

Safety of rehabilitation robots

Jérémie Guiochet
LAAS CNRS, Toulouse, France

jeremie.guiochet@laas.fr

www.laas.fr/~guiochet

Outline

- ▶ Chapter 1 - Introduction
- ▶ Chapter 2 - Dependability of systems
- ▶ Chapter 3 - Risk management concepts
- ▶ Chapter 4 –Three risk analysis techniques into details
(HAZOP, FMECA, FTA)
- ▶ Chapter 5 - A scenario based risk analysis approach
(UML-HAZOP)



Chapter 1. Introduction

Windows

A fatal exception 0E has occurred at 0020:C0011E36 in UXD UMM(01) + 00010E36. The current application will be terminated.

- * Press any key to terminate the current application.
- * Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue

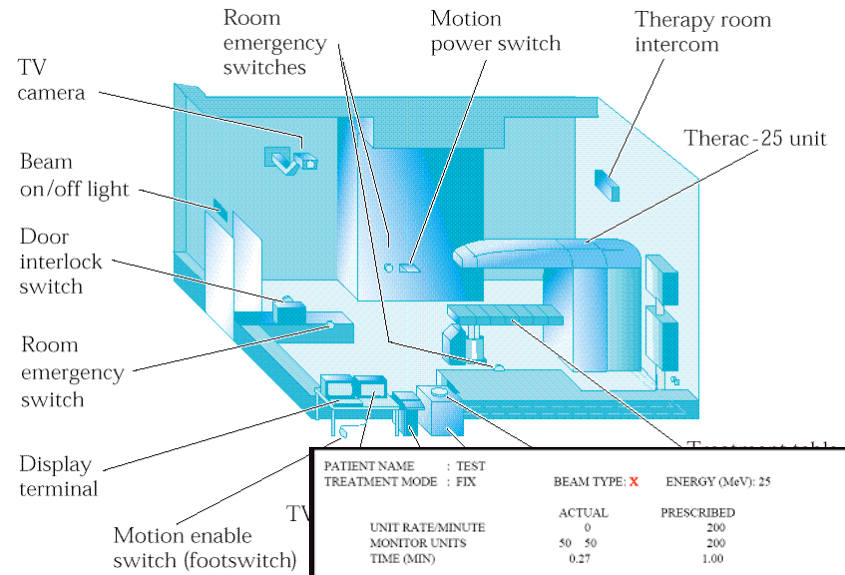

```

...
declare
  vertical_veloc_sensor: float;
  horizontal_veloc_sensor: float;
  vertical_veloc_bias: integer;
  horizontal_veloc_bias: integer;
...
begin
  declare
    pragma suppress(numeric_error, horizontal_veloc_bias);
  begin
    sensor_get(vertical_veloc_sensor);
    sensor_get(horizontal_veloc_sensor);
    vertical_veloc_bias := integer(vertical_veloc_sensor);
    horizontal_veloc_bias := integer(horizontal_veloc_sensor);
  ...
  exception
    when numeric_error => calculate_vertical_veloc();
    when others => use_irs1();
  end;
end irs2;

```



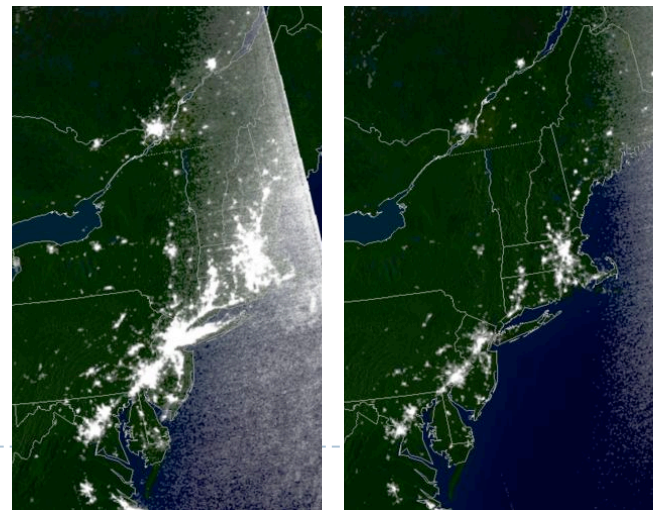
Ariane 5
1996



Therac 25
1985

PATIENT NAME	: TEST	BEAM TYPE: X	ENERGY (MeV): 25
TREATMENT MODE	: FIX		
UNIT RATE/MINUTE	0	ACTUAL	PRESCRIBED
MONITOR UNITS	50.50	0	200
TIME (MIN)	0.27	0.27	1.00
GANTRY ROTATION (DEG)	0.0	0	0 VERIFIED
COLLIMATOR ROTATION (DEG)	359.2	359	359 VERIFIED
COLLIMATOR X (CM)	14.2	14.3	14.3 VERIFIED
COLLIMATOR Y (CM)	27.2	27.3	27.3 VERIFIED
WEDGE NUMBER	1	1	1 VERIFIED
ACCESSORY NUMBER	0	0	0 VERIFIED
DATE	: 84-OCT-26	SYSTEM	: BEAM READY OP. MODE : TREAT AUTO
TIME	: 12:55: 8	TREAT	: TREAT PAUSE X-RAY 173777
OPR ID	: T25V02-R03	REASON	: OPERATOR COMMAND:

USA Blackout
2003



Critical systems

▶ Threats :

- ▶ Hardware (failures of electronic components e.g. sensors)
- ▶ Mechanical (failure of a mechanical part e.g. jamming)
- ▶ Software (presence of bugs and failures e.g. reboot)
- ▶ Humans (human errors e.g. unexpected comportment)
- ▶ Environment (hazardous conditions e.g. no light)

▶ Understand threats causes and consequences and treat them:

- ▶ Prevent : use methods and tool to prevent the presence of threats
- ▶ Eliminate : find threats in the system and eliminate them
- ▶ Forecast : estimate causes/consequences of threats
- ▶ Tolerate : develop the system to tolerate some threats



Critical systems

- ▶ Robotic systems now belongs to critical systems



daVinci



Three Laws of Robotics – Isaac Asimov (1950)

- ▶ 1. A robot must not harm a human being, nor through inaction allow one to come to harm.
- ▶ 2. A robot must always obey human beings, unless that is in conflict with the first law.
- ▶ 3. A robot must protect itself from harm, unless that is in conflict with the first or second laws.



Rehabilitation robotics = critical systems?

- ▶ Potential to harm ?
- ▶ Types of systems
 - ▶ Mobility aid
 - ▶ Manipulation aid
 - ▶ Therapeutic aid



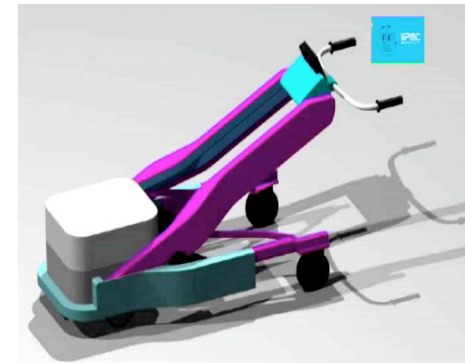
Mobility Aids: Potential/Future

- ▶ > 5 million wheelchair users in the U.S.
- ▶ Safe and reliable mobile robotic assistive devices.
- ▶ Intelligent homes, buildings communicate with smart wheelchairs
- ▶ Extend to other assistive devices (e.g. scooters)

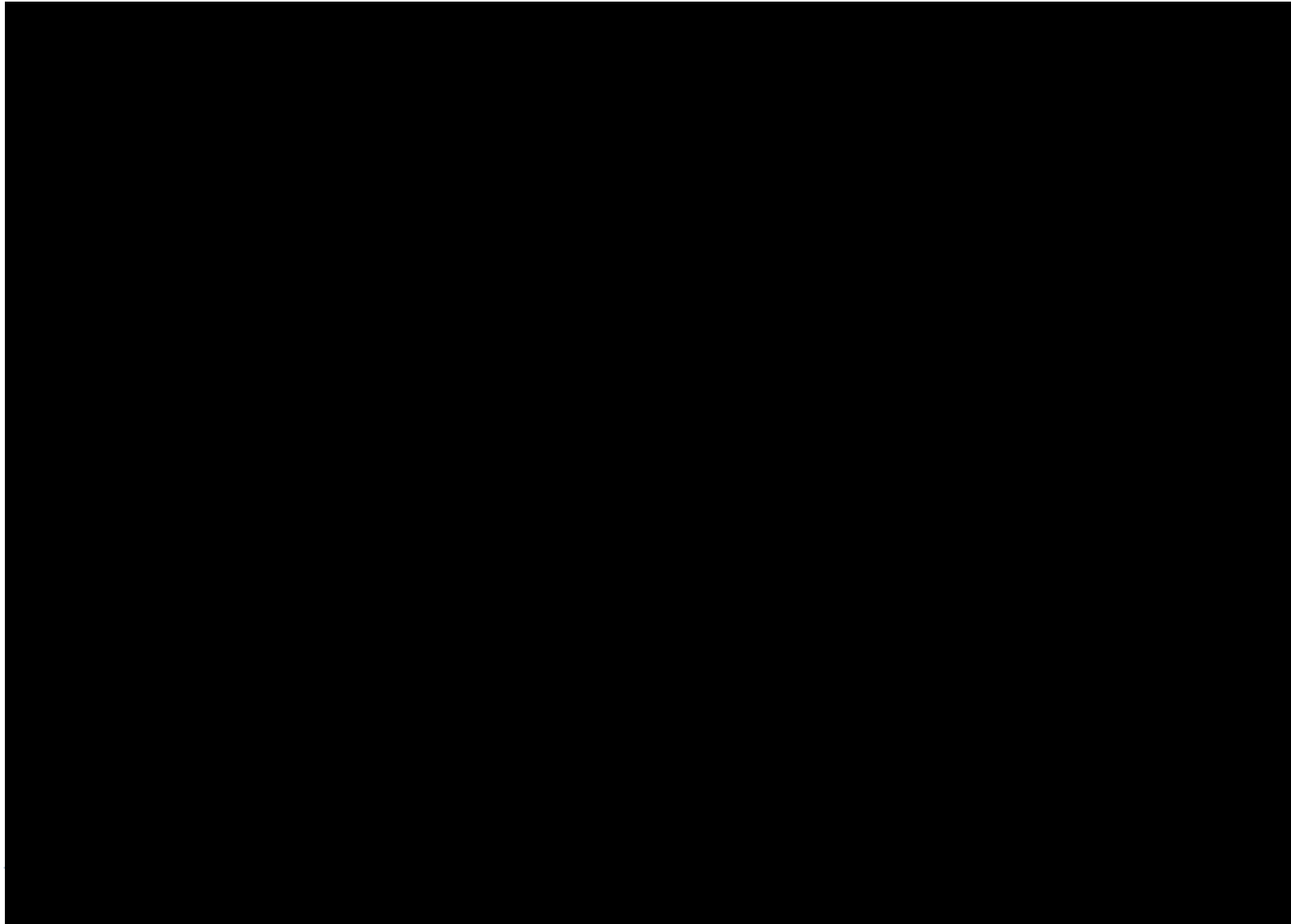


Mobility Aids: State of the Art

- ▶ All-terrain chairs
 - ▶ Tracked Systems (Ishimatsu, Hirose)
- ▶ Wheeled/legged systems (Krovi, Kumar, Wellman)
- ▶ Walkers (SmartWalker, Robuwalker, NurseBot)
- ▶ Prosthetics legs systems ()
- ▶ Other systems (e.g. guides for blind people)



Example : RobuWalker



Mobility Aids: Significant Accomplishments

- ▶ Intelligent chairs
- ▶ Automated navigation tasks/behaviors
- ▶ Input modalities to mobile systems provide access to users who may lack fine motor control
- ▶ Gesture recognition, voice command, vision-based interaction, sip and puff devices
- ▶ Ability to drive on all terrains (stairs/curbs)
- ▶ User monitoring (geolocalization, physiological parameters, etc.)



Manipulation Aids

- Prosthetics arms
- Feeders
- Multifunction manipulators



Manus (Exact Dynamics)



PROVAR Assistive
Robot System (VA Palo Alto)



Raptor (Applied Resources)



Winsford Feeder



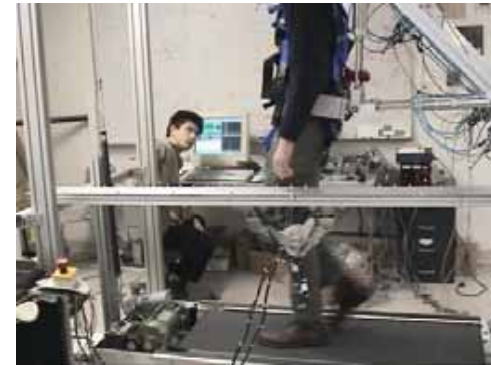
Therapeutic Aids



Schematic (Stanford)



Lokomat (Hocoma)



UCLA/UCI



MIT Manus



Challenges (Research)

- ▶ Identify movement training algorithms that maximize motor learning and neural recovery, by combining robotics, brain imaging, and neurocomputational modelling
- ▶ Automated tools to aid the diagnosis and assessment process (parametric to enable customization)
- ▶ Safe and effective human-robot interaction for hands-off assistive robotics
 - ▶ find, track, follow, and understand the activity of the patient
 - ▶ provide appropriate feedback
 - ▶ motivate, engage patient

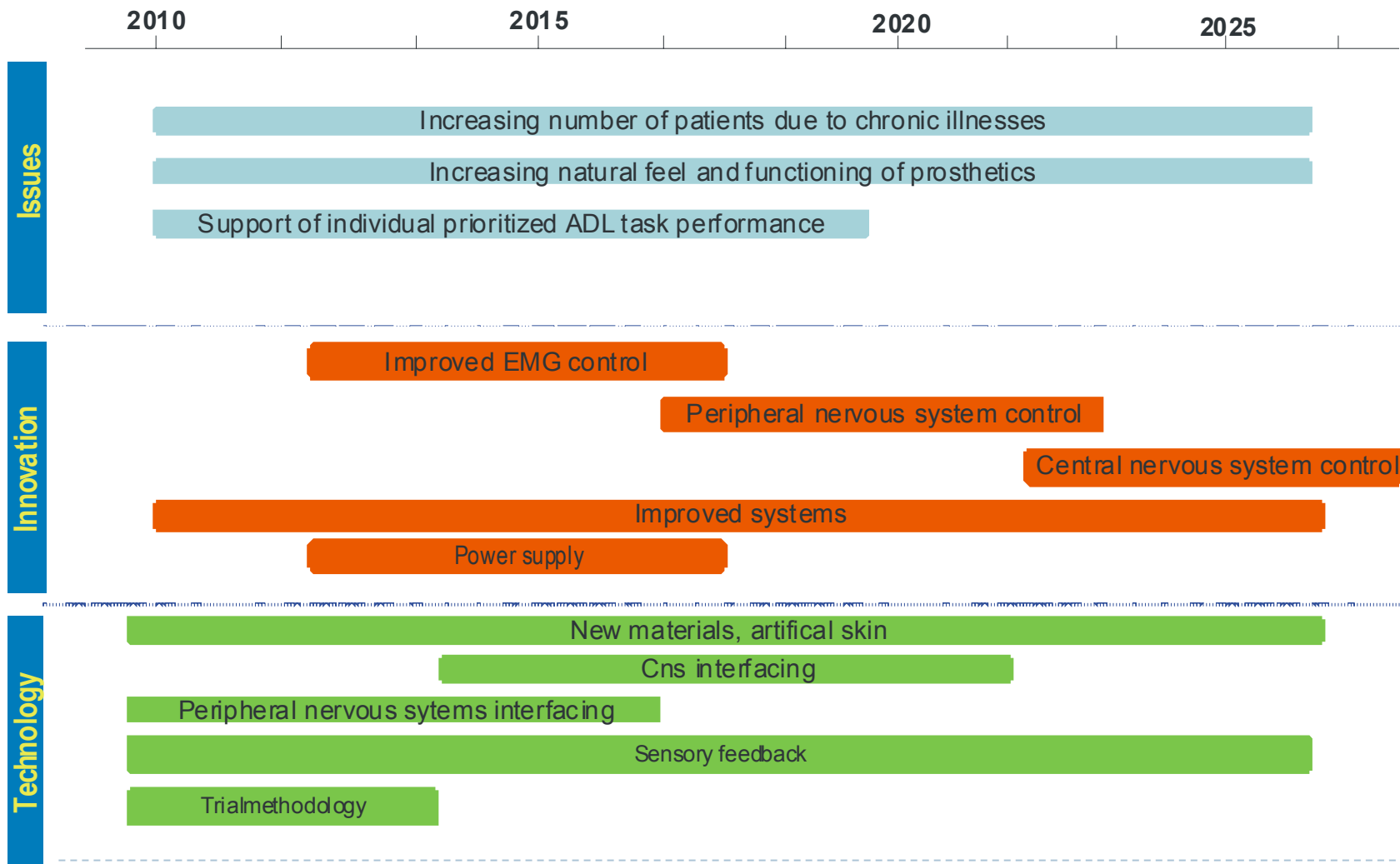


Challenges (Technology)

- ▶ Develop combined therapeutic/assistive rehabilitation robotic systems that are lightweight enough to be worn while performing activities of daily living.
- ▶ Inexpensive, safe, back-driveable robots

