Engineering
Safety-Related Requirements for Software-Intensive Systems

Donald Firesmith, Software Engineering Institute, USA
The Challenge

- Poor requirements are a root cause of many (or most) accidents involving software-intensive systems.

- Requirements engineering and safety 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

- This separation of RE and SE causes poor safety-related requirements.
Topics

- Requirements Engineering Overview for Safety Team
- Safety Engineering Overview for Requirements Team
- Break (3:00PM – 3:30PM)
- Safety-Related Requirements:
  - Safety [Quality] Requirements
  - Safety-Significant Requirements
  - Safety Subsystem Requirements
  - Safety Constraints
- Method for Engineering Safety-Related Requirements
Requirements Engineering Overview

- Definition of Requirements Engineering
- Importance and Difficulty of Requirements Engineering
- Goals vs. Scenarios vs. Requirements
- Characteristics of Good Requirements
- Types of Requirements
Requirements Engineering

- **Requirements engineering (RE)** is 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 ongoing* manner rather than according to the traditional Waterfall development cycle.
Importance of Requirements

- 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
Difficulty of Requirements

“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.”

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.
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.
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.
- Requirements are analyzed using scenarios.
- Requirements must have certain characteristics (e.g., verifiable and feasible).
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
Some Problems due to Poor Requirements

☑ Ambiguous Requirements:
  ● Developers misinterpret Subject Matter Expert (SME) intentions.
  ● “The system shall be safe.”
  ● How safe? Safe in what way?

☑ Incomplete Requirements:
  ● Developers must guess SME intentions.
  ● The system shall do X.”
More Problems

○ Missing Requirements:
  ● What shall the system do if it can’t do X?
  ● Unusual combinations of conditions often result in accidents.
  ● What shall the system do if event X occurs when the system is simultaneously in states Y and Z?

○ Unnecessary Constraints:
  ● Inappropriate architecture and design constraints unnecessarily specified as requirements such as:
    ● User ID and password for identification and authentication.
Types of Requirements

- Product Requirements
  - Process Requirements
  - Functional Requirements
  - Non-Functional Requirements
  - Data Requirements
  - Interface Requirements
  - Quality Requirements
  - Constraints
  - Stakeholder (Business) Requirements
  - Software Requirements
  - System/Subsystem Requirements
  - Main Mission Requirements
  - Specialty Engineering Subsystem Requirements

Engineering Safety-Related Requirements for Software-Intensive Systems
Product Requirements

- A **product requirement** is a requirement for a *product* (e.g., system, subsystem, software application, or component).
  - A **functional requirement** is a product requirement that specifies a mandatory *function* (i.e., behavior) of the product.
  - A **data requirement** is a product requirement that specifies mandatory [types of] data that must be manipulated by the product.
  - An **interface requirement** is a product requirement that specifies a mandatory interface with (or within) the product.
  - A **quality requirement** is a product requirement that specifies a mandatory amount of a type of product quality.
  - A **constraint** is a property of the product (e.g., design decision) that is ordinarily not a requirement but which is being mandated as if it were a normal requirement.
Quality Requirements

- A **quality requirement** is a product requirement that specifies a mandatory amount of a type of product quality.
  - Scalar (How Well or How Much?)
  - Based on Quality Model
  - Should be specified in requirements specifications.
  - Critically important drivers of the architecture
A **Quality Model** is a hierarchical model (i.e., a collection of related abstractions or simplifications) for formalizing the concept of the quality of a system in terms of its quality factors, quality subfactors, and quality measures.
Many Different Quality Factors

Quality Factor

- Development-Oriented Quality Factor
- Usage-Oriented Quality Factor

Quality Subfactor

- Capacity
- Configurability
- Dependability
- Efficiency
- Interoperability
- Performance
- Utility

Quality Factor

- Safety
  - Robustness
  - Security
- Defensibility
  - Correctness
  - Predictability
  - Operational Availability
  - Reliability
  - Stability
- Soundness
  - Correctness
  - Predictability
  - Operational Availability
  - Reliability
  - Stability
- Performance
  - Efficiency
  - Interoperability
  - Capacity
  - Utility

Quality Model

- is measured using a Quality Measure (Measurement Scale)
Components of a Quality Requirement

Quality Requirement

0..* Condition restricts applicability of

1..* System-Specific Quality Criterion provides evidence of existence of

1..* Measurement Threshold is measured against

describes aspect of quality of

is measured using a

Quality Factor

Quality Subfactor

Quality Model

specifies a minimum level of quality of the

defines the meaning of quality for the
Example Quality Requirement

- Hazard Prevention Safety Requirement:
  “Under normal operating conditions, a subway shall not move when one or more of its doors are open more often than an average of once every 10,000 trips.”

