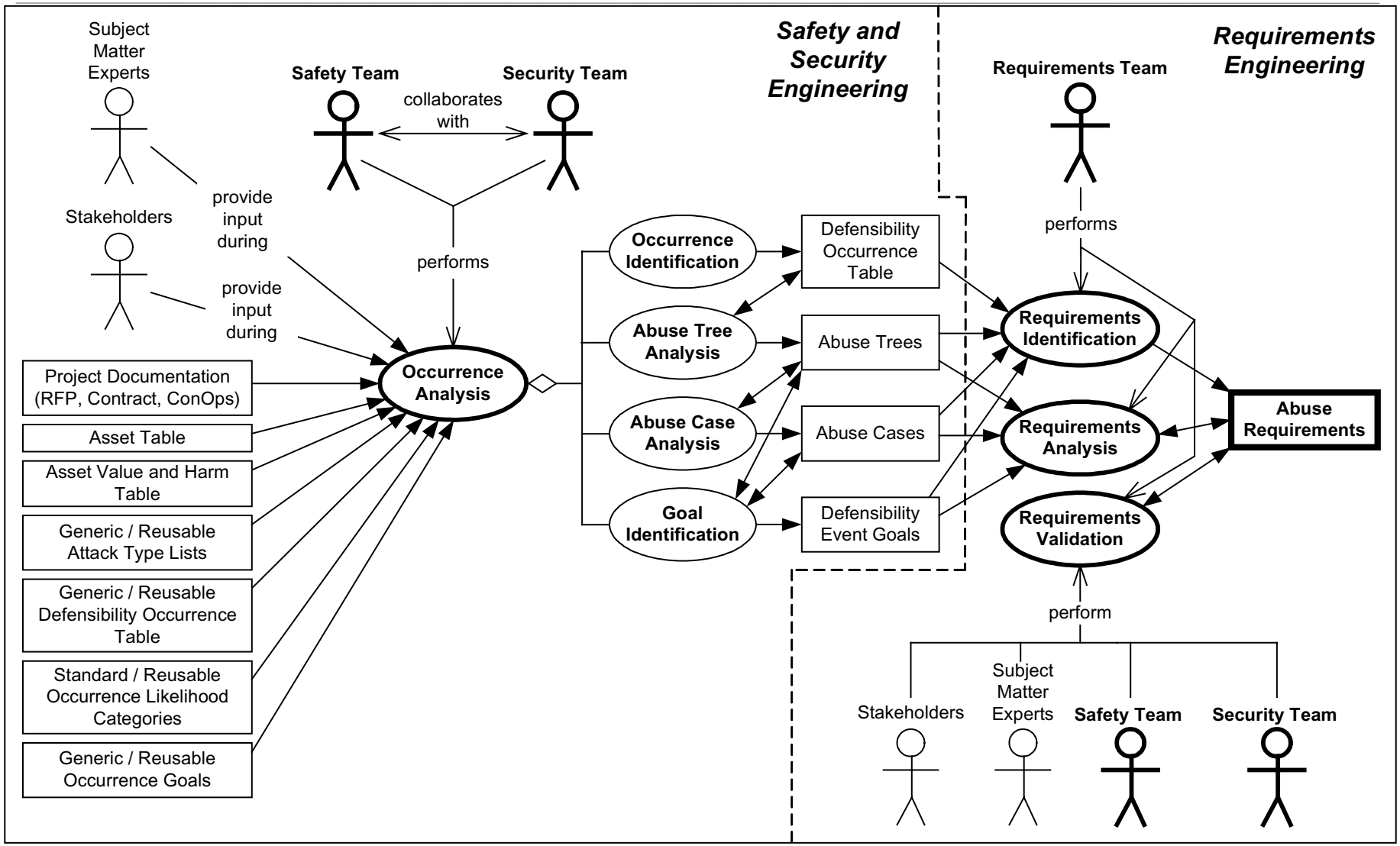
Systems Analysis



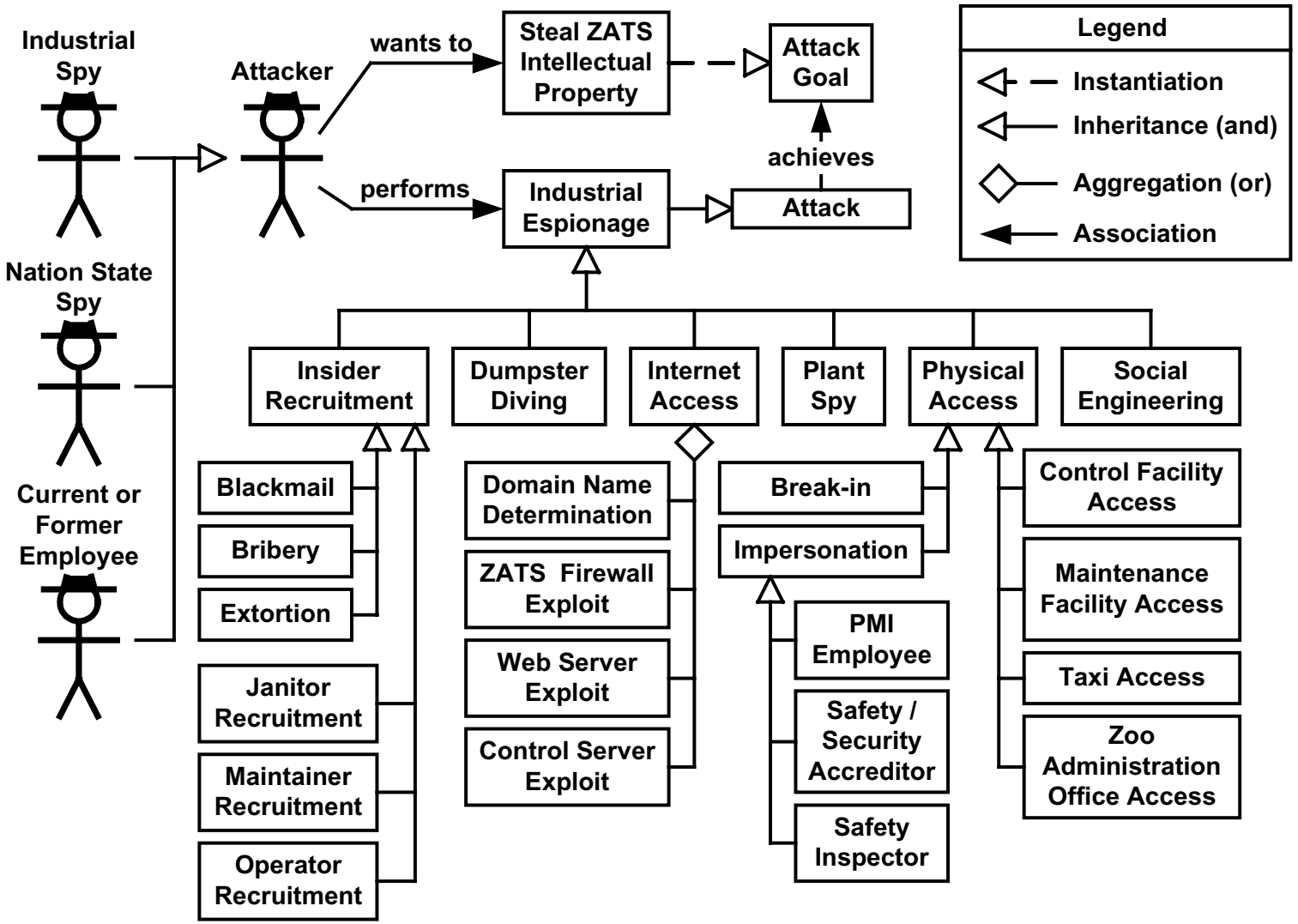
Asset Analysis



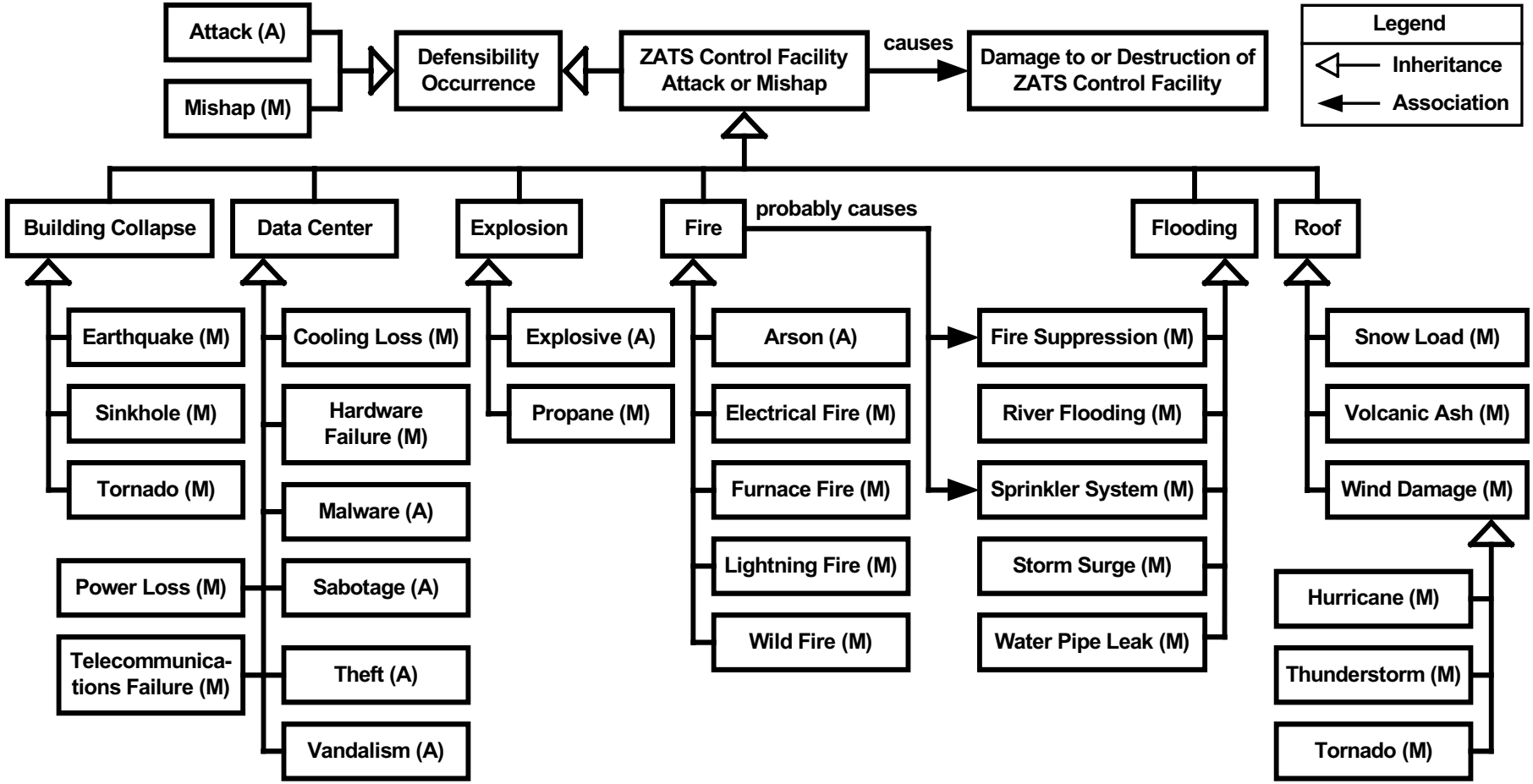
Defensibility Occurrence Analysis



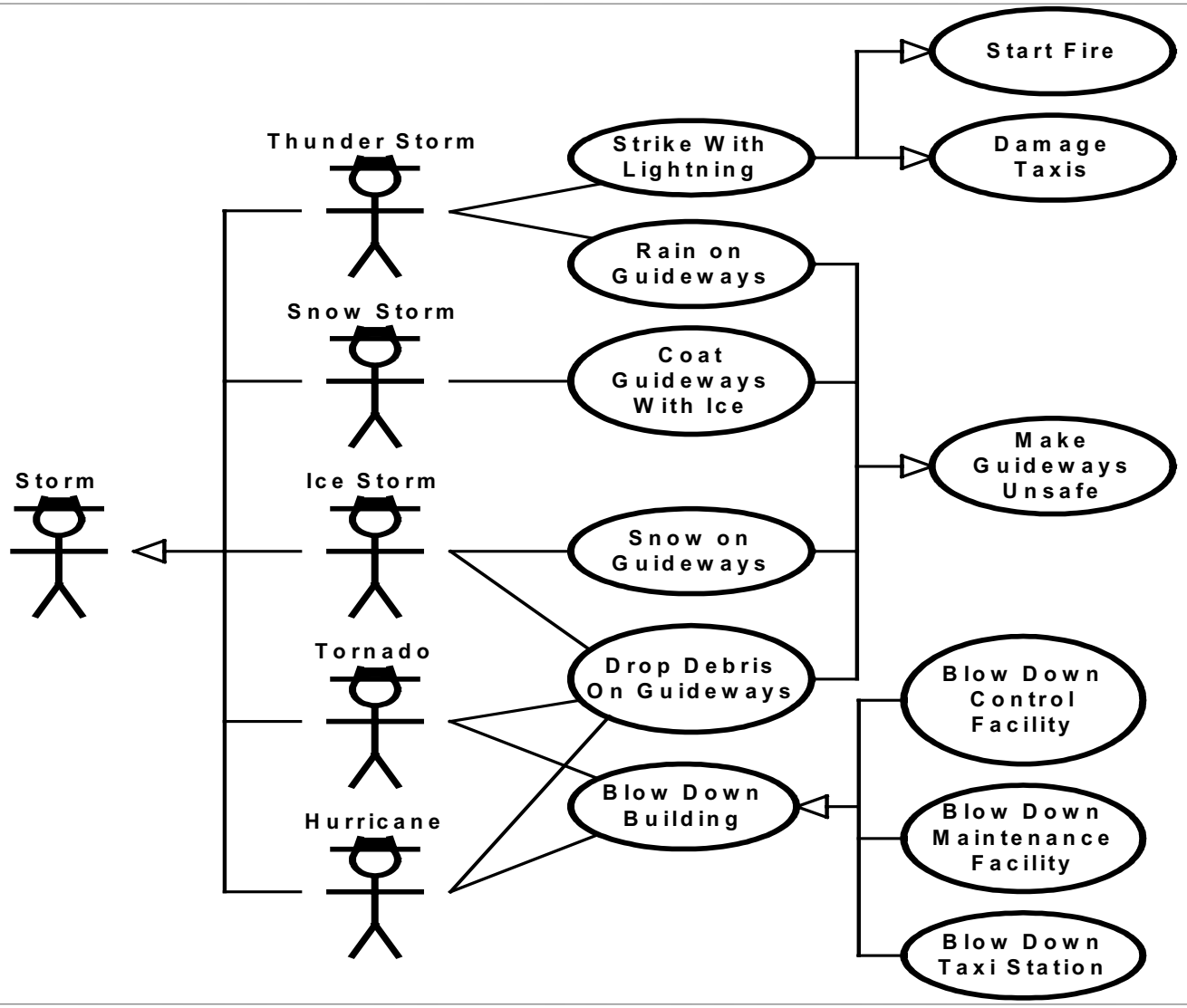
Example Abuse (Attack) Tree



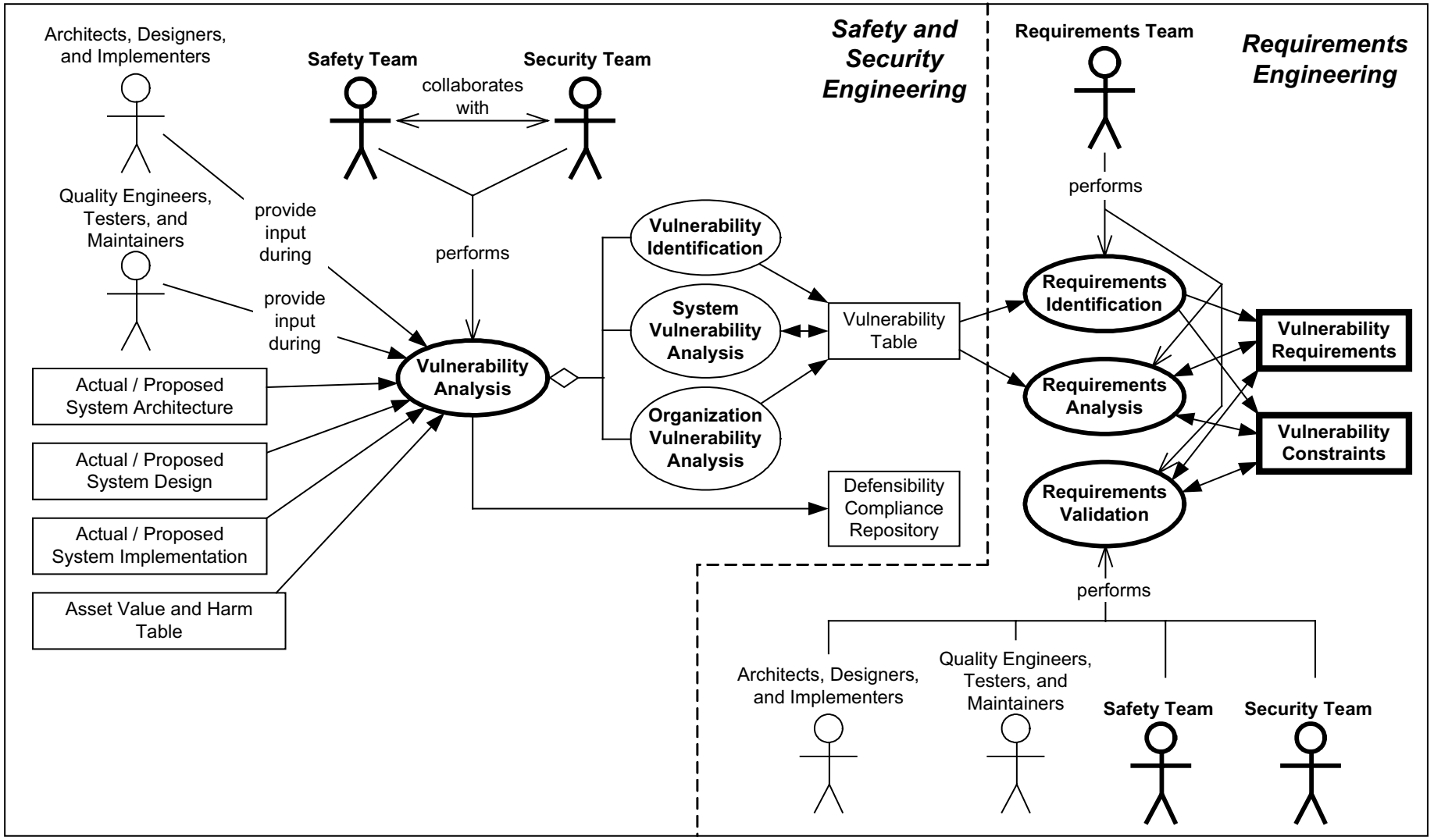