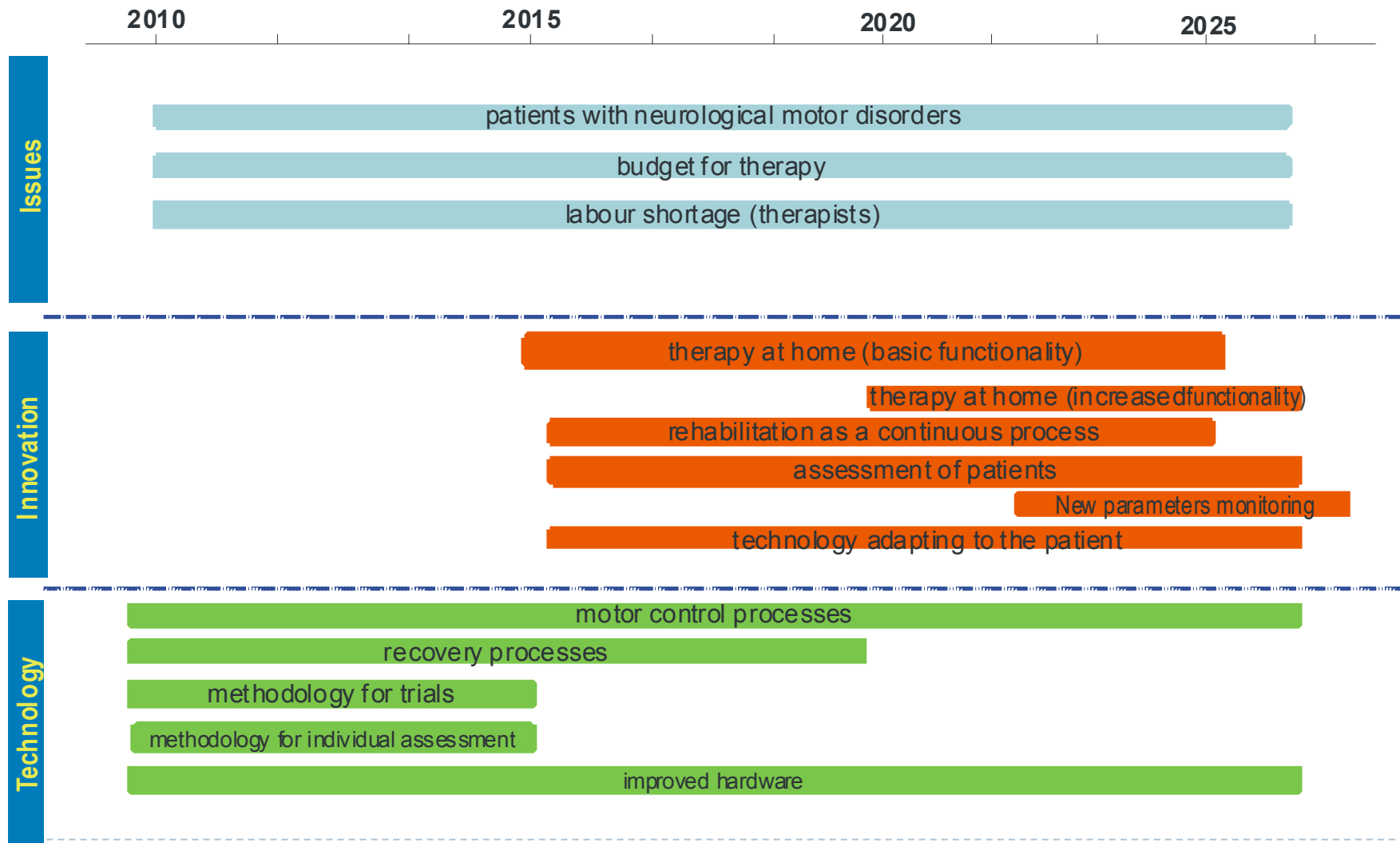


Roadmap of the domain “Intelligent prosthetics”



Source : Rehabilitation Robotics in Robotics for Healthcare; A Roadmap Study for the European Commission G. Gelderblom, M. De Wilt, G. Cremers, A. Rensma, IEEE 11th International Conference on Rehabilitation Robotics, Japan, June 23-26, 2009

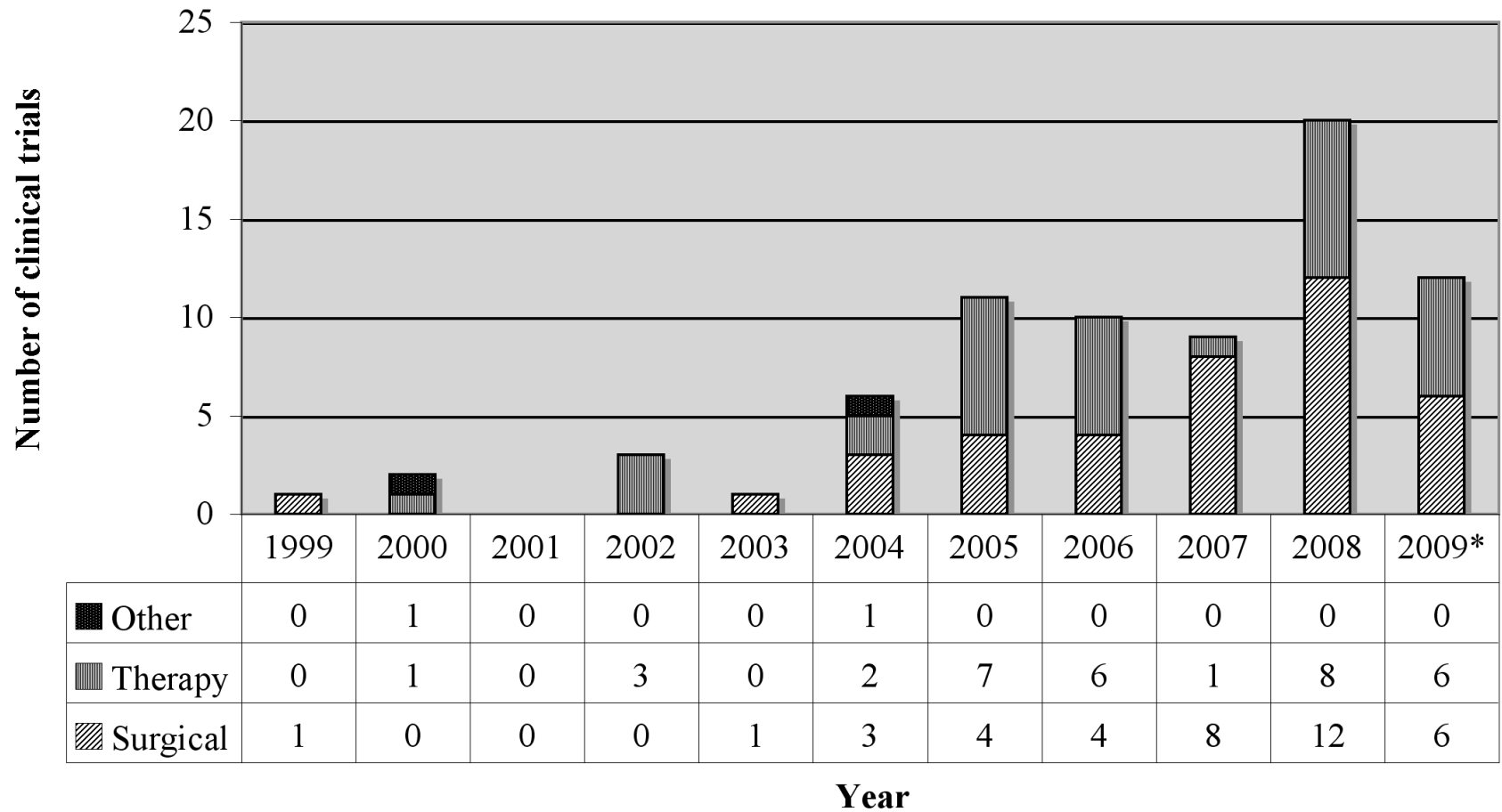
Roadmap of the domain “Robotised motor coordination analysis and therapy »



Source : Rehabilitation Robotics in Robotics for Healthcare; A Roadmap Study for the European Commission G. Gelderblom, M. De Wilt, G. Cremers, A. Rensma, IEEE 11th International Conference on Rehabilitation Robotics, Japan, June 23-26, 2009

The number of clinical trials found in ClinicalTrial.gov which used robots as the experimental intervention. Year indicates the study's start date. Data as of June 9, 2009.
 Source : *Towards Establishing Clinical Credibility for Rehabilitation and Assistive Robots*; Katherine M. Tsui and Holly A. Yanco, 2009

Clinical Trials Featuring Robots (1999-2009)



Preliminary Hazard Identification

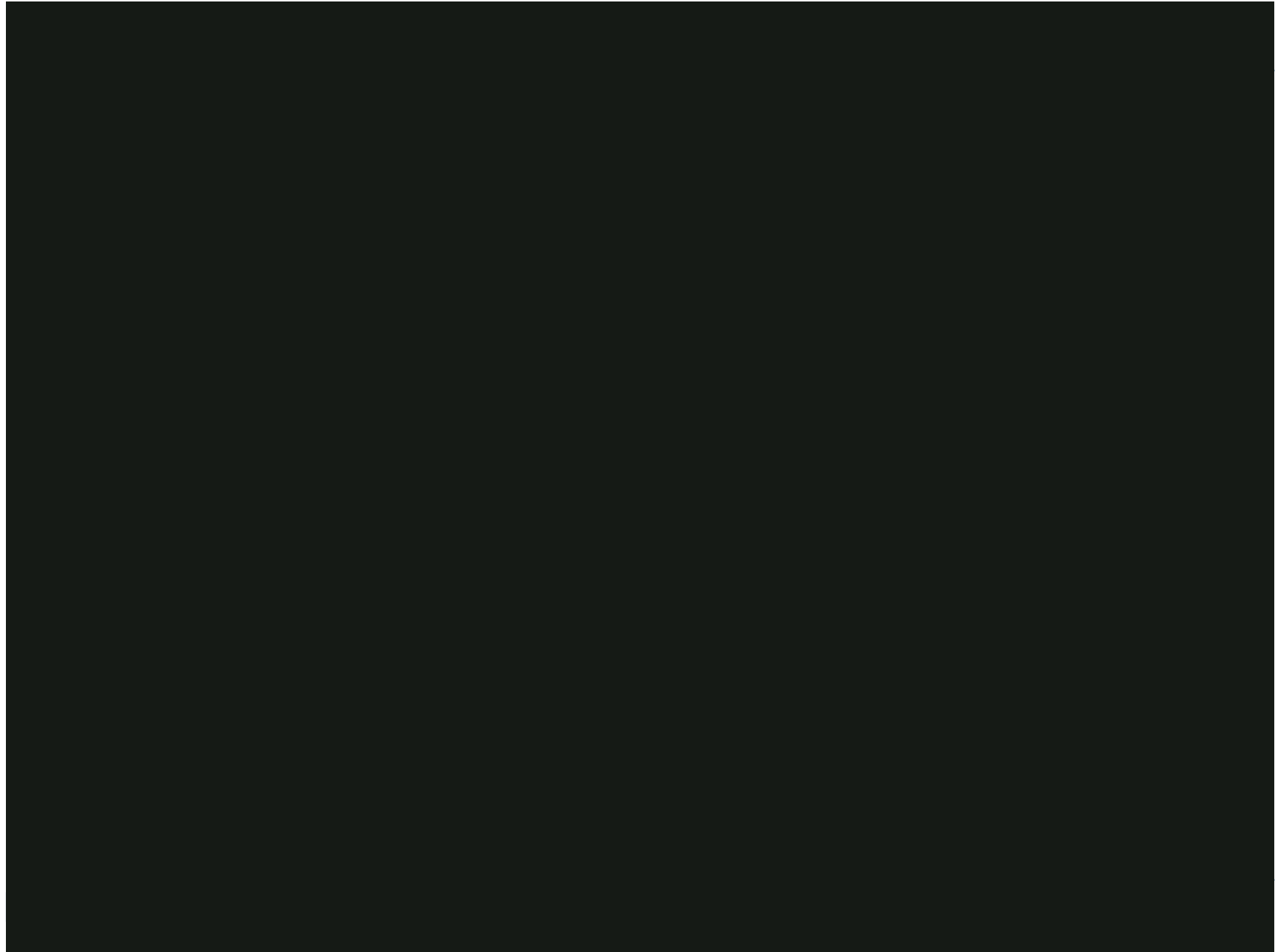
- ▶ Considering the three following case studies, what are the main hazards ?
 - ▶ Therapeutic aid : Training (Lokomat)
 - ▶ Mobility aid : Autonomous wheelchair (Sabre)
 - ▶ Manipulation aid : Feeder (Meal Buddy)



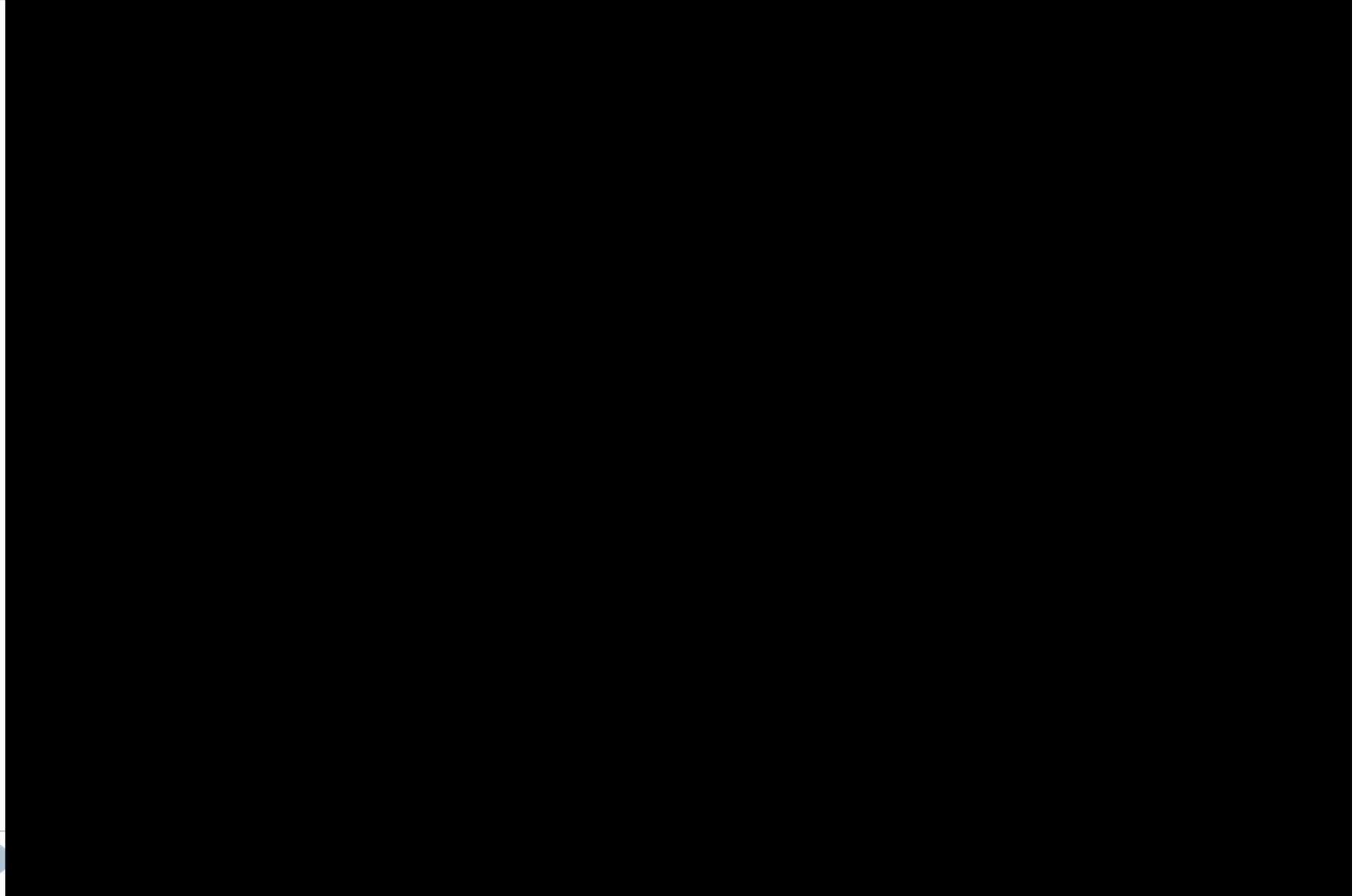
Therapeutic aid : Training (Lokomat)



Mobility aid :Autonomous wheelchair (Sabre)



Manipulation aid: Feeder (Meal Buddy)



First conclusions

- ▶ **Ensure safety:**
 - ▶ A process to analyse safety
 - ▶ Who ? What ? When ?
 - ▶ Methods for safety analysis
 - ▶ Models for quantitative or qualitative analysis
 - ▶ Tools for safety analysis
 - ▶ Computer assisted safety analysis for the application of methods
 - ▶ Technologies to improve safety
 - ▶ From industrial robots
 - ▶ From other domains
 - ▶ New technologies



Standards from machinery to advanced robots

- ▶ Safety of machinery
- ▶ Safety of industrial robots
- ▶ Safety of advanced robots
 - ▶ Generic safety standards
 - ▶ Safety of medical robots
 - ▶ Safety of rehabilitation robots