- Component Parts:
  - Condition:
    “Under normal operating conditions”
    (e.g., neither during maintenance nor fire)
  - Mandatory System-Specific Quality Criterion:
    “the subway shall move when one or more of its doors are open”
    (must define “move,” “doors,” and “open” somewhere)
  - Measurement Threshold:
    “not more often than an average of once every 10,000 trip.”
    (A trip is defined as an intentional move from one station to the next station)
Importance of Measurement Threshold

- Measurement Threshold is:
  - Critical
  - Difficult (but not impossible) to determine
  - Often left out of quality requirements
  - Needed to avoid ambiguity

- States *how much* quality is necessary (adequate)

- Enables architect to:
  - Determine if architecture is adequate
  - Make engineering trade-offs between competing quality factors

- Enables tester to determine test completion criteria
Safety Engineering Overview

Safety engineering is the engineering discipline within systems engineering that lowers the risk of accidental harm to valuable assets to an acceptable level to legitimate stakeholders.

Note:
- Engineering Discipline
- Systems Engineering (not just software)
- Risk
- Accidental Harm
- Harm to Valuable Assets
- Acceptable Level of Risk
- Legitimate Stakeholders
Basic Safety Concepts

- Safety as a Quality Factor of a Quality Model
- Safety Quality Subfactors
- Valuable Assets
- Accidental Harm to Valuable Assets
- Safety Events (Accidents, Incidents, and Hazardous Events)
- Hazards
- Safety Risks
- Goals, Policies, and Requirements
- Safeguards (Safety Mechanisms)
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 Events (Accidents, Incidents, and Hazardous Events)* are eliminated or their negative consequence mitigated
  - *Hazards* are eliminated or mitigated
  - *Safety risks* are kept acceptably low
  - The preceding problems are *prevented, detected, reacted to*, and possibly *adapted to*
Corresponding Safety Subfactors

Accidental Harm
- Safety Event
  - Hazard
  - Safety Risk
          ______________________
          |                      |
          |  Safety Problem Type  |
          |                      |
          ______________________

          ______________________
          |                      |
          |  Safety Solution Type |
          |                      |
          ______________________

Prevention
- Detection
- Reaction
- Adaptation

Safety
- Safety Subfactor
  - Quality Subfactor
    - Quality Factor
    - Quality Subfactor
      - Quality Measure
        (Measurement Scale)

is measured using a Quality Model
Valuable Assets

- A valuable **asset** is anything of *significant* value to a *legitimate stakeholder* that should be protected from *accidental* (or malicious) harm by the system.

![Diagram showing the relationship between System, Asset, Stakeholder, People, Property, Environment, Data, Software, Hardware, Facilities, and Money.]

*Image credit: Engineering Safety-Related Requirements for Software-Intensive Systems*
Accidental Harm

- **Harm** is any significant negative consequence to a valuable asset

- **Accidental harm** is any unauthorized unintentional (i.e., non-malicious) harm (i.e., due to an accident)
Harm Severity

- **Harm severity** is an appropriate categorization of the amount of harm.

- Harm severity categories can be standardized (ISO, military, industry-wide) or endeavor-specific.