Example Abuse (Mishap and Attack) Tree



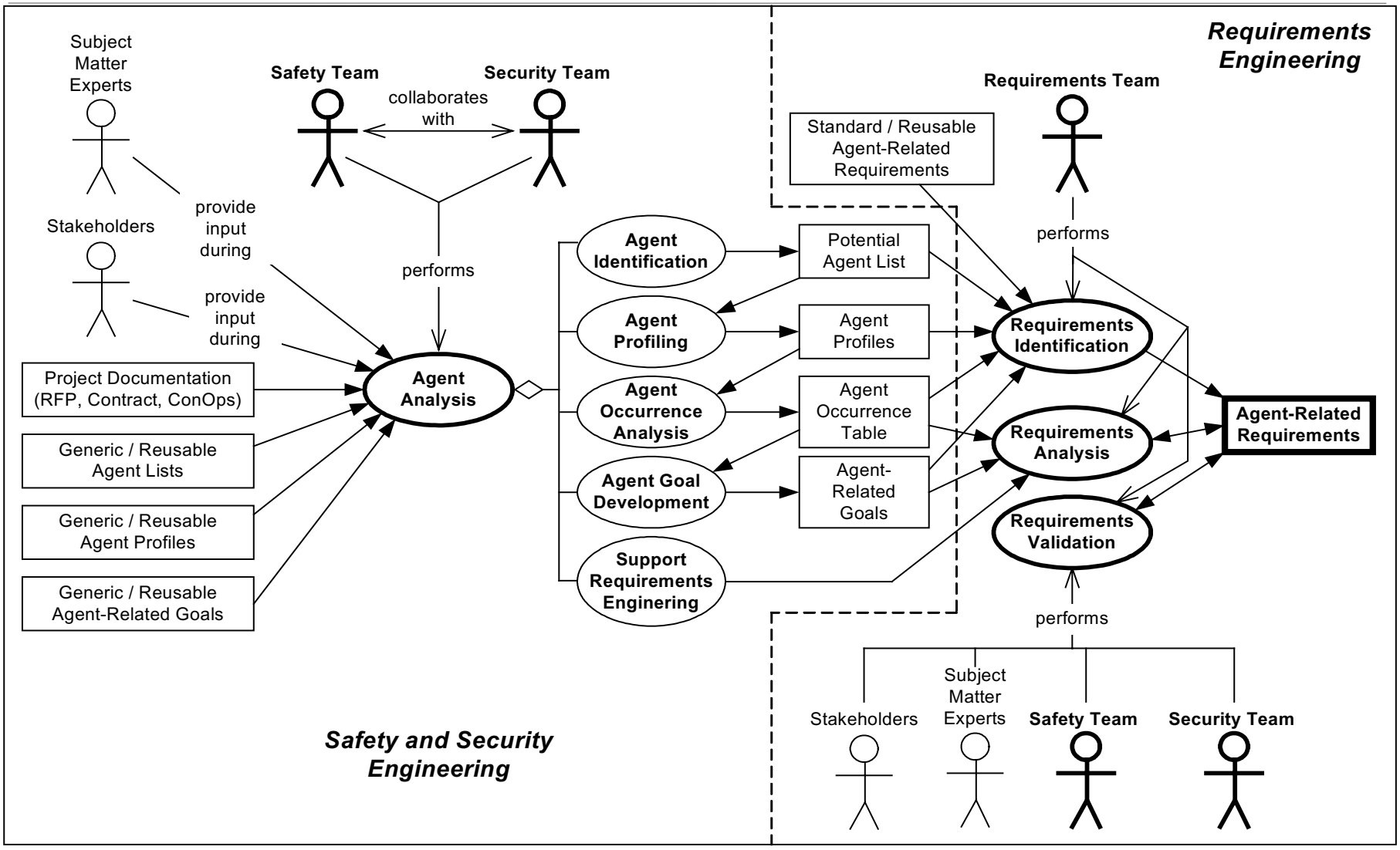
Example Abuse (Mishap and Misuse) Cases



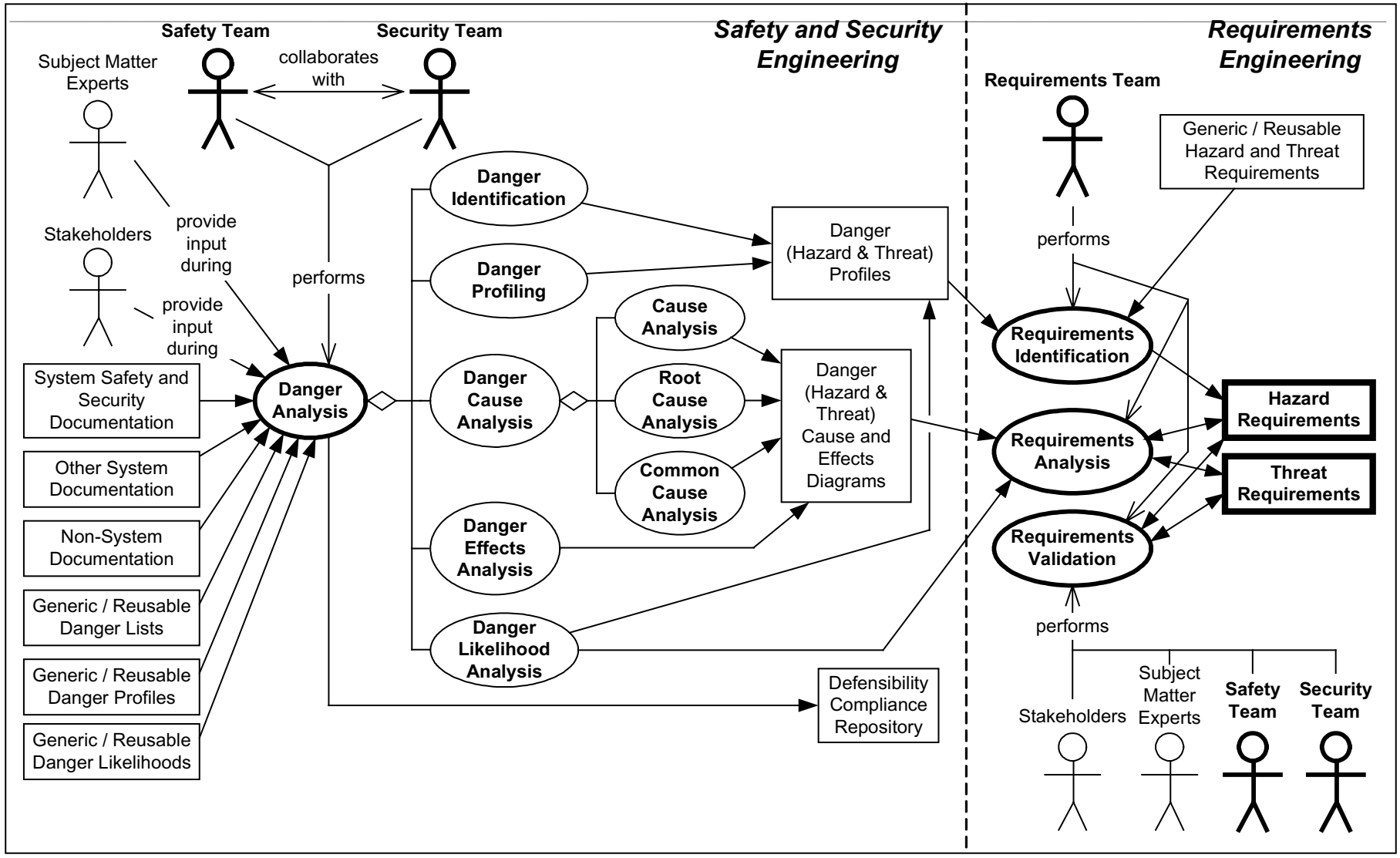
Vulnerability Analysis



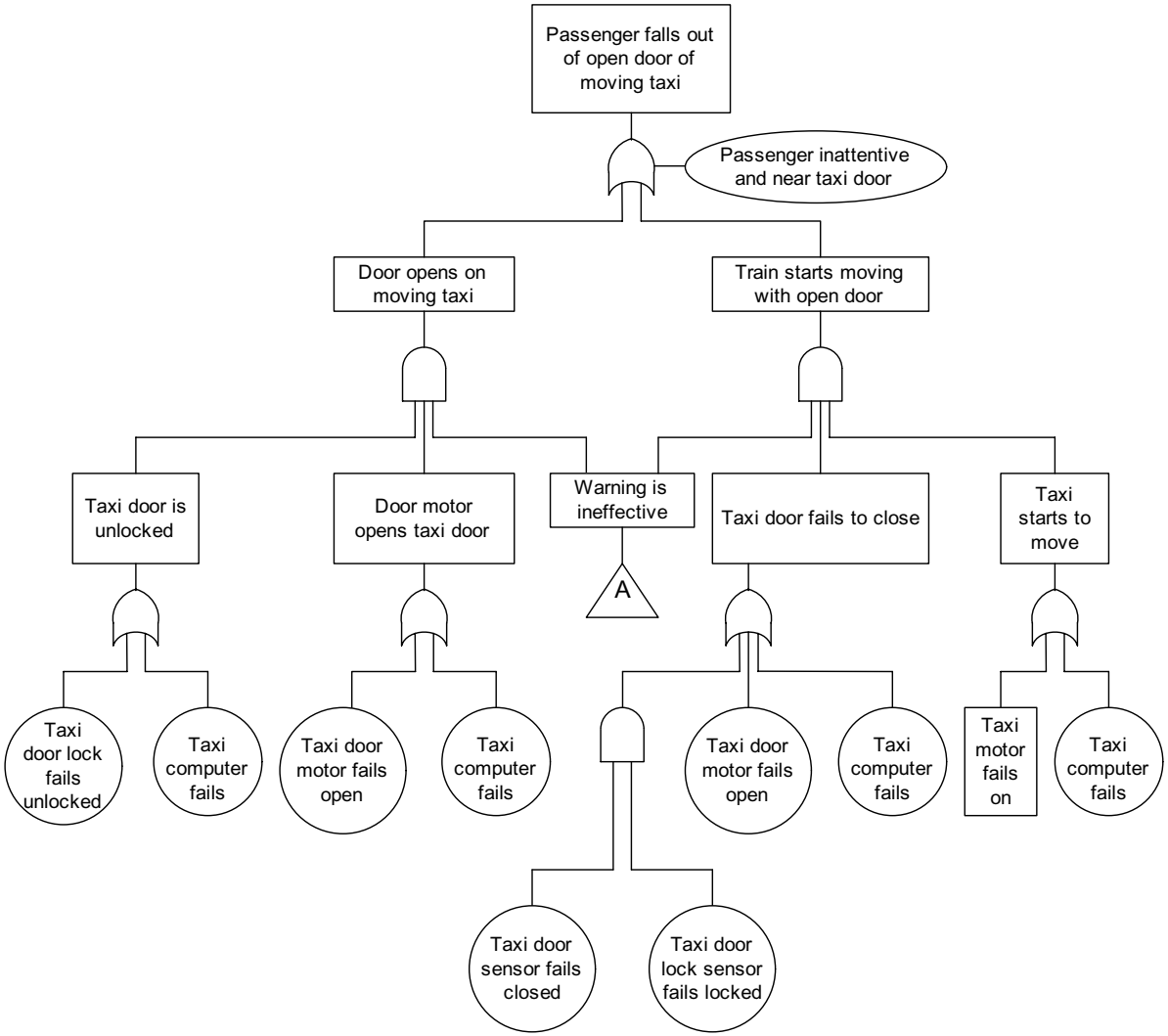
Agent Analysis



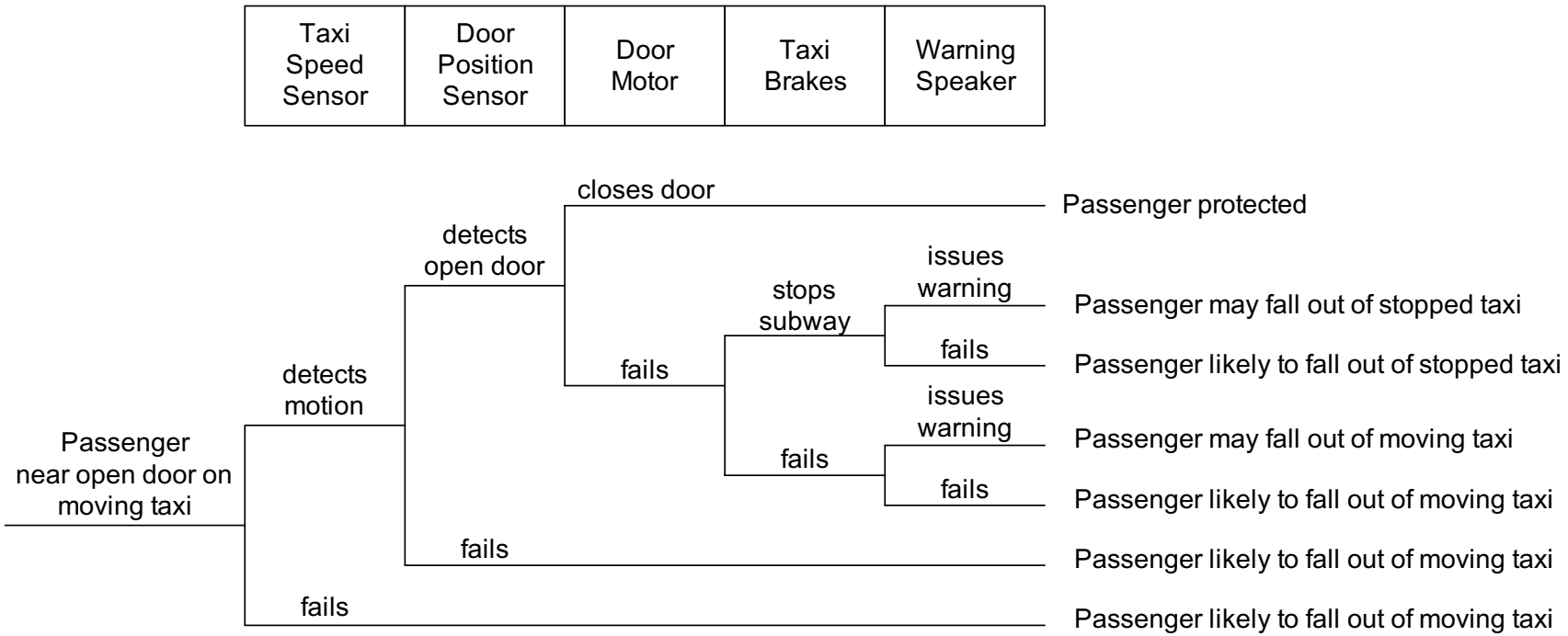