Safety of machinery

Machinery Directive
98/37/EC

ISO/IEC Guide 51

Safety of machinery

Basic concepts, general principle for design (ISO 12100)
principles of risk assessment principles of risk assessment (ISO 14121)

Interlocking devices associated with guards (ISO 14119)
Guards (ISO 14120)

Safety-related parts of control systems (ISO 13849-1,-2)

Safety distances to prevent danger zones
being reached by the upper limbs (ISO 13852)

Prevention of unexpected start-up (ISO 14118)

Two-hand control devices (ISO 13851)

Pressure-sensitive protective devices (ISO 13856)

Permanent means of access to machinery (ISO 14122)

precise technologies for each type of machinery

Industrial Robots - Relevant Robot Safety Standards



▶ Europe :

- ▶ ISO 10218 Robots for industrial environments – Safety requirements

▶ Other countries

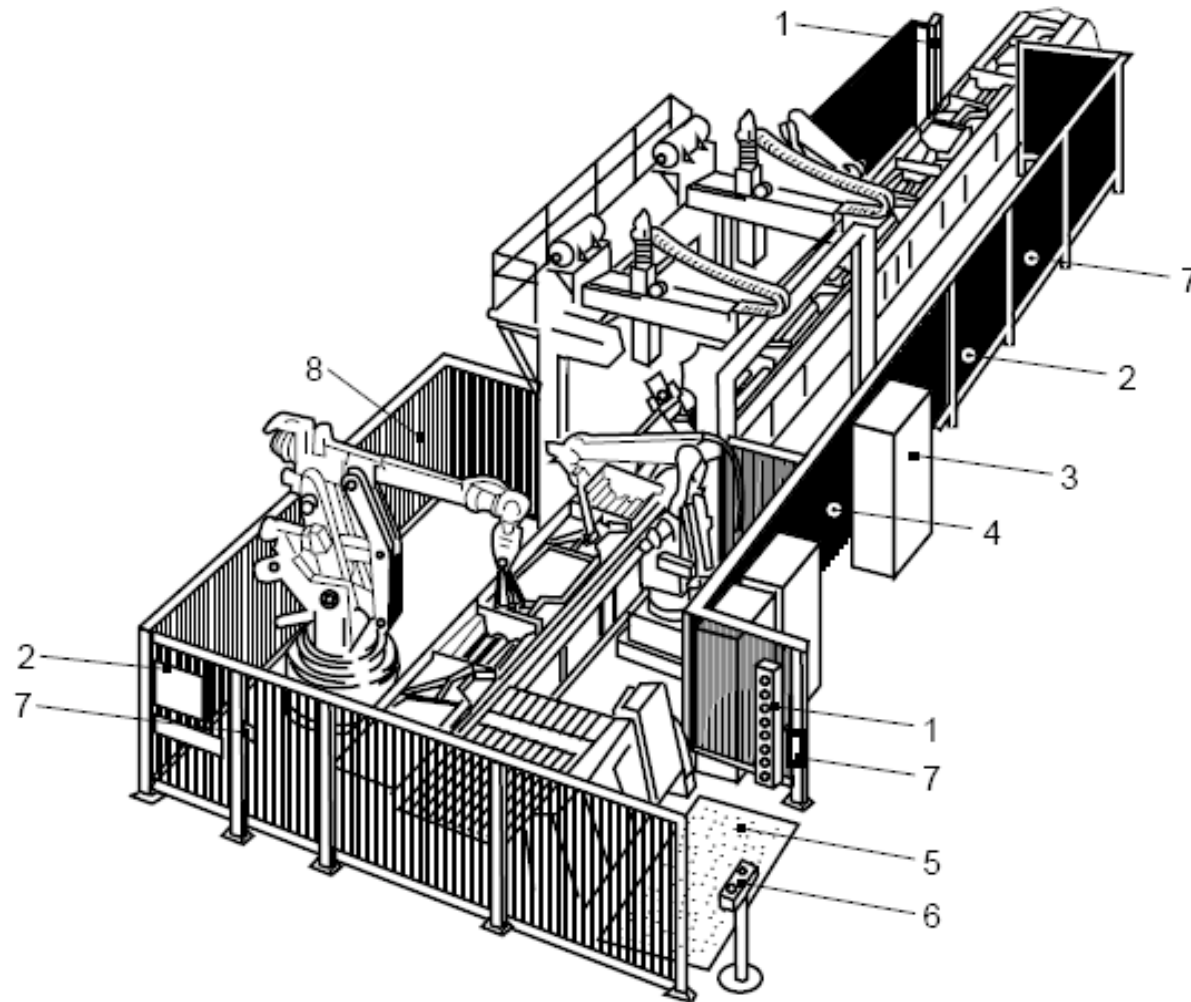


- ▶ ANSI/RIA R15.06-1999 Industrial robots and robot systems – Safety requirements



- ▶ AS 2939-1987 Industrial robot systems – Safe design and usage





LEGEND:

1 = Optoelectronic curtain

2 = Interlocking guard

3 = Electrical cabinet

4 = Internal fence allowing sectional access

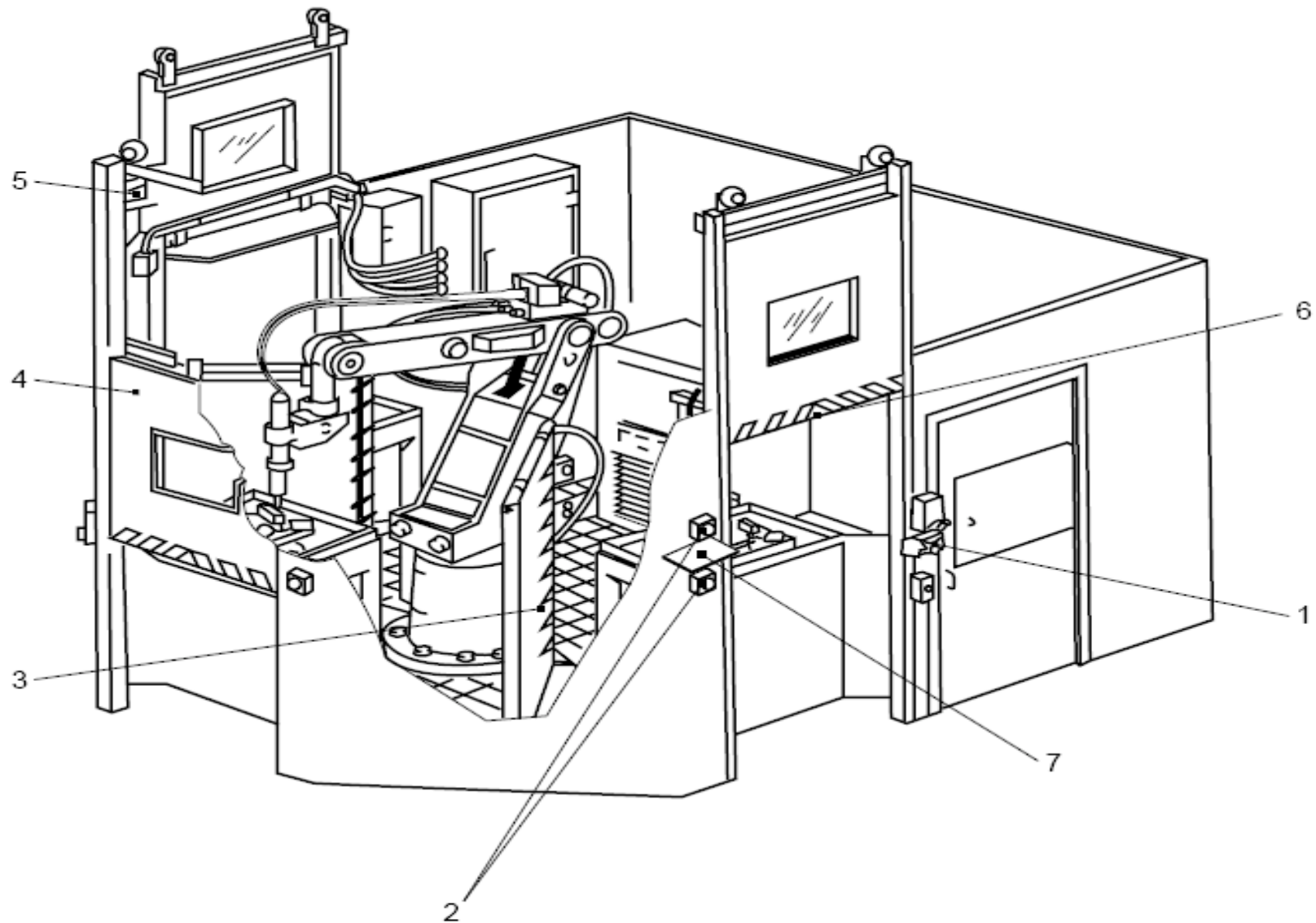
5 = Pressure sensitive mat

6 = Two-hand control device

7 = Reset actuator

8 = Distance guard

FIGURE 9 EXAMPLE 1 OF COMBINATION OF DIFFERENT GUARDS AND GUARDS WITH OTHER PROTECTIVE DEVICES



LEGEND:

- | | |
|-----------------------------|-----------------------------|
| 1 = Trapped key system | 4 = Interlocking guard |
| 2 = Two-hand control device | 5 = Guard locking device |
| 3 = Screen between stations | 6 = Pressure-sensitive edge |

FIGURE 10 EXAMPLE 2 OF COMBINATION OF DIFFERENT GUARDS AND GUARDS WITH OTHER PROTECTIVE DEVICES

Safety of advanced robotics

- ▶ **Generic safety standards**

- ▶ ISO/IEC Guide 51 : Safety aspects — Guidelines for their inclusion in standards
- ▶ IEC 61508 Functional safety of electrical / electronic / programmable electronic safety-related systems

- ▶ **Advanced Robotics**

- ▶ Nothing... except in ISO 10218 (Robots for industrial environments – Safety requirements)
 - ▶ “If a person enters the collaborating workspace the robot performs a safety-rated monitored stop.” p5
 - ▶ => incompatible with advanced robotics (simultaneous movements)



ISO/FDIS 10218-2

- ▶ Definition of collaborative robot: Robot designed for direct contact with a human within a defined collaborative workspace.
 - ▶ The design of the robot system and cell layout is a key process in the elimination of hazards and reduction of risks.
 - ▶ The safety function shall fulfil a least safety category d- or Safety Integrity Level (SIL) 2
 - ▶ Persons/Operators shall be safeguarded by a combination of protective devices and compliance with robot performance features.
 - ▶ section 5.11.5: Collaborating robot operation
 - ▶ Collaboration is only
 - ▶ used for pre-determined tasks,
 - ▶ possible when all required protective measures are active, and
 - ▶ for robots with features specifically designed for collaborating operation complying with ISO 10218-1:2006 Clause 5.10.
 - ▶ Operation in the collaborating workspace – one or more of the following condition need to be fulfilled
 - ▶ If a person enters the collaborating workspace the robot performs a safety-rated monitored stop.
 - ▶ Hand guided mode under the conditions of a defined hand over position, a hand guided device that meets the requirements of ISO 10218-1:2006 and a clear visibility over the entire collaborative workspace.
 - ▶ Speed and position monitoring, for example under the observance of a safe distance.
-
- ▶ Power and force limiting by design or control.



Safety of medical robots

- ▶ **Medical robots as “medical devices”**
 - ▶ Directive 2007/47/EEC amending Council Directive 93/42/EEC concerning medical devices
 - ▶ ISO 14971 - Risk management for medical devices
- ▶ **Nothing specific for medical robots**



Safety of rehabilitation robots

- ▶ **Generic**
 - ▶ Machinery standards ?
 - ▶ Robotics standards ?
- ▶ **Domain specific standards (not robotic specific)**
 - ▶ Mobility aid
 - ▶ Manipulation aid
 - ▶ Therapeutics



Industrial robots Vs advanced Robots

Industrial robotics	Advanced robotics		New hazards Examples
No movement if human presence	Simultaneous movements	➔	Bad synchronization / Non human legible movements
Human is “far”	Human is close / Physical contact	➔	Impacts / forces too high
Teach pendant	Advanced interaction (cognitive)	➔	Mode confusion / communication errors / understanding errors
Automatic	Autonomous	➔	Hazardous decisions / Bad decisions
Heavy/Stiff/Powerful	Light/Compliant/Limited power (“intrinsically safe”)	➔	Precision hazards
Mono function	Multi function	➔	Tasks too complex
Structured environment	Unstructured environment	➔	Hazardous situations (Robustness issues)

Conclusion

- ▶ Rehabilitation robotics = advanced robotics
- ▶ New functionalities + new hazards => no specific safety standards (application of some machinery or industrial robotics concepts but impossibility to reach a complete conformity)
- ▶ Apply methods and tools from other critical systems

How **dependability** techniques could be used in a **risk management** process for advanced robotics ?



Chapter 2. Dependability concepts

Credits : Most of the slides have been generously given by Jean-Claude Laprie
(Resist courseware <http://resist.isti.cnr.it/>)

Dependability: ability to deliver service that can justifiably be trusted

Service delivered by a system: its behavior as it is perceived by its user(s)

User: another system that interacts with the former

Function of a system: what the system is intended to do

(Functional) Specification: description of the system function

Correct service: when the delivered service implements the system function

(Service) Failure: event that occurs when the delivered service deviates from correct service, either because the system does not comply with the specification, or because the specification did not adequately describe its function

Failure modes: the ways in which a system can fail, ranked according to failure severities

Part of system state that may cause a subsequent service failure: **error**

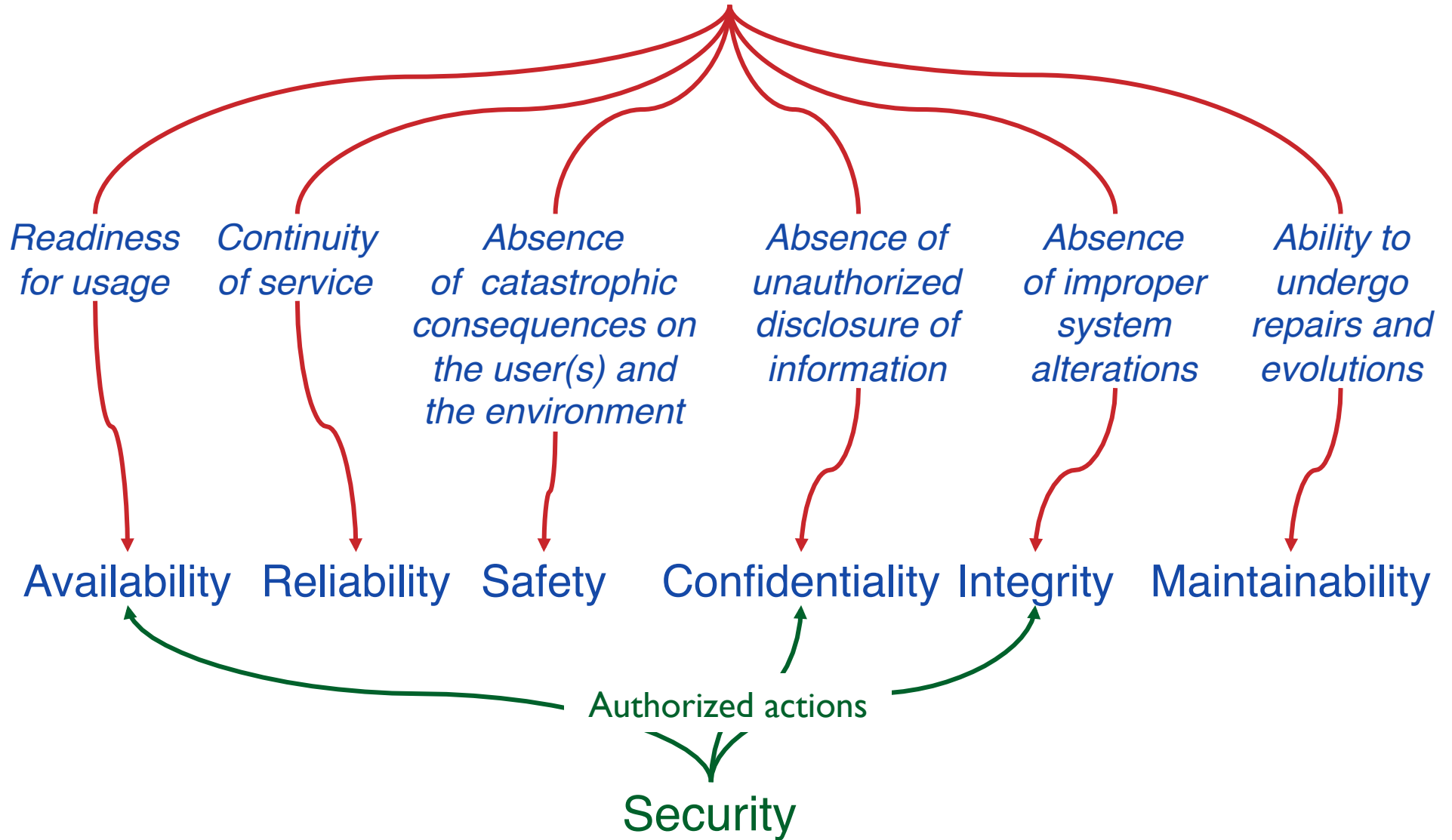
Adjudged or hypothesized cause of an error: **fault**

Dependability: ability to avoid failures that are unacceptably frequent or severe