- Harm severity categories need to be:
  - Clearly identified.
  - Appropriately and unambiguously defined.
Example Harm Severity Categories

- Example from the commercial aviation standard, *Software Considerations in Airborne Systems and Equipment Certification* (RTCA/DO 178B: 1992):
  - **Catastrophic:**
    - Failure conditions, which prevent the continued safe flight and landing of the aircraft
  - **Severe-Major:**
    - Failure conditions, which reduce the capability of the aircraft or the ability of the crew to cope with adverse operation conditions
    - Serious or potentially fatal injuries to some passengers
  - **Major:**
    - Failure conditions, which reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions
    - Discomfort and possible injury to the passengers
  - **Minor:**
    - Failure conditions, which do not cause a significant reduction in aircraft safety
  - **No-Effect:**
    - Failure conditions, which do not effect the operational capability of the aircraft or increase the crew’s workload
Safety-Related Events

- A safety event is any event with significant safety ramifications:
  - A accident trigger is a safety-related event that directly causes an accident.
  - A harm event is a safety-related event that causes significant harm.
  - A hazardous event is a safety-related event that causes the existence of a hazard (i.e., hazardous conditions).

- A network of safety events is any cohesive set of safety events:
  - An accident is a series of one or more related safety events causing actual non-malicious (i.e., accidental) harm to valuable assets.
  - A safety incident (a.k.a., close call, near miss) is a series of one or more related hazardous events that only by luck did not cause non-malicious actual harm.
Safety-Related Events and their Relationships
Importance of Accidents

- Accidents can have expensive and potentially fatal repercussions:
  - Ariane 5 Maiden Launch
    - Reuse of Ariane 4 software not matching Ariane 5 specification
  - Mars Climate Orbiter ($125 million)
    - English vs. Metric units mismatch
  - Mars Polar Lander
    - Missing requirement concerning touchdown sensor behavior
  - Therac–25 Radiation Therapy Machine
  - Patriot Missile Battery Misses SCUD
    - Missing availability (uptime) requirement
Poor Requirements Cause Accidents - 1

“The majority of software-related accidents are caused by requirements errors.”

“Software-related accidents are usually caused by flawed requirements. Incomplete or wrong assumptions about the operation of the controlled system can cause software related accidents, as can incomplete or wrong assumptions about the required operation of the computer. Frequently, omitted requirements leave unhandled controlled-system states and environmental conditions.”

Nancy G. Leveson, 2003

<http://www.safeware-eng.com/index.php/white-papers/accidents>
Poor Requirements Cause Accidents - 2

- Large percentage of accidents are caused by poor requirements:
  - “For the 34 (safety) incidents analyzed, 44% had inadequate specification as their primary cause.”
    Health and Safety Executive (HSE), *Out of Control: Why Control Systems Go Wrong and How to Prevent Failure* (2nd Edition), 1995

  - “Almost all accidents related to software components in the past 20 years can be traced to flaws in the requirements specifications, such as unhandled cases.”
Safety Event Likelihood Categories

- **Safety Event Likelihood Categorization** is an appropriate categorization of the probability that a safety event occurs.

- Safety event likelihood categories:
  - Can be standardized (ISO, military, industry-wide) or endeavor-specific.
  - Need to be identified and defined.

- Example safety event likelihood categories include:
  - Frequent
  - Probable
  - Occasional
  - Remote
  - Implausible

- Safety event likelihood categories need to be carefully and unambiguously defined.
Safety Hazards

- **Danger** (Defensibility) is one or more conditions, situations, or states of a system that in conjunction with condition(s) in the environment of the system can cause or contribute to the occurrence of a defense-related event:
  - **Hazard** (Safety) is a danger that can cause or contribute to the occurrence of an safety event.
  - **Threat** (Security and Survivability) is a danger that can cause or contribute to the occurrence of a security or survivability event (e.g., a security vulnerability combined with an attacker with means, motive, and opportunity).
Hazards and their Relationships

- Safety
- Security
- Survivability

Hazard

- Threat

Danger

- May result in Dangerous Event
- May cause Unauthorized Harm
- May occur to Valuable Asset

Condition (State)

- Relevant

Environment

System

Attacker

Involves the existence and profile of

Is responsible for protecting or not harming any

Engineering Safety-Related Requirements for Software-Intensive Systems
Example Hazard, Events, Harm, and Asset

Passenger = valuable asset
Passenger death = harm to asset

Elevator Starts Moving (normal event)
Door Unexpectedly Starts Opening (hazardous event)
Passenger Falls Out (accident trigger)
Passenger lands and is killed (harm event)

Passenger Falling (condition)
Door Not Closed (condition)
Elevator Moving (condition)
Moving Elevator with Door not Closed (hazard)