Danger Analysis



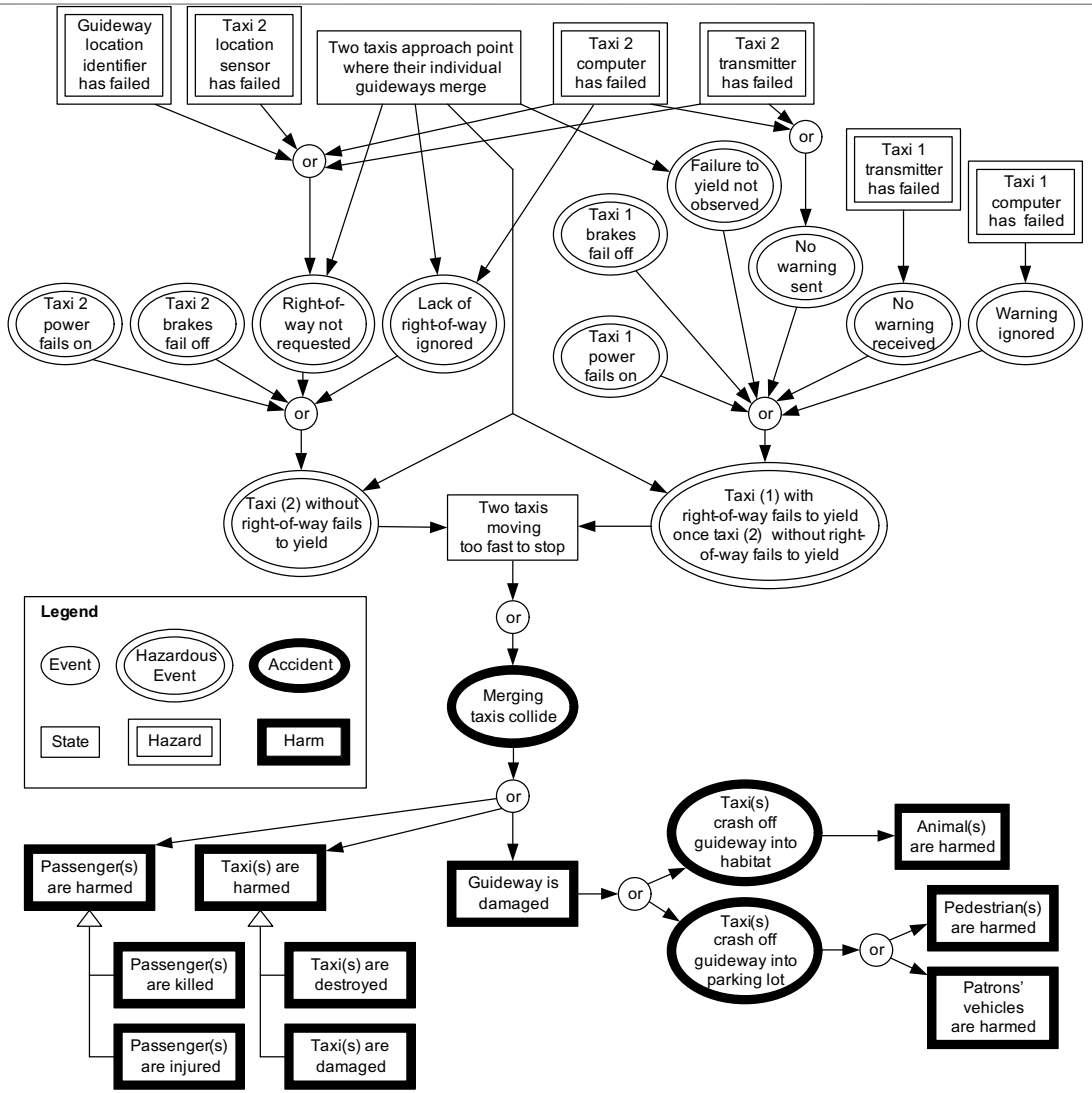
Example Fault Tree



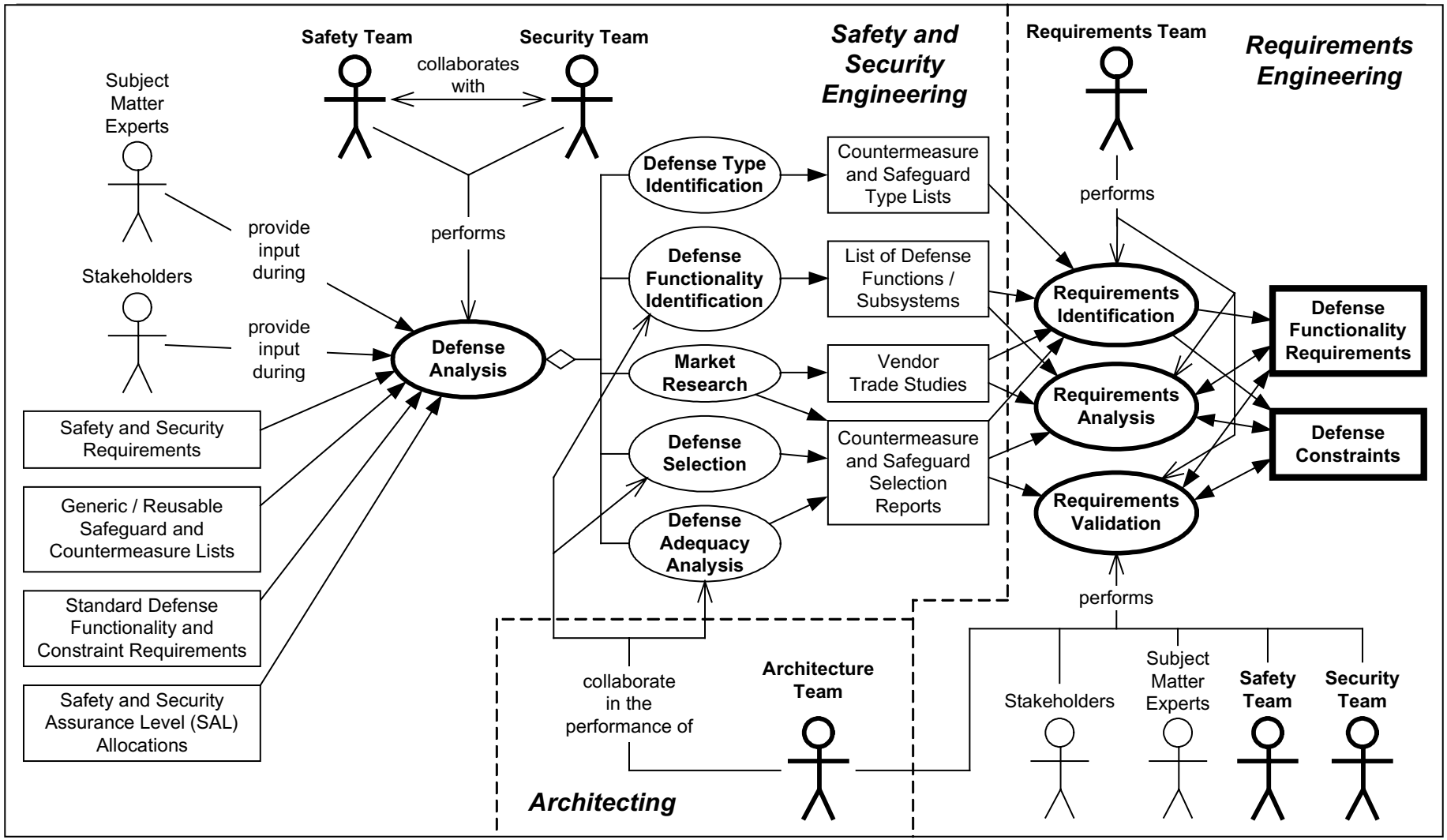
Example Event Tree



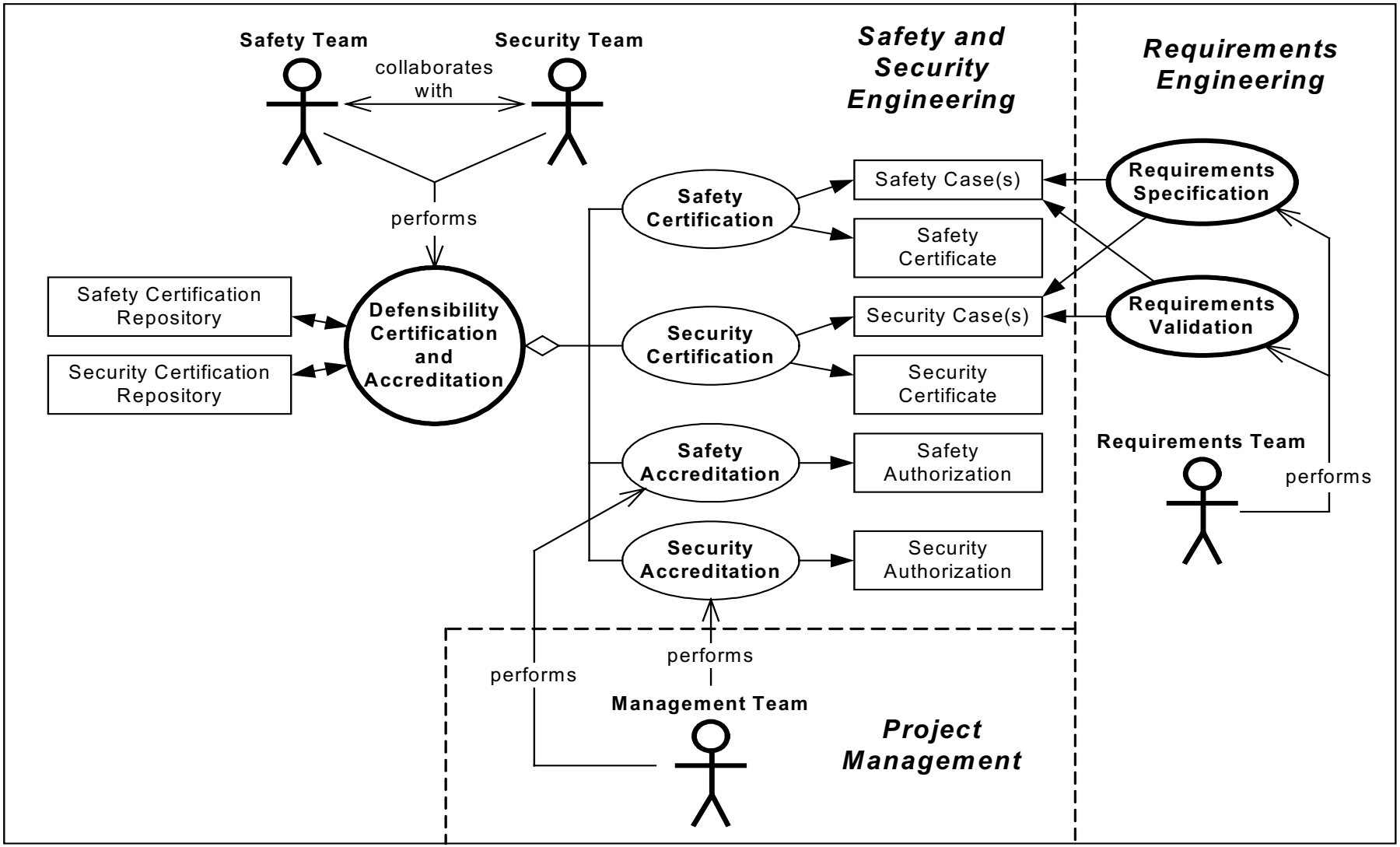
Example Cause and Effect Tree



Defense Analysis



Defense Certification and Accreditation



Conclusion:

Process Improvement Recommendations

Conclusion$_1$

Engineering safety-significant requirements requires *appropriate*:

- Concepts
- Methods
- Techniques
- Tools
- Expertise

These must come from *both:*

- Requirements Engineering
- Safety Engineering

Conclusion$_2$

There are four types of Safety- and Security-related Requirements:

- Safety and Security Quality Requirements
- Safety- and Security-Significant Requirements
- Safety and Security Function/Subsystem Requirements
- Safety and Security Constraints

Different Types of Safety- and Security-related Requirements have different Structures.

These different Types of Requirements need to be identified, analyzed, and specified differently.

Conclusion$_3$

Processes for Requirements Engineering, Safety Engineering, and Security Engineering need to be:

- Properly interwoven.
- Consistent with each other.
- Performed collaboratively and in parallel (i.e., overlapping in time).

Process Improvement Recommendations

Ensure close Collaboration among Safety, Security, and Requirements Teams.

Better Integrate Safety and Security Processes:

- Concepts and Terminology
- Techniques and Work Products
- Provide Cross Training

Better Integrate Safety and Security Processes with Requirements Process:

- Early during Development Cycle
- Clearly define Team Responsibilities
- Provide Cross Training

Develop all types of Safety- and Security-related Requirements.

Ensure that these Requirements have the proper Properties.

Software Engineering Institute | Carnegie Mellon