Failures unacceptably frequent or severe: **dependability failure**

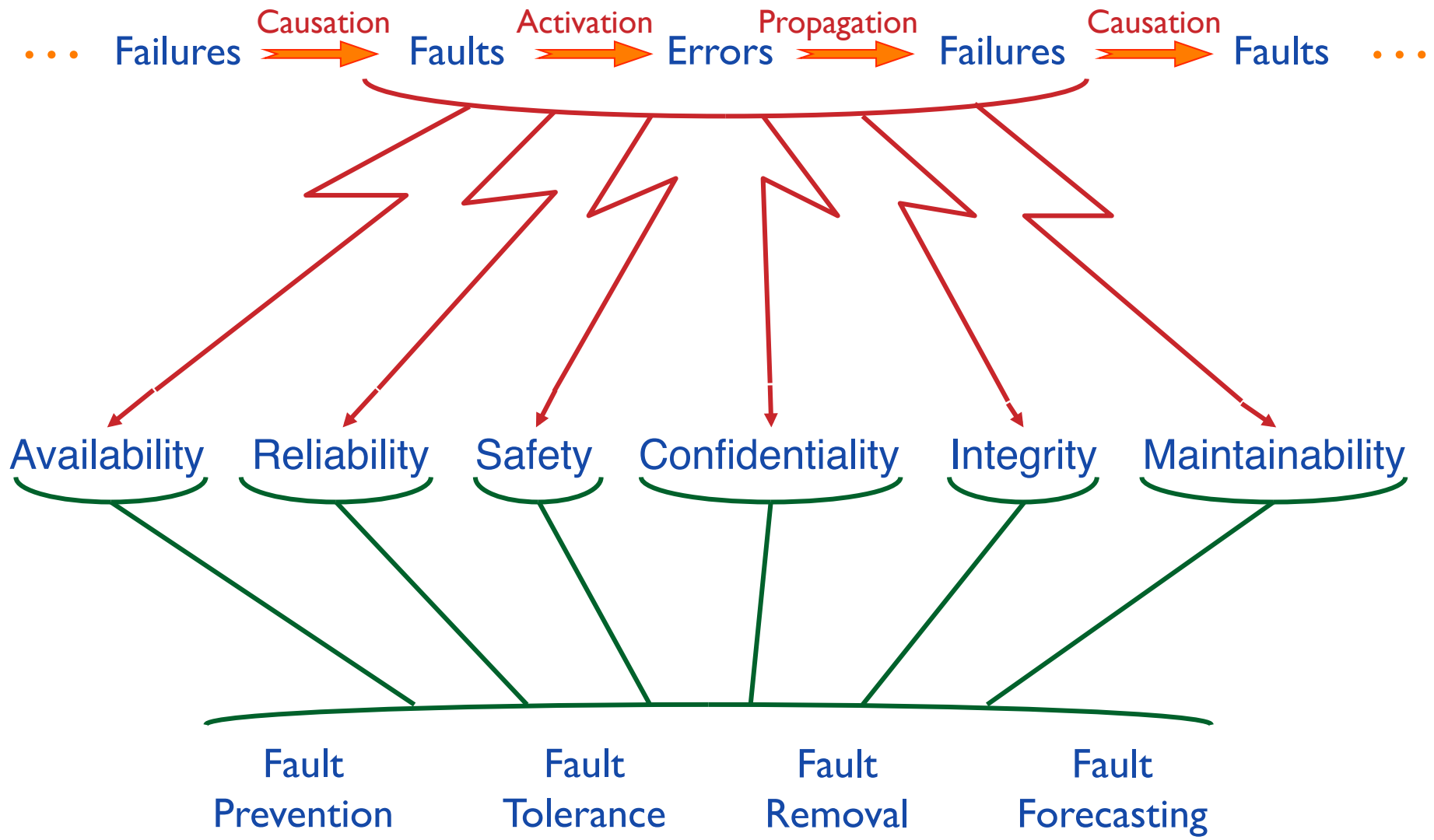


Dependability



Absence of unauthorized access to, or handling of, system state



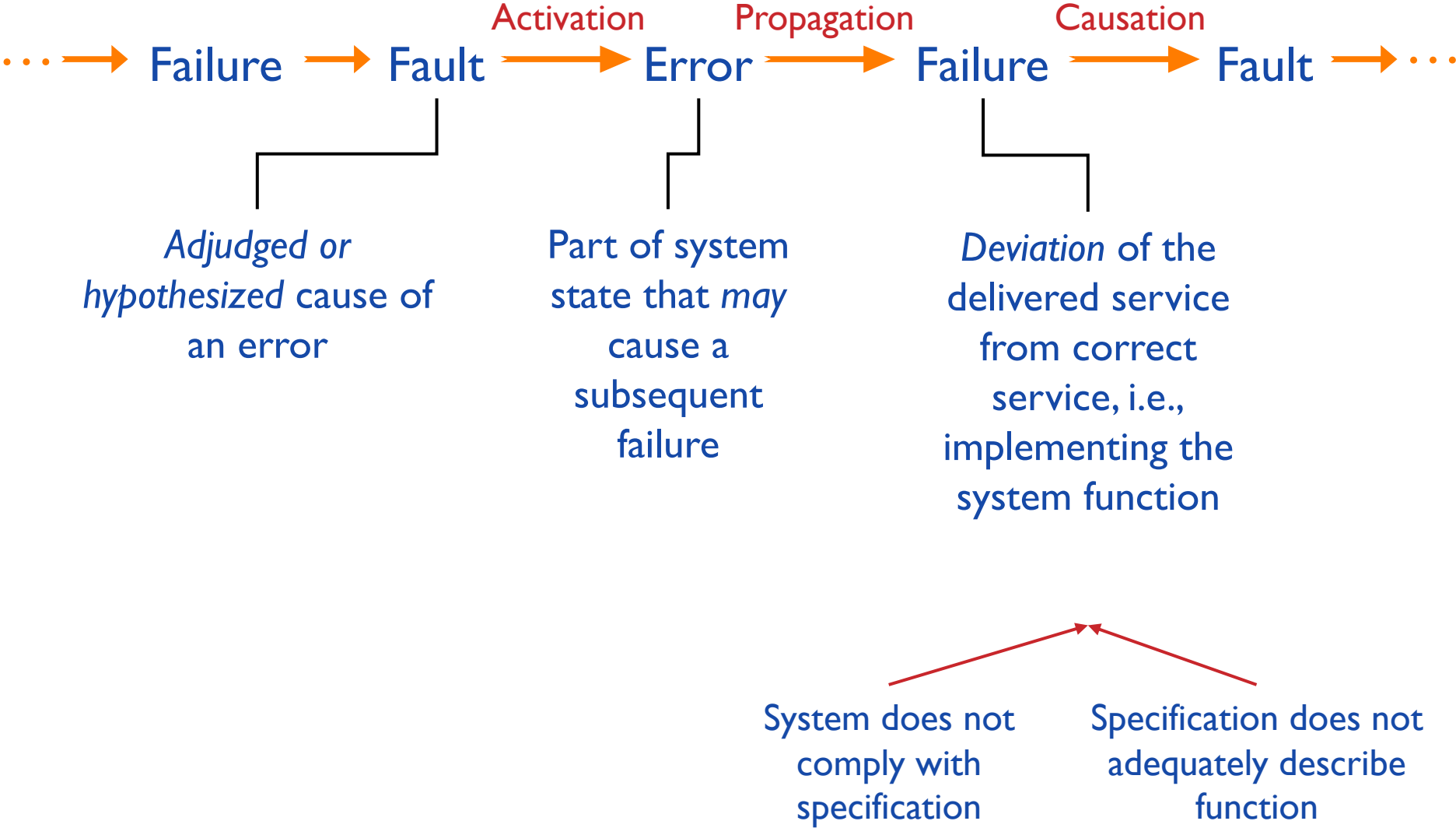


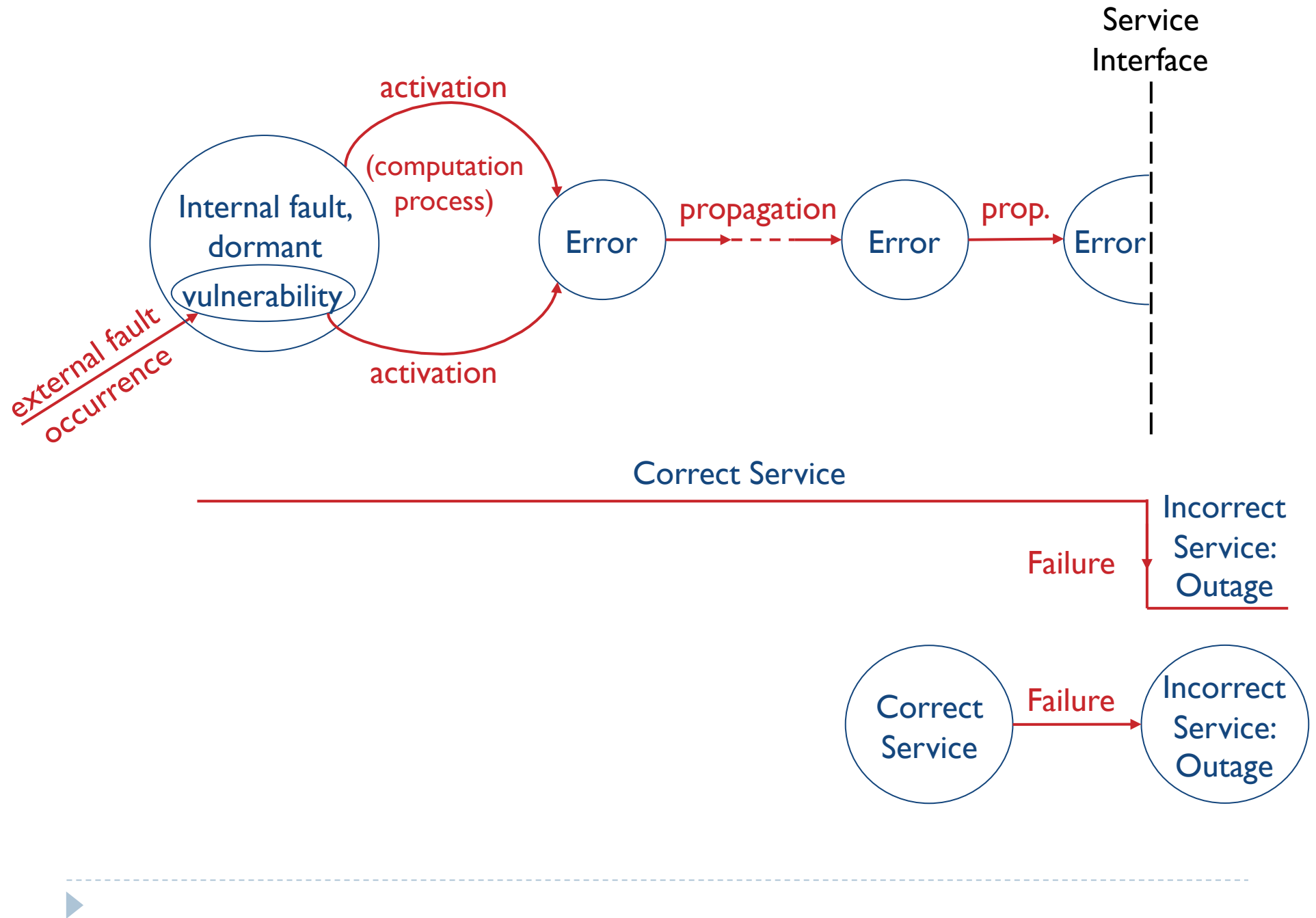
Dependability attributes

- ❖ Availability, Reliability, Safety, Confidentiality, Integrity, Maintainability: **Primary attributes**
- ❖ **Secondary attributes**
 - Specialization
 - ✓ Robustness: dependability with respect to external faults
 - ✓ Survivability: dependability in the presence of active fault(s)
 - ✓ Resilience: dependability when facing functional, environmental, or technological changes
 - Distinguishing among various types of (meta-)information
 - ✓ Accountability: availability and integrity of the person who performed an operation
 - ✓ Authenticity: integrity of a message content and origin, and possibly some other information, such as the time of emission
 - ✓ Non-repudiability: availability and integrity of the identity of the sender of a message (non-repudiation of the origin), or of the receiver (non-repudiation of reception)