Time ———>
Hazard Analysis

- Hazard analysis usually implies the analysis of assets, harm, incidents, hazards, and risks.
- Hazard analysis often occurs multiple times before various milestones:
  - Preliminary Hazard Analysis
  - System Hazard Analysis
- Hazard analysis should probably be continuous.
- Fault Trees, Event Trees, and Cause/Effect Graphs can be used to determine potential causes and consequences of potential accidents and hazards.
Example Fault Tree (Cause of Failure)

- Passenger falls out of open door of moving taxi
  - Passenger inattentive and near taxi door
    - Door opens on moving taxi
    - Train starts moving with open door
      - Taxi door fails to close
      - Taxi starts to move
        - Taxi computer fails
  - Taxi door is unlocked
    - Taxi door lock fails unlocked
    - Taxi computer fails
      - Taxi door motor fails open
        - Taxi computer fails
          - Taxi door sensor fails closed
          - Taxi door lock sensor fails locked
  - Door motor opens taxi door
    - Warning is ineffective
      - A
  - Taxi motor fails open
    - Taxi computer fails
      - Taxi motor fails on
      - Taxi computer fails
Example Cause/Effect Graph

Legend
- Event
- Hazardous Event
- Accident
- State
- Hazard
- Harm

Guideway location identifier has failed

Taxi 2 location sensor has failed

Two taxis approach point where their individual guideways merge

Taxi 2 computer has failed

Taxi 2 transmitter has failed

Or

Taxi 2 power fails on

Taxi 2 brakes fail off

Right-of-way not requested

Lack of right-of-way ignored

Or

Taxi (2) without right-of-way fails to yield

Two taxis moving too fast to stop

Taxi (1) with right-of-way fails to yield once taxi (2) without right-of-way fails to yield

Failure to yield not observed

No warning sent

No warning received

Warning ignored

Taxi 1 transmitter has failed

Taxi 1 computer has failed

Or

Merging taxis collide

Or

Guideway is damaged

Or

Tax(s) crash off guideway into habitat

Animal(s) are harmed

Pedestrian(s) are harmed

Patrons' vehicles are harmed

Or

Tax(s) crash off guideway into parking lot

Tax(s) are harmed

Passenger(s) are injured

Passenger(s) are damaged

Tax(s) are destroyed

Passenger(s) are killed
Defensibility Risks including Safety Risks

- **Risk** is the combination of the severity of harm to a valuable asset with either the likelihood that the harm will occur or else the level of software control.

- **Harm severity** is usually set *conservatively* to the maximum credible category of harm.

- The likelihood of harm is the likelihood of danger multiplied by either the likelihood that the danger results in a harm-causing event (e.g., accident or attack).
Safety Integrity Levels (SILs)

- **Safety integrity levels (SILs)** are categories of requirements based on their associated safety risk level.

- SILs can be determined for:
  - Individual requirements.
  - Groups of related requirements (e.g., features or functions).

- SILs should be appropriately, clearly, and unambiguously defined.
Example Safety Integrity Levels (SILs)

- **Intolerable:**
  The risk associated with the requirement(s) is totally unacceptable to the major stakeholders. The requirement(s) *must* therefore be deleted or modified to lower the associated risk.

- **Undesirable:**
  The risk associated with the requirement(s) is so high that major (e.g., architecture, design, implementation, and testing) steps should be taken to lower the risk (e.g., risk mitigation and risk transfer) to lower the risk.

- **As Low As Reasonably Practical (ALARP):**
  Reasonable practical steps should be taken to lower the risk associated with the requirement(s).

- **Acceptable:**
  The risk associated with the requirement(s) is acceptable to the major stakeholders and no additional effort must be taken to lower it.
Example Safety Risk Matrix

- Safety Risk Matrix defines safety risk (and SIL) as a function of:
  - Harm severity
  - Accident/hazard frequency of occurrence.