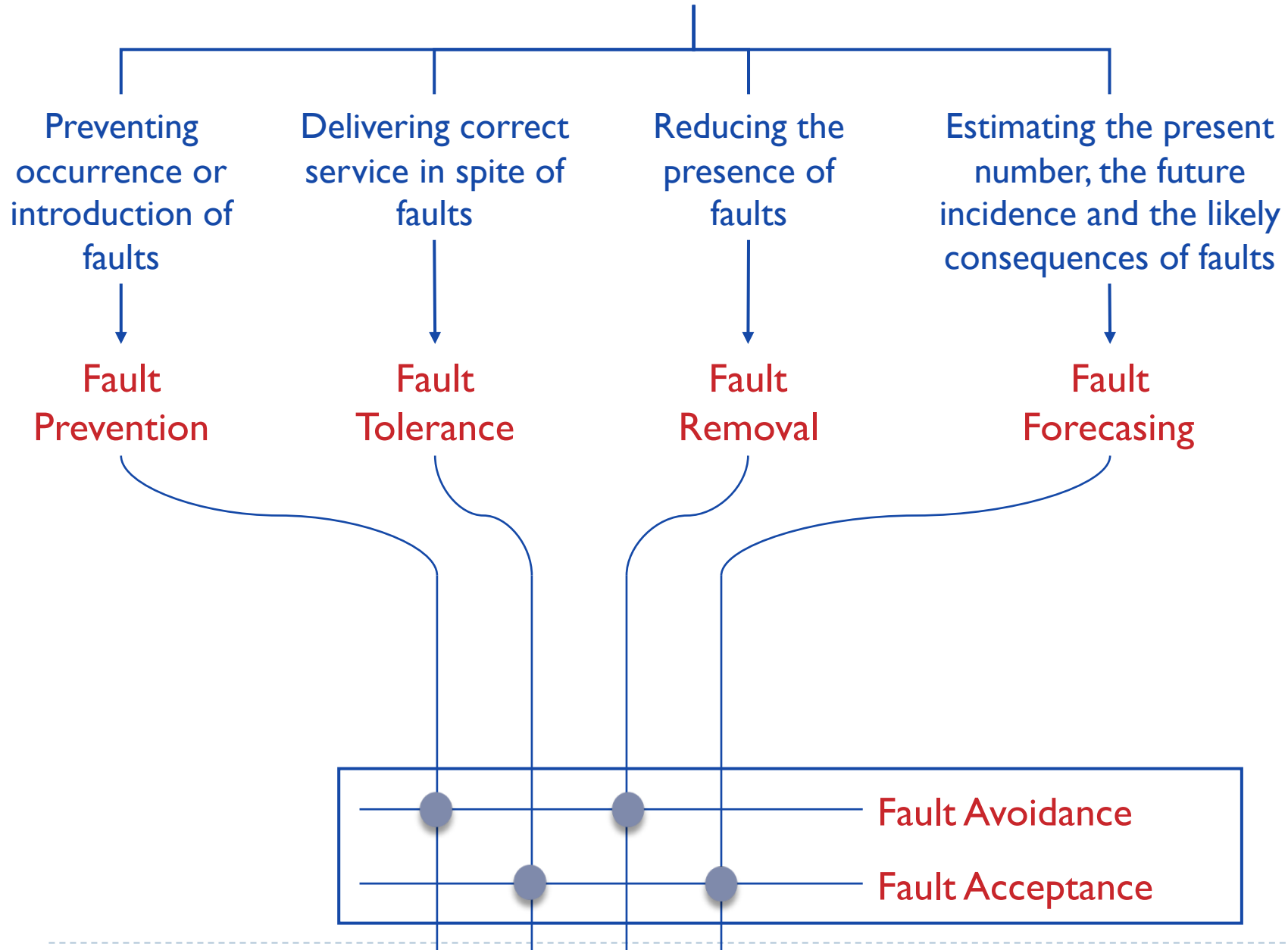


Dependability Threats





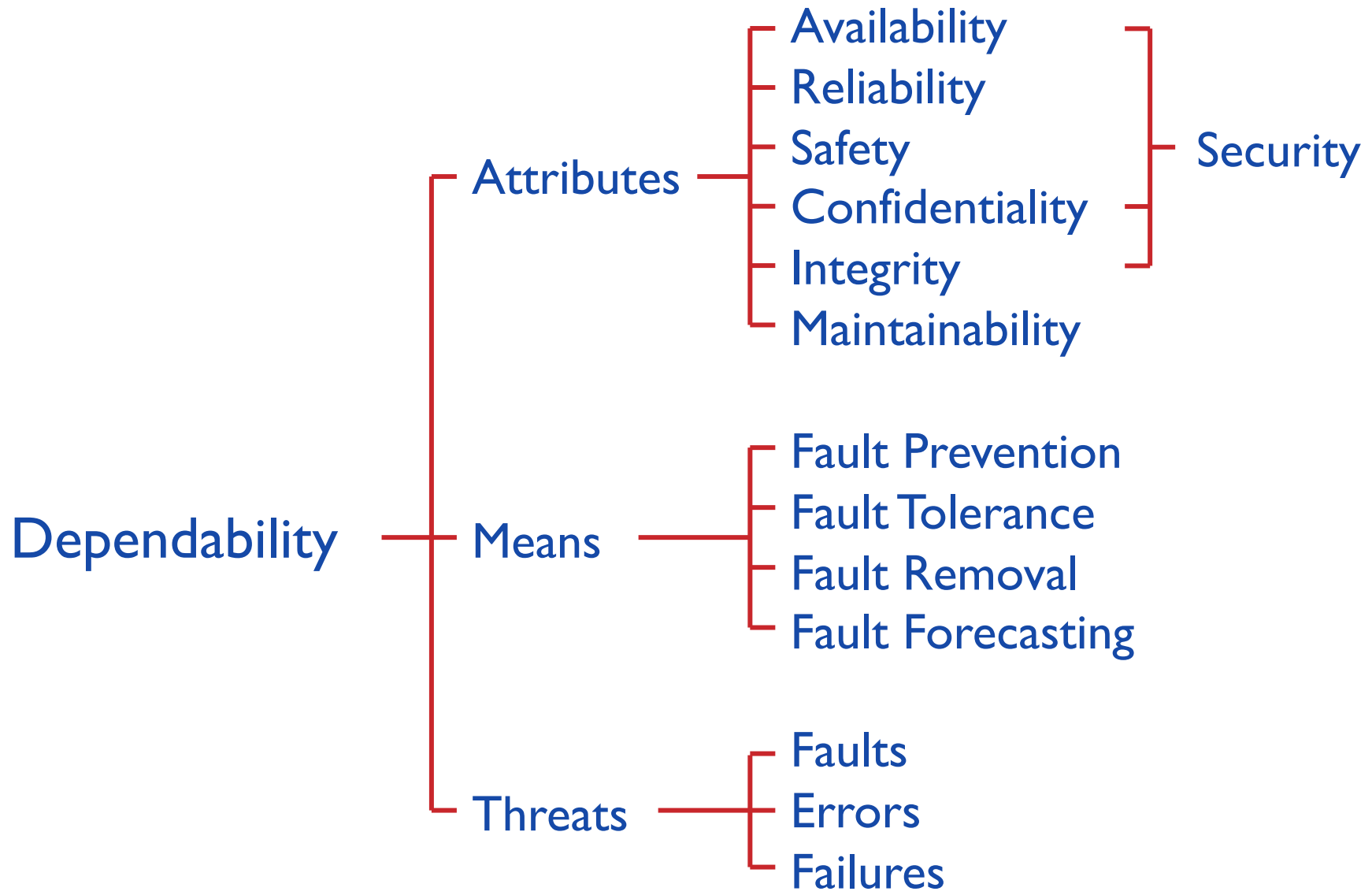
Means for Dependability

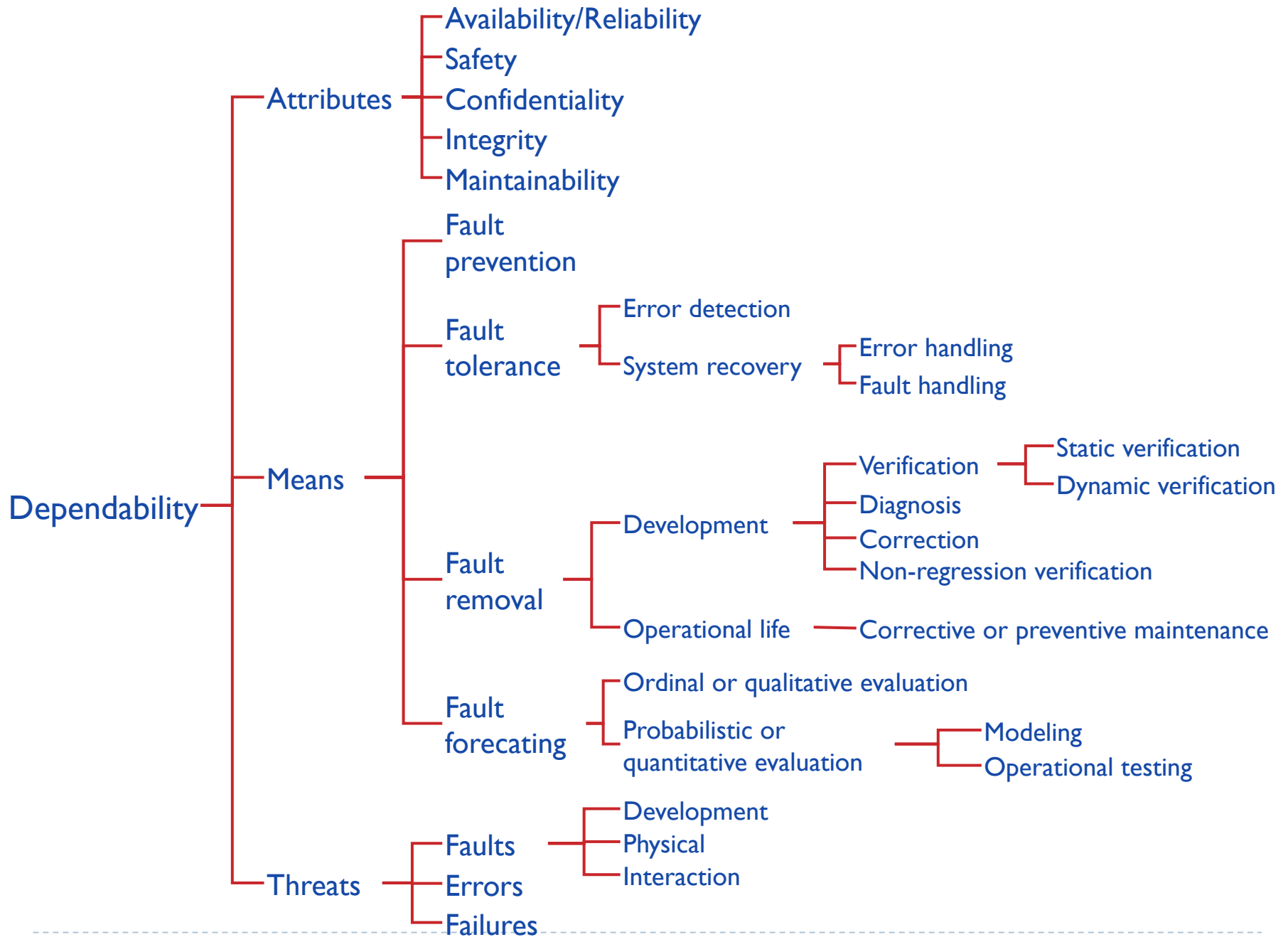


Dependability and similar notions

Concept	Dependability	High Confidence	Survivability	Trustworthiness
Goal	<ol style="list-style-type: none"> 1) ability to deliver service that can justifiably be trusted 2) ability of a system to avoid service failures that are unacceptably frequent or severe 	<p>consequences of the system behavior are well understood and predictable</p>	<p>capability of a system to fulfill its mission in a timely manner</p>	<p>assurance that a system will perform as expected</p>
Threats present	<ol style="list-style-type: none"> 1) development faults (e.g., software flaws, hardware errata, malicious logic) 2) physical faults (e.g., production defects, physical deterioration) 3) interaction faults (e.g., physical interference, input mistakes, attacks, including viruses, worms, intrusions) 	<ul style="list-style-type: none"> • internal and external threats • naturally occurring hazards and malicious attacks from a sophisticated and well-funded adversary 	<ol style="list-style-type: none"> 1) attacks (e.g., intrusions, probes, denials of service) 2) failures (internally generated events due to, e.g., software design errors, hardware degradation, human errors, corrupted data) 3) accidents (externally generated events such as natural disasters) 	<ol style="list-style-type: none"> 1) hostile attacks (from hackers or insiders) 2) environmental disruptions (accidental disruptions, either man-made or natural) 3) human and operator errors (e.g., software flaws, mistakes by human operators)
References		<p>'Information technology frontiers for a new millenium', Blue Book 2000, NTSC</p>	<p>A. Ellison et al., 'Survivable network systems', SEI Report, 1999</p>	<p>F. Schneider, ed., 'Trust in cyberspace', National Academy Press, 1999</p>







Dependability threats:
faults, errors, failures

Error of a programmer



Fault

Impaired instructions or data



Activation

Faulty component **and** inputs



Error



Propagation

When delivered service deviates (value, timing) from implementing function



Failure



Short-circuit in integrated circuit

Failure



Causation



Fault

Stuck-at connection, modification of circuit function



Activation

Faulty component **and** inputs



Error



Propagation

When delivered service deviates (value, timing) from implementing function



Failure



Operator **Error**

Inappropriate human-system interaction

Fault



Error



Propagation

When delivered service deviates (value, timing) from
implementing function



Failure



Electromagnetic perturbation

Fault

Fault

Impaired memory data

Activation

Faulty component **and** inputs

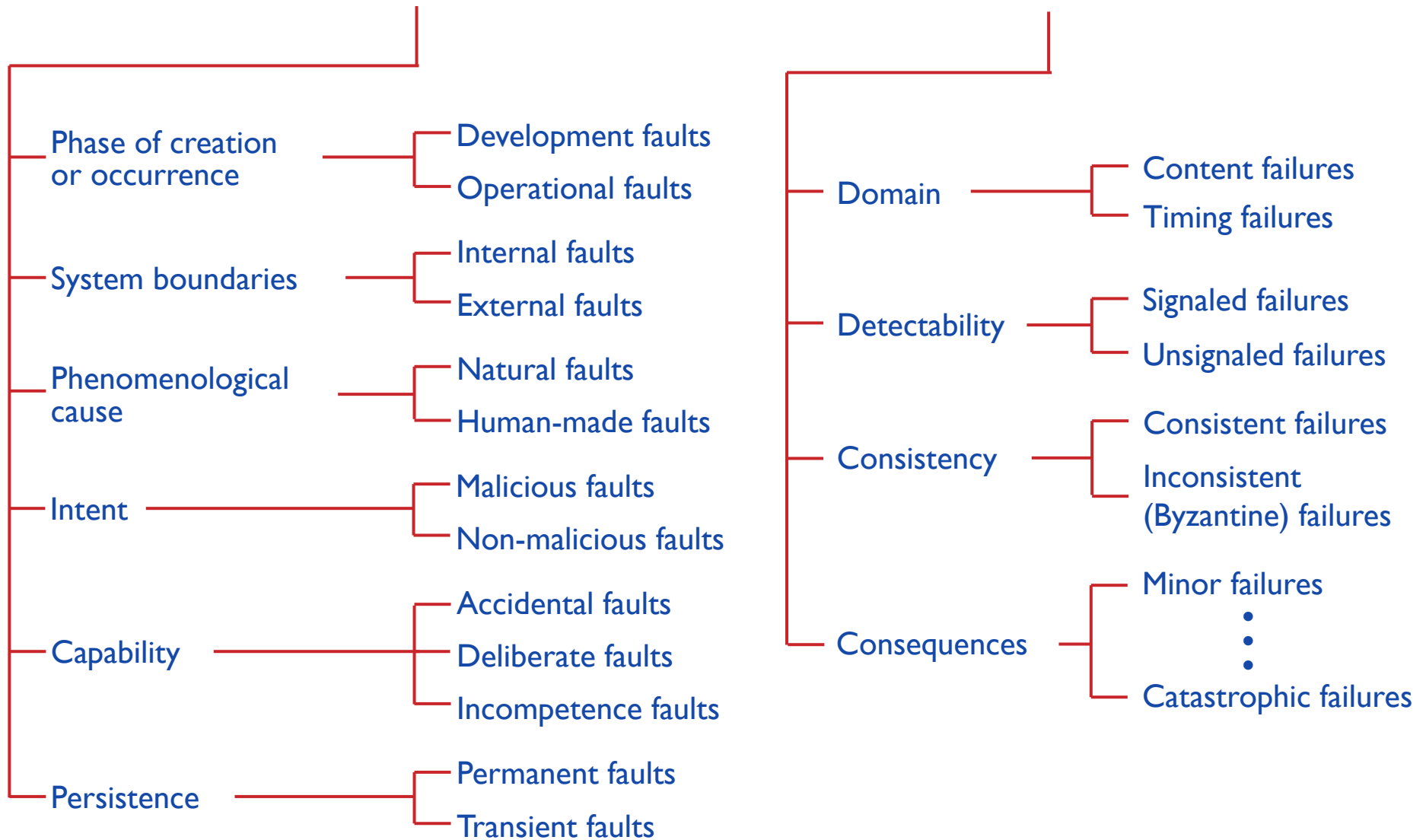
Error

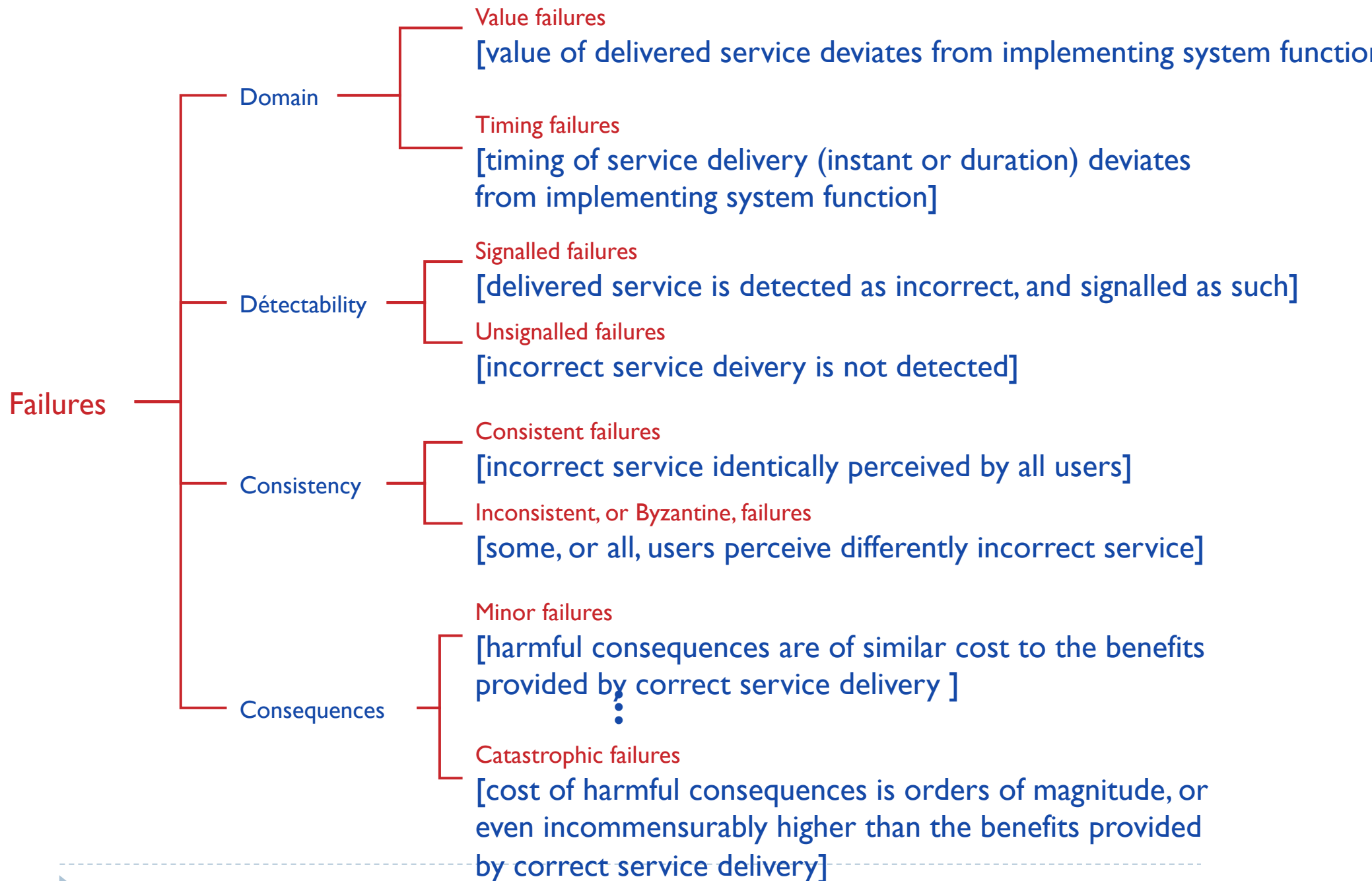
Propagation

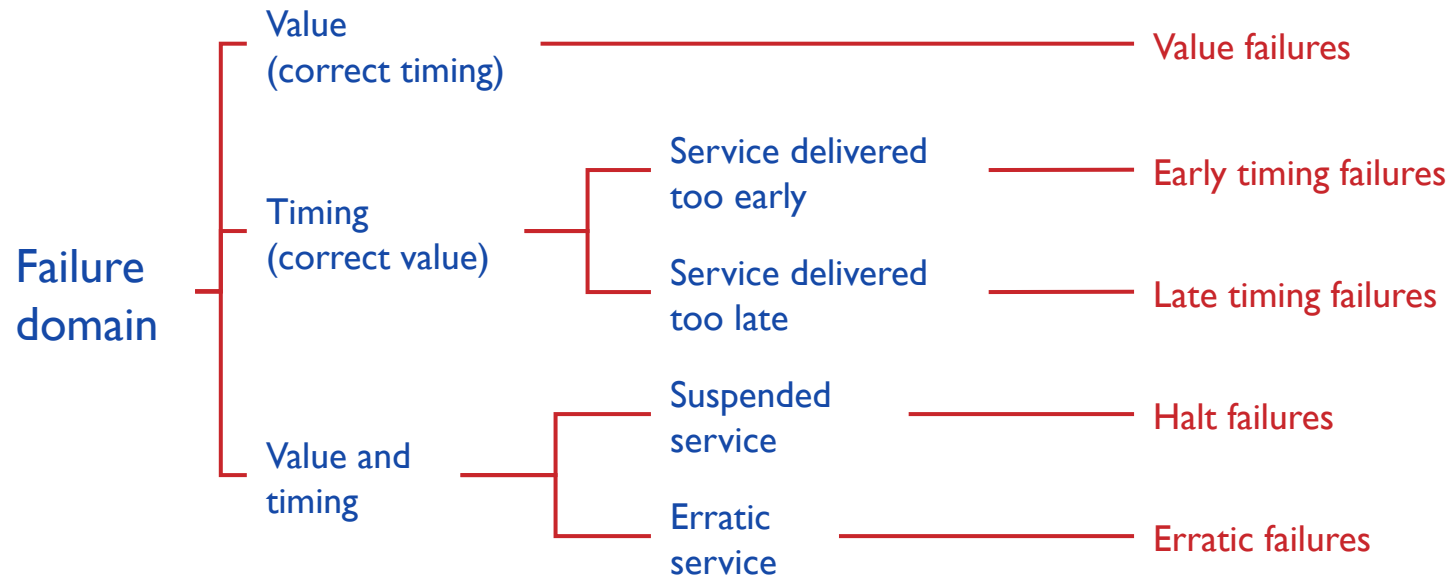
When delivered service deviates (value, timing) from implementing function