<table>
<thead>
<tr>
<th>Safety Risks / Safety Integrity Levels (SILs)</th>
<th>Frequency of Accident / Hazard Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harm Severity</td>
<td>Frequent</td>
</tr>
<tr>
<td>Catastrophic</td>
<td>Intolerable</td>
</tr>
<tr>
<td>Critical</td>
<td>Intolerable</td>
</tr>
<tr>
<td>Major</td>
<td>Undesirable</td>
</tr>
<tr>
<td>Minor</td>
<td>Undesirable</td>
</tr>
<tr>
<td>Negligible</td>
<td>ALARP</td>
</tr>
</tbody>
</table>
Safety Goals

- **Safety Goals** are high-level stakeholder desires regarding safety:
  - “The system must be safe.”
  - “There can be no serious accidents.”
  - “The system will never kill or injure its users.”

- Goals are typically unrealistic and unverifiable (i.e. impossible to guarantee 100% safety).

- Goals are *not* requirements.

- A *major* problem is safety goals that are specified as if they were verifiable requirements.
Safety Policies

● **Policy** – a strategic *process* decision that establishes a desired goal.

● **Safety policy** – a policy that enables the achievement of one or more *safety goals*:
  - “The overall responsibility for safety must be identified and communicated to all stakeholders.”
  - “A hazard analysis shall be performed during early in the project.”
  - “All users will have safety training.”

● Safety policies are collected into safety policy documents.

● In practice, safety policies are confused with safety requirements, and conversely policy documents may sometimes include safety requirements.
Safety-Related Requirements

- A safety-related requirement is a product requirement that has significant safety ramifications.

- Safety-related requirements include:
  - Safety Requirements
  - Safety-Significant Requirements
  - Safety Subsystem Requirements
  - Safety Constraints
Safeguards (Safety Control, Safety Mechanism)

- A **safeguard** is a kind of defense that helps fulfill a safety-related requirement and thereby eliminates or reduces the impact of a safety vulnerability.

- A safeguard is a part of the system (e.g., component, procedure, training)

- Only relevant to requirements if specified as safety constraints.
Safety-Related Requirements

- Safety Requirements
- Safety-Significant Requirements
- Safety Subsystem Requirements
- Safety Constraints
Safety-Related Requirement Definitions

○ **Safety-Related Requirements** are any system requirements having *significant safety ramifications*:
  - **Safety Requirements** are requirements that specify mandatory minimum safety levels in terms of pairs of subfactors of the safety quality factor.
  - **Safety-Significant Requirements** are *non-safety primary mission* requirements with significant safety ramifications.
  - **Safety Subsystem Requirements** are requirements for safety subsystems (as opposed to primary mission requirements).
  - **Safety Constraints** are constraints intended to ensure a minimum level of safety.
Types of Requirements

- Process Requirements
- Product Requirements
  - Functional Requirements
  - Non-Functional Requirements
- System/Subsystem Requirements
  - Data Requirements
  - Interface Requirements
  - Quality Requirements
- Main Mission Requirements
  - Constraints
    - Defensibility Requirements
  - Safety Constraints
- Software Requirements
  - Stakeholder (Business) Requirements
- Hardware Requirements
  - Safety Subsystem Requirements
  - Security Subsystem Requirements
- System/Subsystem Requirements
  - Safety Requirements
  - Security Requirements
  - Survivability Requirements

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

Safety-Intolerable Requirements
SIL = 5

Safety-Critical Requirements
SIL = 4

Safety-Major Requirements
SIL = 3

Safety-Moderate Requirements
SIL = 2

Safety-Minor Requirements
SIL = 1

Safety-Significant Requirements
SIL ≥ 1

Safety-Independent Requirements
SIL = 0

Non-Safety Quality Requirements

Safety Event Requirements

Hazard Requirements

Safety Risk Requirements

Protection of Valuable Assets Requirements

Detection of Safety Incidents Requirements

Reaction to Safety Incidents Requirements

Adaptation to Safety Incidents Requirements

Safety Constraints

Main Mission Requirements

System Requirements

Data Requirements

Functional Requirements

Interface Requirements

Quality Requirements

Engineering Safety-Related Requirements for Software-Intensive Systems
[Pure] Safety Requirements

- A safety requirement is a kind of quality (defensibility) requirement because safety is a kind of defensibility. (Safety requirements are like security requirements.)

- Safety requirements specify minimum required amounts of:
  - Safety
  - Two quality subfactors of safety:
    - Defensibility Problem Type: Accidental Harm, Safety Event, Hazard, Safety Risk
    - Defensibility Solution Type: Prevention, Detection, Reaction, Adaptation
A quality requirement is composed of conditions, a system-specific criterion, and a required measurement threshold.