Failure









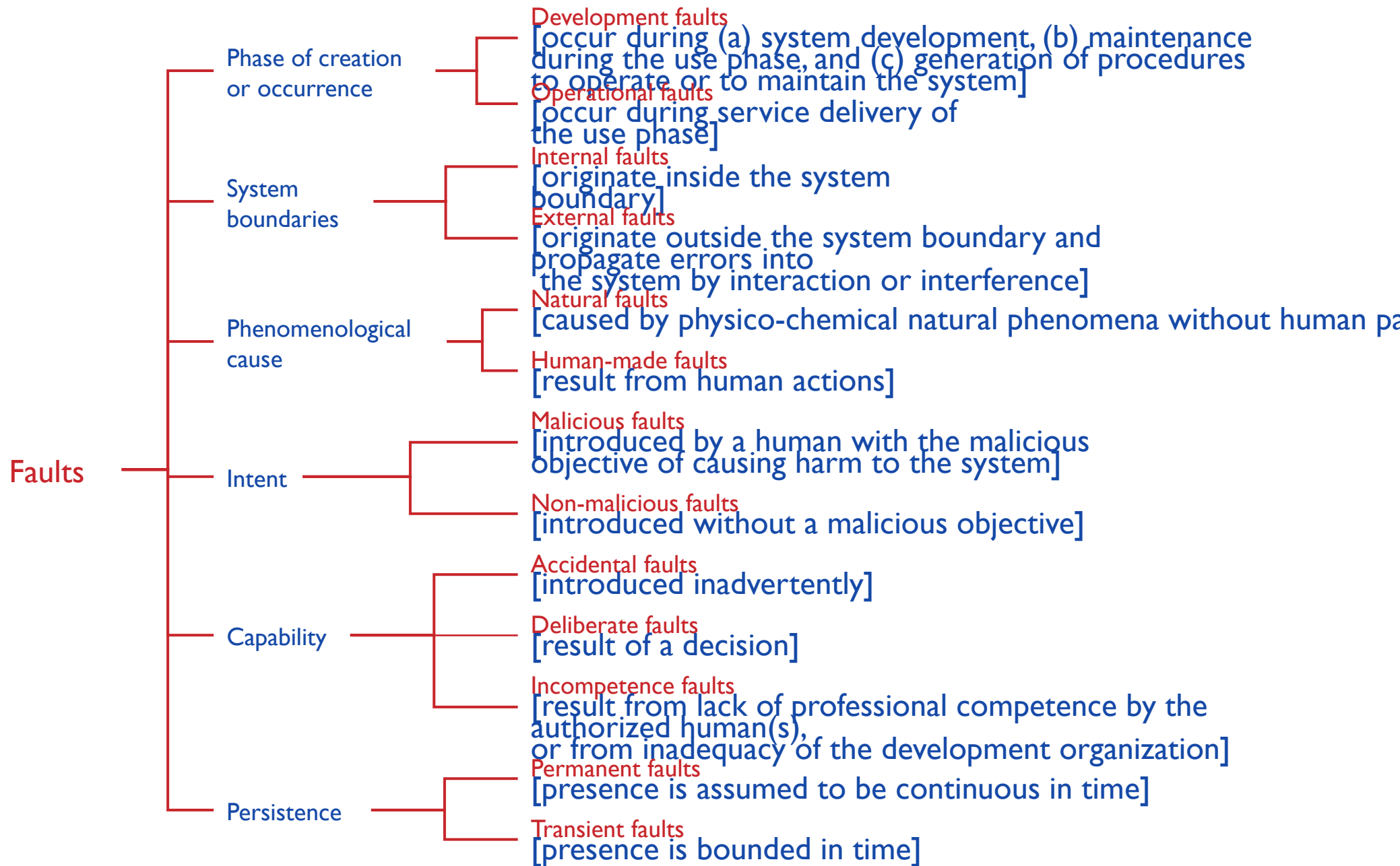
Failure of detecting mechanisms

- Non signalling of incorrect service: **unsignalled failure**
- Signalling incorrect service in absence of failure: **false alarm**

System whose all failures are, to an acceptable extent

- halt failures: **fail-halt system**
- minor failures: **fail-safe system**





Faults

Phase of creation or occurrence

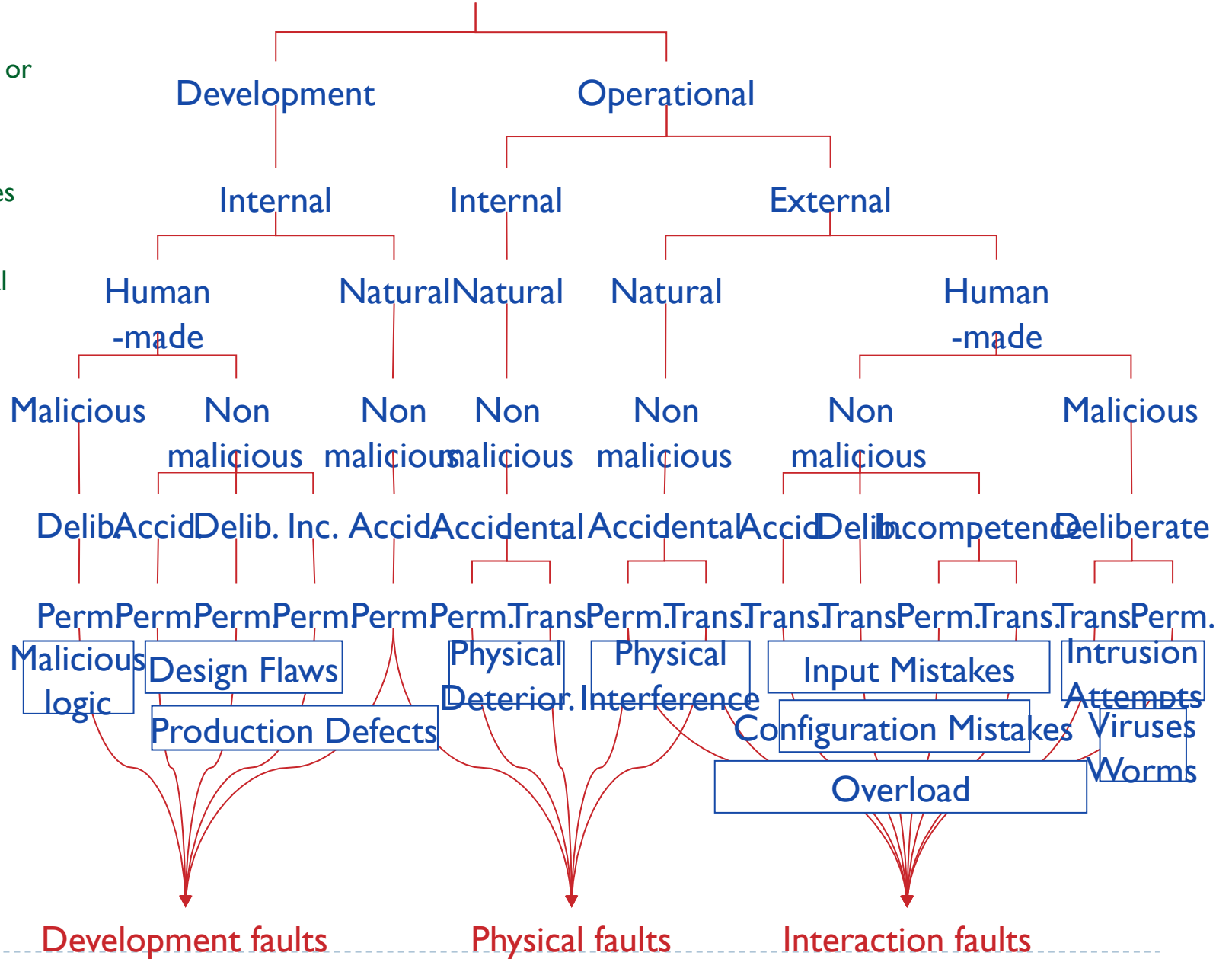
System boundaries

Phenomenological cause

Intent

Capability

Persistence



Physical faults Development faults Interaction faults

Hardware Software System

Human-made Faults

Intent

Non-malicious

Malicious

Capability

Accidental
(Mistakes)

Deliberate
(Bad decisions)

Incompetence

Deliberate

Interaction
(operators,
maintainers)
&
Development
(designers)

Individuals
&
organizations

Malicious logic
faults:
logic bombs,
Trojan horses,
trapdoors,
viruses, worms,
zombies

Intrusion
attempts

State of the art from statistics

Fault examples

	Faults			Failures		Availability/ Reliability	Safety	Confidentiality
	Physical	Development	Interaction	Localized	Distributed			
June 1980: False alerts at the North American Air Defense (NORAD)	✓			✓		✓		
June 1985 - January 1987: Excessive radiotherapy doses (Therac-25)		✓		✓			✓	
August 1986 - 1987: the "wily hacker" penetrates several tens of sensitive computing facilities		✓	✓	✓				✓
November 1988: Internet worm		✓	✓		✓	✓		
15 January 1990: 9 hours outage of the long-distance phone in the USA		✓			✓	✓		
February 1991: Scud missed by a Patriot (Dhahran, Gulf War)		✓	✓	✓		✓	✓	
November 1992: Crash of the communication system of the London ambulance service		✓	✓		✓	✓	✓	
26 and 27 June 1993: Authorization denial of credit card operations in France	✓	✓			✓	✓		
4 June 1996: Failure of Ariane 5 maiden flight		✓		✓			✓	
13 April 1998: Crash of the AT&T data network		✓	✓		✓	✓		
February 2000: Distributed denials of service on large Web sites		✓	✓		✓	✓		
May 2000: Virus <i>I love you</i>		✓	✓		✓	✓		
July 2001: Worm <i>Code Red</i>		✓	✓		✓	✓		
August 2003: Propagation of the electricity blackout in the USA and Canada		✓	✓		✓	✓		
October 2006: 83,000 e-mail addresses, credit card info, banking transaction files stolen in UK		✓	✓		✓			✓



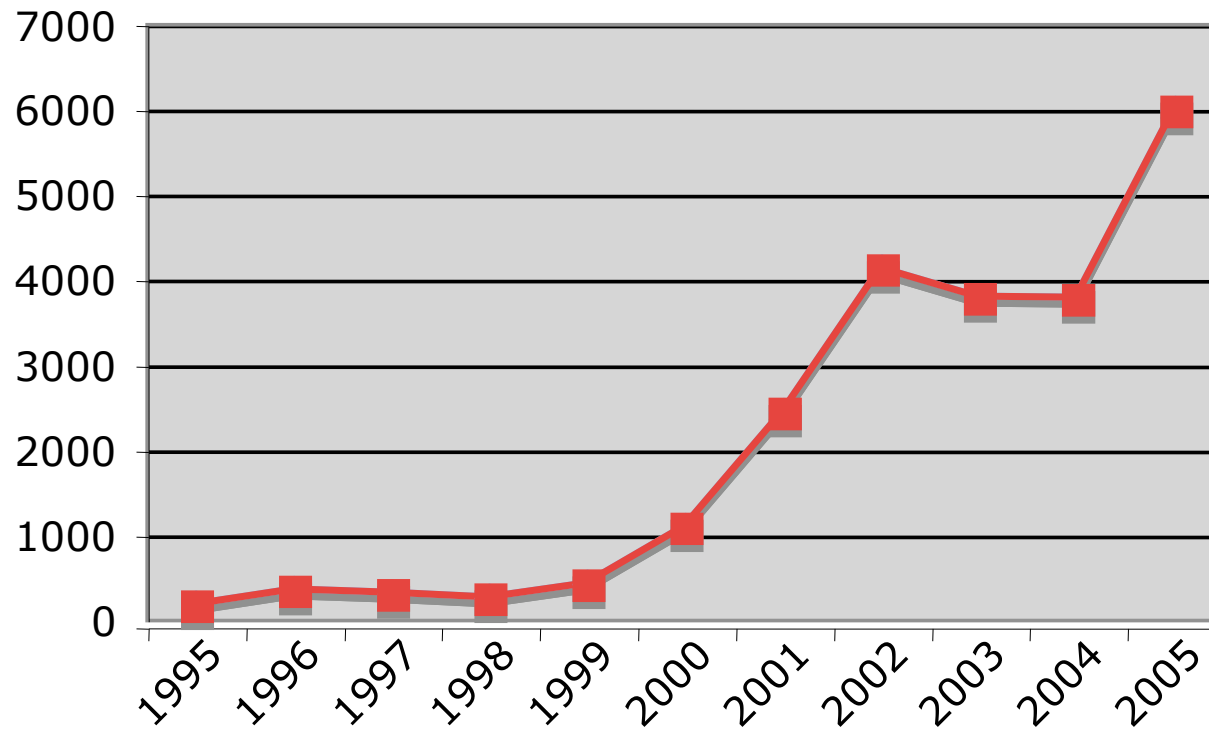
Accidental faults

Number of failures [consequences and outage durations depend upon application]	Dedicated computing systems (e.g., transaction processing, electronic switching, Internet backend servers)		Controlled systems (e.g., civil airplanes, phone network, Internet frontend servers)	
Faults	Rank	Proportion	Rank	Proportion
Physical internal	3	~ 10%	2	15-20%
Physical external	3	~ 10%	2	15-20%
Human interactions	2	~ 20%	1	40-50%
Development	1	~ 60%	2	15-20%



Malicious faults

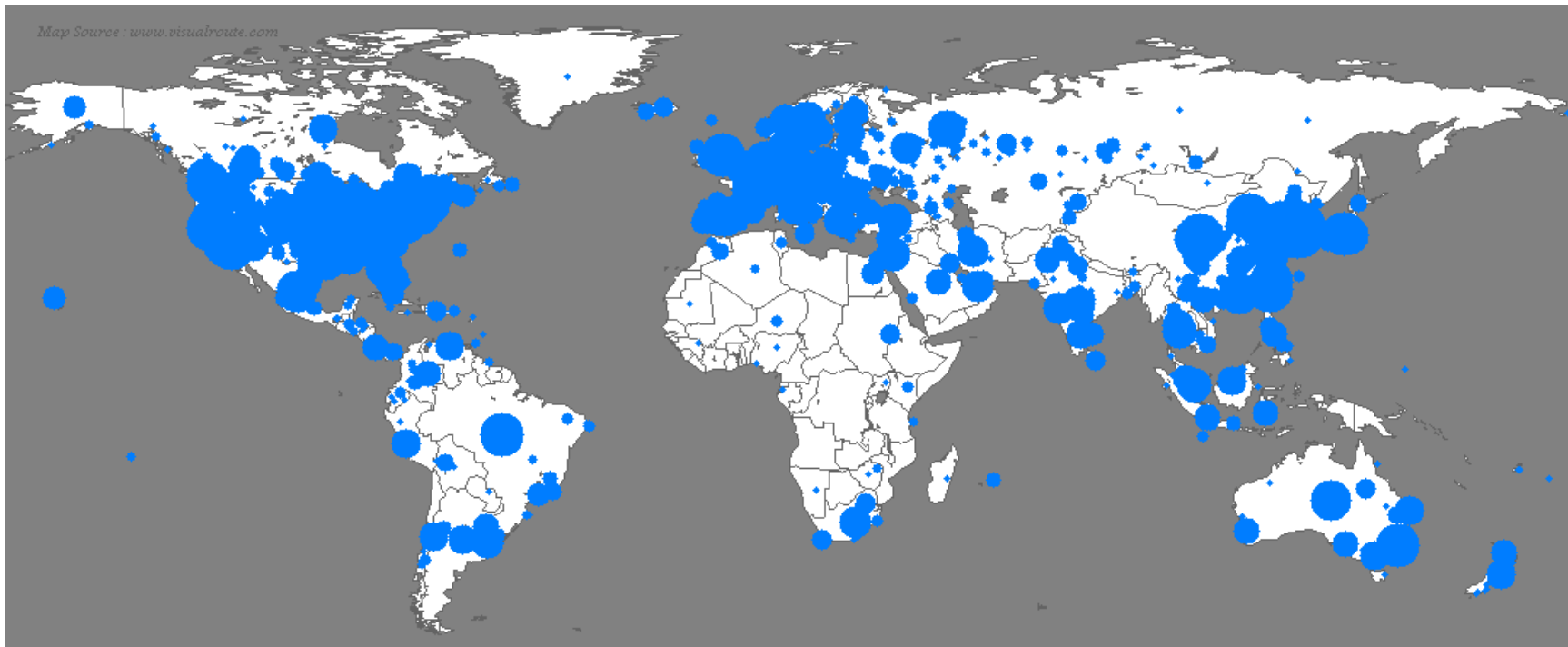
SEI/CERT Statistics: vulnerabilities reported



Slammer/Sapphire worm

[From: <http://www.caida.org/publications/papers/2003/sapphire/sapphire.html>]

The fastest computer worm in history. As it began spreading throughout the Internet, it doubled in size every 8.5 seconds. It infected more than 90 percent of vulnerable hosts within 10 minutes. The worm began to infect hosts slightly before 05:30 UTC on Saturday, January 25, 2003. Sapphire exploited a buffer overflow vulnerability in computers on the Internet running Microsoft's SQL Server or MSDE 2000 (Microsoft SQL Server Desktop Engine). This weakness in an underlying indexing service was discovered in July 2002; Microsoft released a patch for the vulnerability before it was announced. The worm infected at least 75,000 hosts, perhaps considerably more, and caused network outages and such unforeseen consequences as canceled airline flights, interference with elections, and ATM failures.