- **Condition**: restricts applicability of the system-specific criterion.
- **System-Specific Criterion**: describes aspect of quality of the system.
- **Measurement Threshold**: must meet or exceed the required measurement threshold.
- **Quality Requirement**: specifies a minimum level of quality of the system.
- **Quality Factor**: defines the meaning of quality for the system.
- **Quality Subfactor**: provides evidence of existence of the system-specific criterion.
- **Quality Model**: is measured using a quality measure.
Safety Requirements

Safety Requirements are a kind of quality requirement.

- Safety Requirements are a kind of quality requirement.

1..* 1..* 1..*

Safety Subfactor provides evidence of existence of System-Specific Criterion, which must meet or exceed the Measurement Threshold.

Condition restricts applicability of System-Specific Criterion.

System describes aspect of safety of Safety.

Safety defines the meaning of quality for the Quality Model.

Quality Measure (Measurement Scale) is measured against the Measurement Threshold, which is measured using a Quality Measure (Measurement Scale).

Engineering Safety-Related Requirements for Software-Intensive Systems
Based on Safety Subfactors

Accidental Harm
Safety Event
Hazard
Safety Risk

Prevention
Detection
Reaction
Adaptation

Safety Problem Type
Safety Solution Type

Safety
Safety Subfactor

Quality Factor
Quality Subfactor

is measured using a

Quality Measure
(Measurement Scale)

Quality Model
## Sixteen Types of Safety Requirements

<table>
<thead>
<tr>
<th></th>
<th>Accidental Harm</th>
<th>Safety Event</th>
<th>Hazard</th>
<th>Safety Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prevention</strong></td>
<td>Prevent accidental harm</td>
<td>Prevent safety event</td>
<td>Prevent existence of hazard</td>
<td>Prevent existence of safety risk</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td>Detect accidental harm</td>
<td>Detect safety event</td>
<td>Detect existence of hazard</td>
<td>Detect existence of safety risk</td>
</tr>
<tr>
<td><strong>Reaction</strong></td>
<td>React to accidental harm</td>
<td>React to safety event</td>
<td>React to existence of hazard</td>
<td>React to existence of safety risk</td>
</tr>
<tr>
<td><strong>Adaptation</strong></td>
<td>Adapt due to accidental harm</td>
<td>Adapt due to safety event</td>
<td>Adapt due to existence of hazard</td>
<td>Adapt due to existence of safety risk</td>
</tr>
</tbody>
</table>
Example Safety Requirements

- “With 99% confidence, the system shall not cause more than X amount of accidental harm per year.”
- “With 99% confidence, the system shall not cause more than X safety incidents (accidents, near misses) per passenger mile traveled.”
- “With 99% confidence, the system shall not under normal conditions cause hazard X to exist more than Y percent of the time.”
- “The system shall not allow a safety risk level of X to exist.”
- “The system shall detect accidents of type X at least Y percent of the time.”
- “Upon detecting an accident of type X, the system shall react by performing Y at least Z percent of the time.”
Safety-Significant Requirements

- Are identified based on safety (hazard) analysis

- Subset of non-safety requirements:
  - Functional Requirements
  - Data Requirements
  - Interface Requirements
  - Non-safety Quality Requirements
  - Constraints

- Safety Integrity Level (SIL) is not 0:
  - May have minor safety ramifications
  - May be safety-critical
  - May have intolerable safety risk
SILs and SEALs

- **Safety Integrity Level (SIL)** – a category of required safety for safety-significant requirements.

- **Safety Evidence Assurance Level (SEAL)** – a category of required evidence needed to assure stakeholders (e.g., safety certifiers) that the system is sufficiently safe (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)

- **SILs** do not map 1-1 to **SEALs**.
Safety-Significant Requirements (cont)

- Require enhanced Safety Evidence Assurance Levels (SEALs) including more rigorous development process (including better requirements engineering):
  - Formal specification of requirements
  - Fagan inspections of requirements

- Too often SEALs only apply to design, coding, and testing:
  - Safe subset of programming language
  - Design inspections
  - Extra testing
Example Safety-Significant Requirements

- Requirements for controlling subway doors:
  - Keep doors closed when moving
  - Not crush passengers

- Requirements for firing missiles from military aircraft:
  - When to arm missile
  - Controlling doors providing stealth capabilities
  - Detecting weight-on-wheels

- Requirements for chemical plant:
  - Mixing and heating chemicals
  - Detecting temperature and pressure
Safety Subsystem Requirements

- **Safety Subsystem Requirements** are requirements for safety subsystems (as opposed to primary mission requirements).