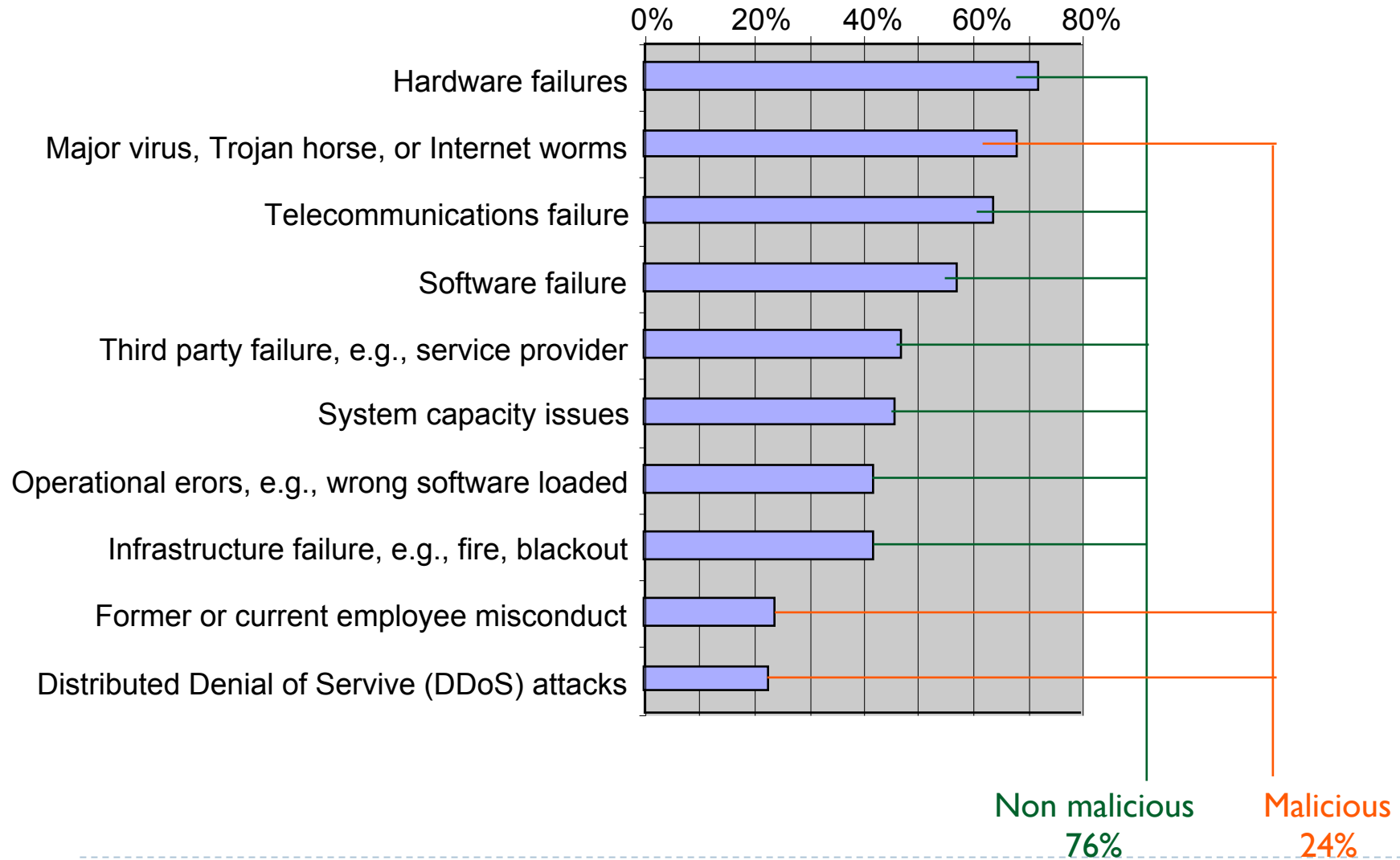


The geographic spread of Sapphire in the 30 minutes after release.

Global Information Security Survey 2004 — Ernst & Young

Loss of availability: Top ten incidents

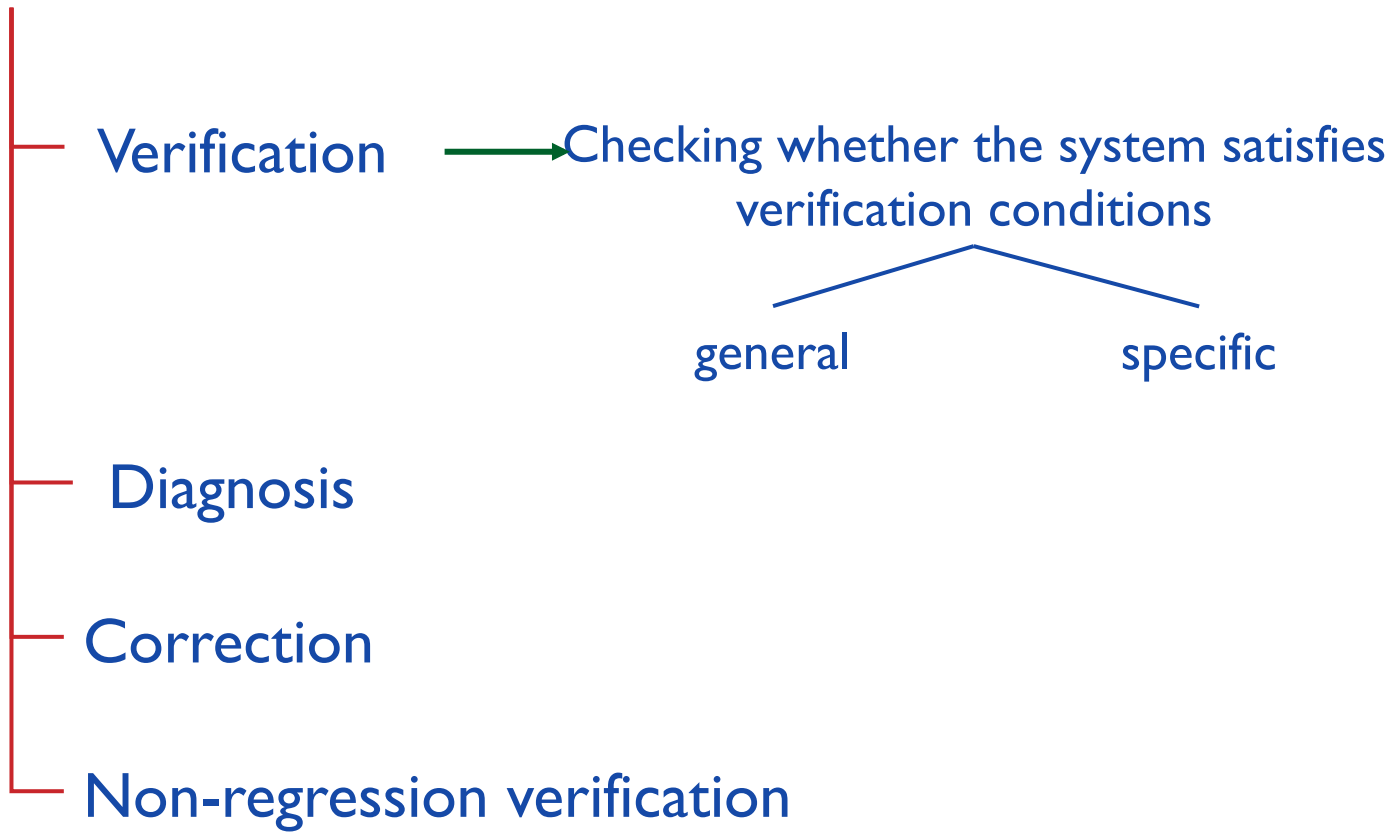
Percentage of respondents that indicated the following incidents resulted in an unexpected or unscheduled outage of their critical business



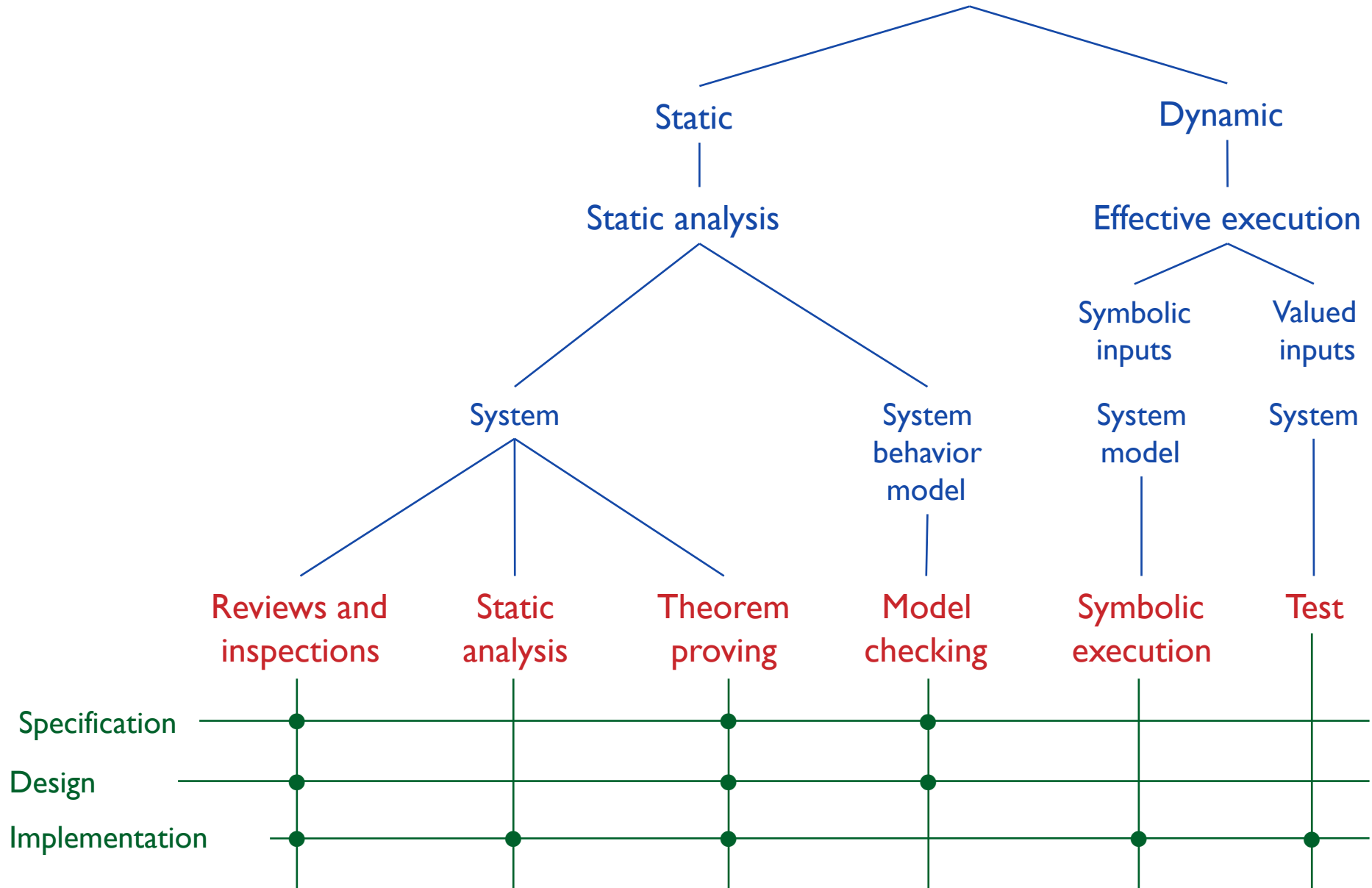


Fault Removal

Reducing the presence of faults



Verification



Reviews and manual inspection

- Specification review
 - User requirements, consistence, feasibility
 - Use of checklists
- Design review
 - Algorithms, interface modules
 - Cheklists
 - modelling, simulation
- Coding review : critical code reading
 - Inspection (e.g. : several developers read the code with “classical faults” lists
 - Audit : verification of programming best practices (control structure, comments, variables names, etc.)
 - Many other approaches

Software reviews and automatic inspections



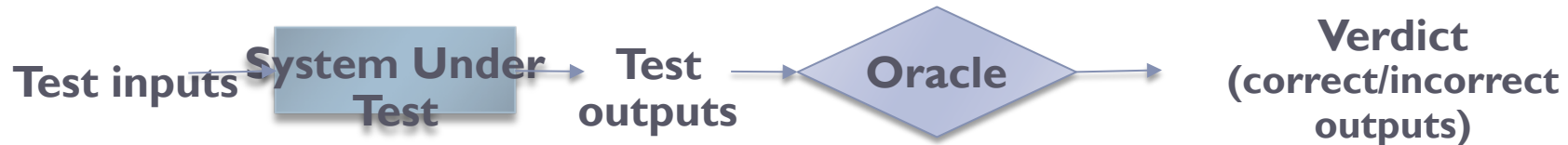
▶ Program Characteristics

- ▶ Instruction lines amount / quantity of types and variables
- ▶ Crossing references
- ▶ Complexity metrics
- ▶ etc.

▶ Detected faults

- ▶ variables : unused, missing initialisation
- ▶ loops : infinite, imbricated
- ▶ Dead instructions
- ▶ Global variables
- ▶ ... see next slide

Test



- ▶ Oracle issue : observe outputs and decide if they are consistent with verification conditions
 - ▶ Manual estimation of expected outputs
- ▶ To identify a fault, it is required that
 - ▶ The fault is activated with a test input
 - ▶ Error is propagating to affect an observable output
 - ▶ A verification condition is violated

Exhaustive test is impossible

- ▶ **Input domain is huge, even infinite**
 - ▶ Conditions of execution for a robot is infinite
 - ▶ Users actions
 - ▶ Environment

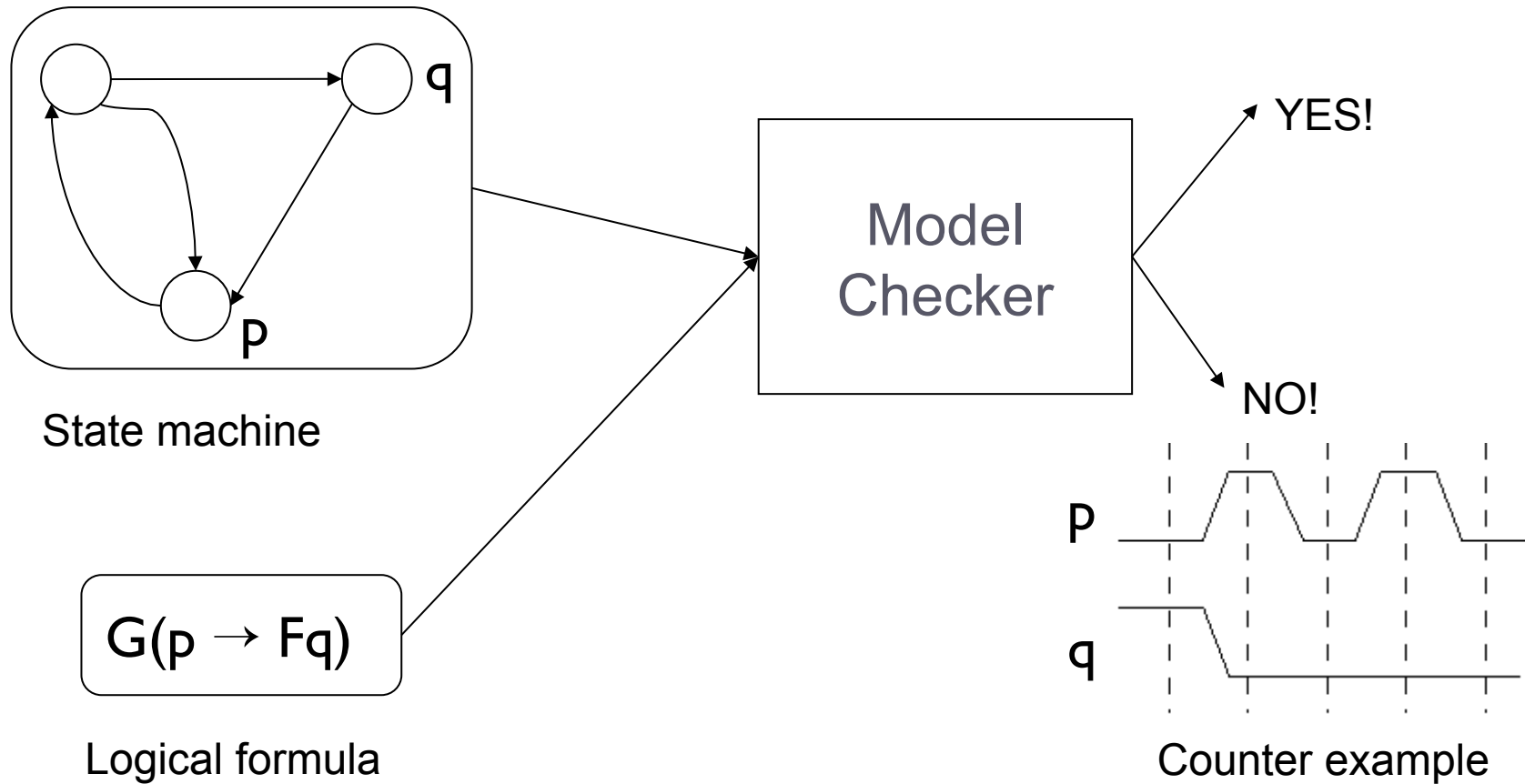
➔ **Partial Verification => test plan / test case should be justified**