- Subsystems or components 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 systems are safety-related.
Example Safety 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.”
Safety 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 standards often mandate best practices as safety constraints.
Example 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.”
Recommended Combined Method

- How should safety-related requirements be engineered?
- Need to combine (include) tasks, teams, and work products from:
  - Requirements Engineering
  - Safety Engineering
- What is appropriate?
  - What tasks need to be performed?
  - Who should perform them?
  - What collaboration is appropriate/necessary?
  - What work products should be produced?
  - Where do requirements work products fit in?
Basic Safety Engineering Tasks

- Six basic safety engineering tasks.
- Not all directly related to engineering safety-related requirements.
- Some tasks are:
  - Up front
  - Ongoing
  - Event driven

![Diagram of safety engineering tasks]

- Safety Program Planning
- Safety Analysis
- Safety Monitoring
- Safety Compliance Assessment
- Safety Event Investigation
- Safety Certification
Overlap between RE and SE

- Requirements Engineering includes:
  - Requirements Identification
  - Requirements Analysis
  - Requirements Specification

- Safety Engineering includes Safety Analysis.

![Diagram showing the overlap between Requirements Engineering and Safety Engineering]

Engineering Safety-Related Requirements for Software-Intensive Systems
Safety & Requirements Engineering Interface

- Set Safety Goals
- Safety Program Planning
- Safety Significance Analysis
- Safety Analysis
- Safety Control Analysis
- Safety Goals
- Safety Program Plan
- Application Visioning
- Application Vision Statement (ConOps)
- System Requirements Specification
- Safety-Related Requirements
  - Safety Significant Requirements
  - Safety Requirements
  - Safety Subsystem Requirements
  - Safety Constraints
- System Requirements
- Requirements Specification
- Requirements Analysis
- Requirements Identification

Requirements Team

Engineering Safety-Related Requirements for Software-Intensive Systems
Safety Program Planning

Inputs:
- Subject Matter Experts
- Stakeholders

Subtasks:
- Set Safety Policy
- Set Safety Goals
- Determine Safety Categories
- Develop Safety Program

Outputs:
- Safety Policy
- Safety Goals
- Asset Value Categories
- Harm Severity Categories
- Hazard Likelihood Categories
- Safety Event Likelihood Categories
- Safety Risk Matrix
- Safety Integrity Levels (SILs)
- Safety Evidence Assurance Levels
- Safety Program Plan

Flow:
- Safety Team performs
- Project Documentation (RFP, Contract, ConOps)
- Legacy Documentation
- Generic/Reusable Safety Categories
- Standard/Reusable Safety Integrity Levels (SILs)
- Standard/Reusable Safety Evidence Assurance Levels (SEALs)

This diagram illustrates the process of safety program planning, detailing inputs, subtasks, and outputs involved in setting and determining safety policies, goals, and categories for software-intensive systems.
Safety Analysis Yields Safety-Related Rqmts
Safety Analysis Requires Collaboration

Safety Analysis

Asset Analysis
- Asset Safety Requirements

Safety Event Analysis
- Safety Event Requirements

Hazard Analysis
- Hazard Safety Requirements

Safety Risk Analysis
- Safety Risk Safety Requirements

Safety Significance Analysis

Safety Control Analysis

Prelim. Safety Analysis

System Safety Analysis

Architecture Team

Supports

Helps perform

Performs

Requirements Team
Asset Analysis

Subject Matter Experts

Stakeholders

provide input during

Project Documentation (RFP, Contract, ConOps)

Generic / Reusable Asset Lists

Generic / Reusable Asset / Harm Tables

Standard / Reusable Harm Severity Categories

Standard / Reusable Asset / Harm Requirements

Safety Team

performs

Asset Identification

Value Analysis

Harm Analysis

Asset / Harm Requirements Production

helps perform

Asset List

Asset Value and Harm Table

Asset / Harm Requirements

Requirements Team

Engineering Safety-Related Requirements for Software-Intensive Systems
Safety Event Analysis

Subject Matter Experts

Safety Team

provide input during

performs

Project Documentation (RFP, Contract, ConOps)

Generic / Reusable Safety Event Type Lists

Asset Value and Harm Table

Harm Severity Categories

Generic / Reusable Safety Event / Harm Tables

Standard / Reusable Safety Event Likelihood Categories

Safety Event Likelihood Categories

Safety Event Type Identification

Safety Event Harm Analysis

Safety Event Likelihood Analysis

Safety Event Requirements Production

Safety Event Requirements

Stakeholders

Engineering Safety-Related Requirements for Software-Intensive Systems

77
Hazard Analysis

Hazard Identification

Hazard Categorization

Hazard Cause Analysis

Root Cause Analysis

Network of Causes Analysis

Common Cause Analysis

Hazard Cause & Effect Diagrams and Tables

Fault/Event Trees

HAZOP/ FEMA

Hazard Likelihood Analysis

Hazard Reporting

Hazard Requirements Production

Hazard List

Hazard Categories

Hazard Likelihood Table

Hazard Reports

Hazard Safety Requirements

Safety Team

Subject Matter Experts

Stakeholders

Project Documentation (System Architecture)

Generic / Reusable Hazard Lists

Standard / Reusable Hazard Categories

Standard / Reusable Hazard Likelihoods

Generic / Reusable Hazard Safety Requirements

Requirements Team

Engineering Safety-Related Requirements for Software-Intensive Systems
Safety Risk Analysis

Subject Matter Experts

Stakeholders

Safety Team

Harm Severity Categories

Generic / Reusable Safety Risk Matrices

Standard / Reusable Safety Risk Categories

Safety Event / Hazard Likelihood Categories

Standard / Reusable Safety Integrity Levels

Standard / Reusable Safety Evidence Assurance Levels (SEALs)

Safety Risk Categories

Safety Risk Determination

Safety Risk Estimation

Safety Risk Requirements Production

Safety Risks

Accident Type Safety Risk Table

Hazard Safety Risk Table

Safety Risk Requirements

Helps perform Requirements Team

Engineering Safety-Related Requirements for Software-Intensive Systems
Safety-Significance Analysis

Safety Team

Subject Matter Experts

performs

Requirements Team

helps perform

Safety-Significant Requirements Identification

Categorization of Safety-Significant Requirements

Identify Safety-Significant Functional Requirements

Identify Safety-Significant Data Requirements

Identify Safety-Significant Interface Requirements

Identify Safety-Significant Non-Quality Requirements

Safety Integrity Level (SiL) Allocation

Safety Evidence Assurance Level (SEAL) Allocation

Functional Requirements

Data Requirements

Interface Requirements

Non-Safety Quality Requirements

Safety Risk Tables

Safety Integrity Levels

Stakeholders
Safety Control Analysis

Safety Team supports Architecture Team

Safety Team performs

Stakeholders provide input during

Subject Matter Experts provide input during

Safety-Significant Requirements
Safety Analyses
System Architecture

Safety Control Identification

Safety Subsystem Identification

Safety Subsystem Requirements Allocation

Safety Constraints Determination

Updated System Architecture

Safety Controls

Safety Subsystem Requirements

Safety Constraints

Requirements Team

Engineering Safety-Related Requirements for Software-Intensive Systems
Conclusion

- 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-related requirements:
  - Safety Requirements
  - Safety-Significant Requirements
  - Safety Subsystem Requirements
  - Safety Constraints

- They have different forms (structures, contents).
- They need to be identified, analyzed, and specified differently.
Conclusion (3)

- The requirements engineering and safety engineering processes need to be:
  - Properly interwoven.
  - Consistent with each other.
  - Performed collaboratively and in parallel (i.e., overlapping in time).
Final Thoughts

- Full day tutorial with examples and student exercises to be given at ICSE’06 in Shanghai (22 May 2006).
- Look for my upcoming book by the same title.
- For more information, check out this repository of over 1,100 free open-source reusable method components including many on safety at www.opfro.org.