- ▶ Main functionalities
- ▶ Critical scenarios

Formal method (proof and model checking)

- ▶ **Formal specification : use of notations to describe**
 - ▶ Hypothesis on the environments
 - ▶ Requirements
 - ▶ A design to satisfy requirements
- ▶ **Formal verification : use of formal methods for**
 - ▶ Analyse specifications considering consistence and completeness criteria
 - ▶ Test specifications
 - ▶ Prove that design satisfy requirements

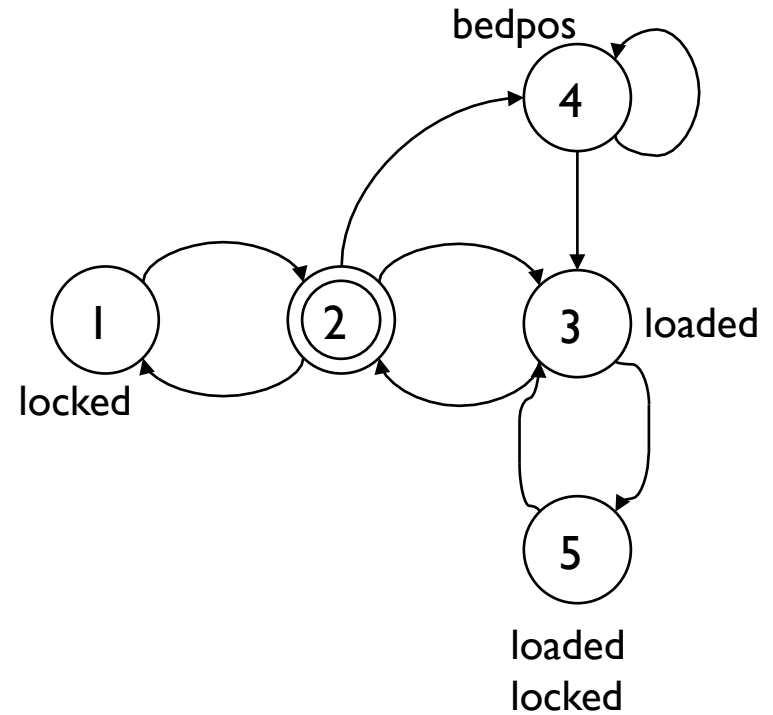
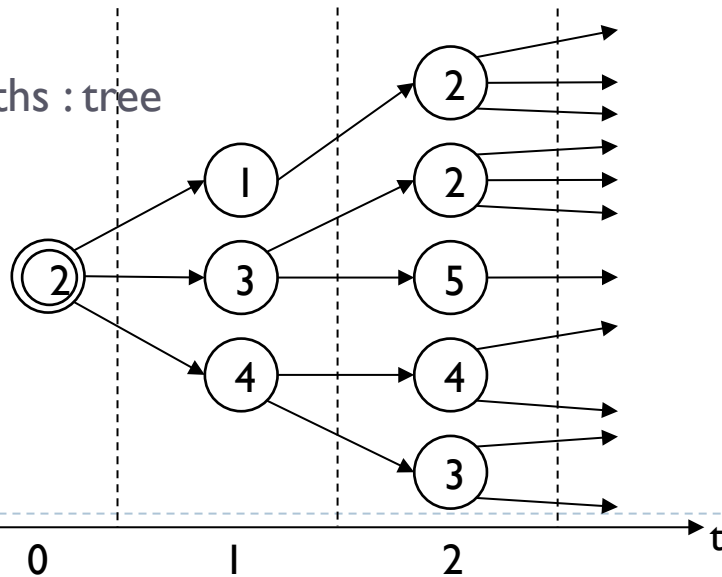
Example : Model checking



Automata

Multiplet	$M = \langle S, S_0, T, P, L \rangle$
S	Finite set of states
S_0	Set of initial state
P	Finite set of propositions
$T \subseteq S \times S$	Transition set
$L(p) : P \mapsto 2^S$	Assign to each proposition the set of states where p is verified

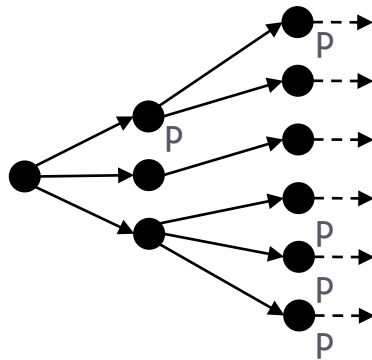
Execution paths : tree



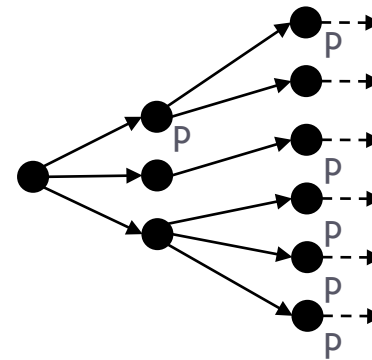
Temporal Logic: CTL (Computation Tree Logic)

Properties on execution paths

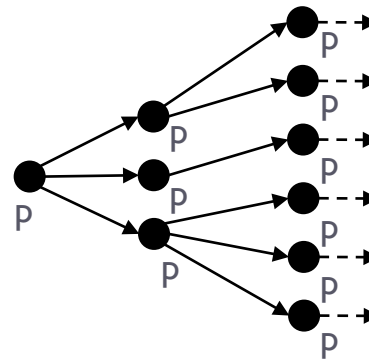
It finally exists a path where p is verified: **EF** p



Property p is finally verified for all paths :**AF** p

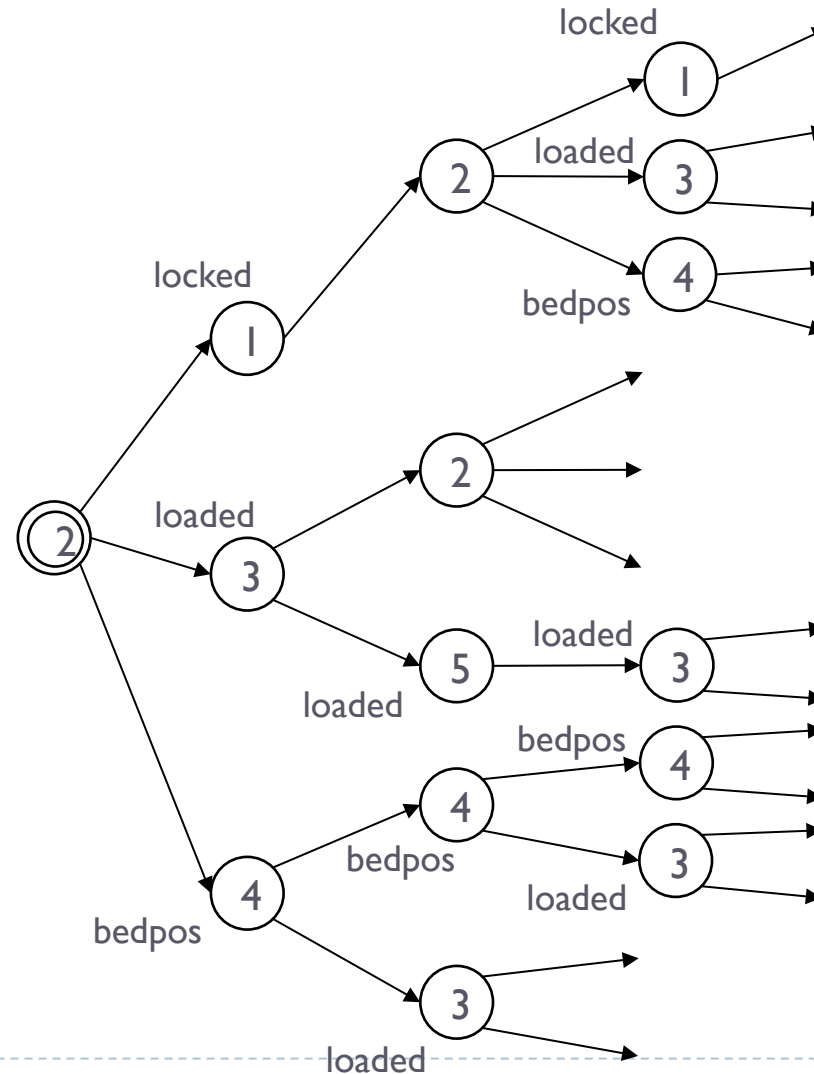


Property p is verified for all tracks (invariant) :**AG** p



Example :AG EF

It is always possible to reach a state where **loaded** is verified : **AG EF loaded**





Fault Forecasting



Estimation of presence, creation and consequences of faults

Identification, analysis of consequences, and classification of failures

Ordinal or qualitative evaluation

Probabilistic evaluation of the extent to which some dependability attributes are satisfied

Probabilistic or quantitative evaluation

Modelling

Behavior model of system / failure, maintenance actions, solicitations

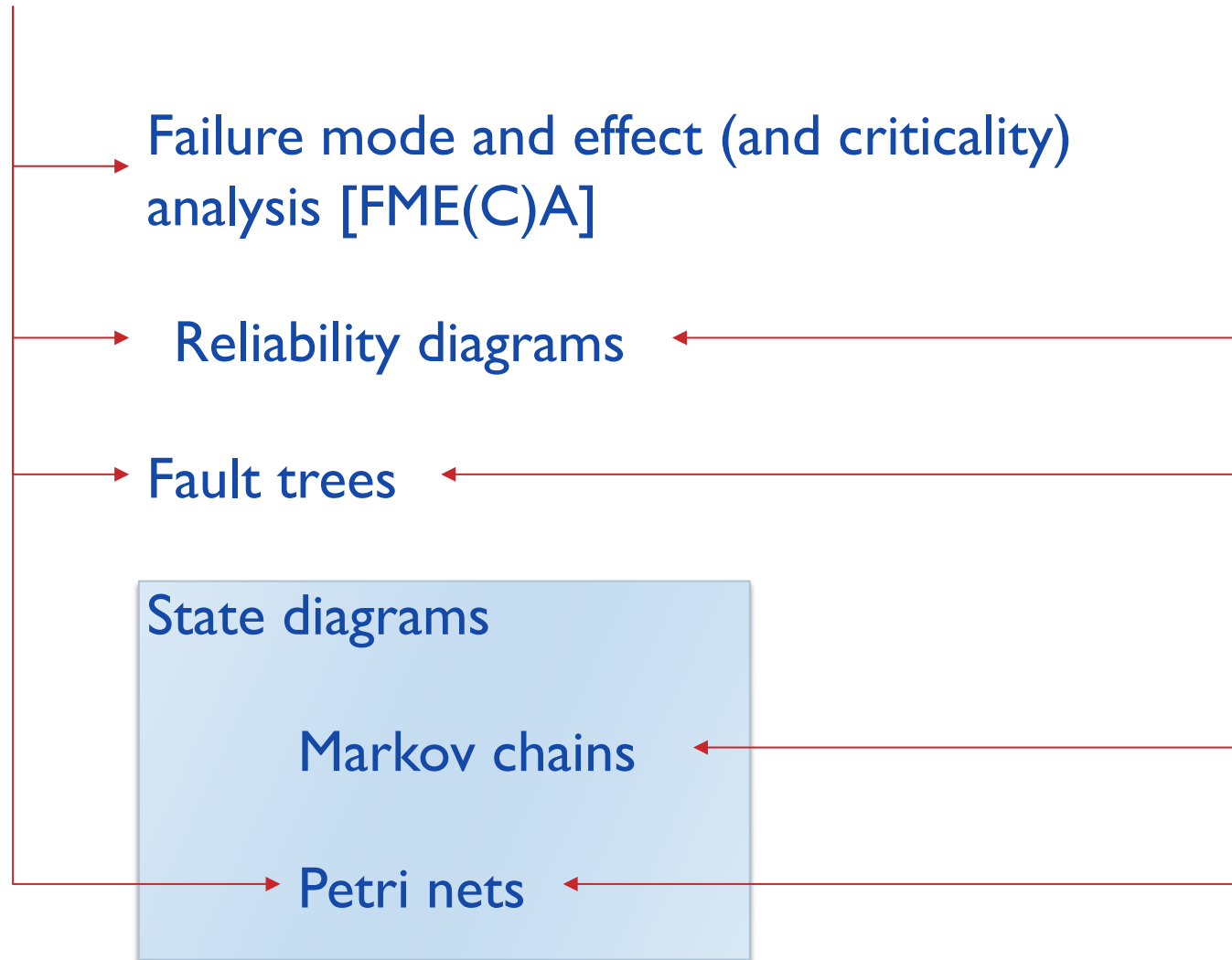
Operational testing

Evaluation test according to operational input profile



Ordinal evaluation

Probabilistic evaluation



Failure mode and effect (and criticality) analysis



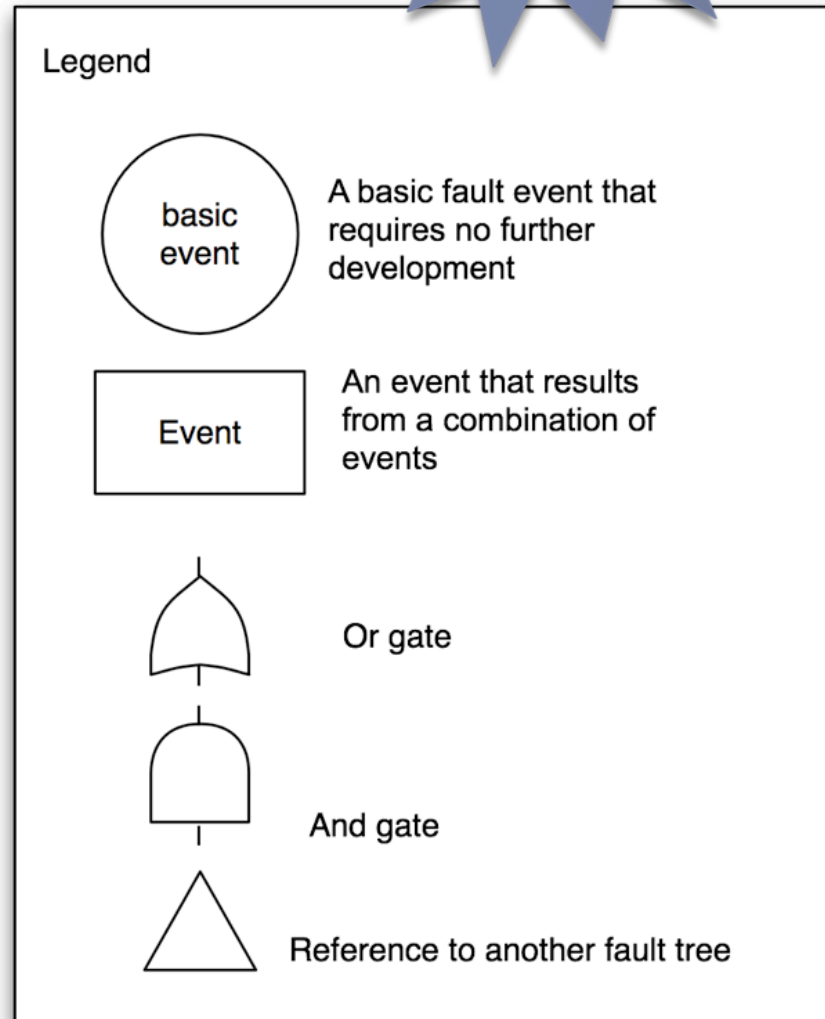
- ▶ Identification for each component:
 - ▶ Failure modes
 - ▶ Consequences of failures
- ▶ Example : interplanetary sonde (JPL-NASA)

Item / Function	FEA/EAA
Potential Failure Mode	CPU Reset
Potential Effect(s) of Failure	Loss of all state information
Severity	0.5
Potential Cause(s) Mechanisms of Failure	Power surge or drop; internal software error
Probability	0.3
Current design controls	Store state information
Effectiveness	0.9

Fault tree analysis

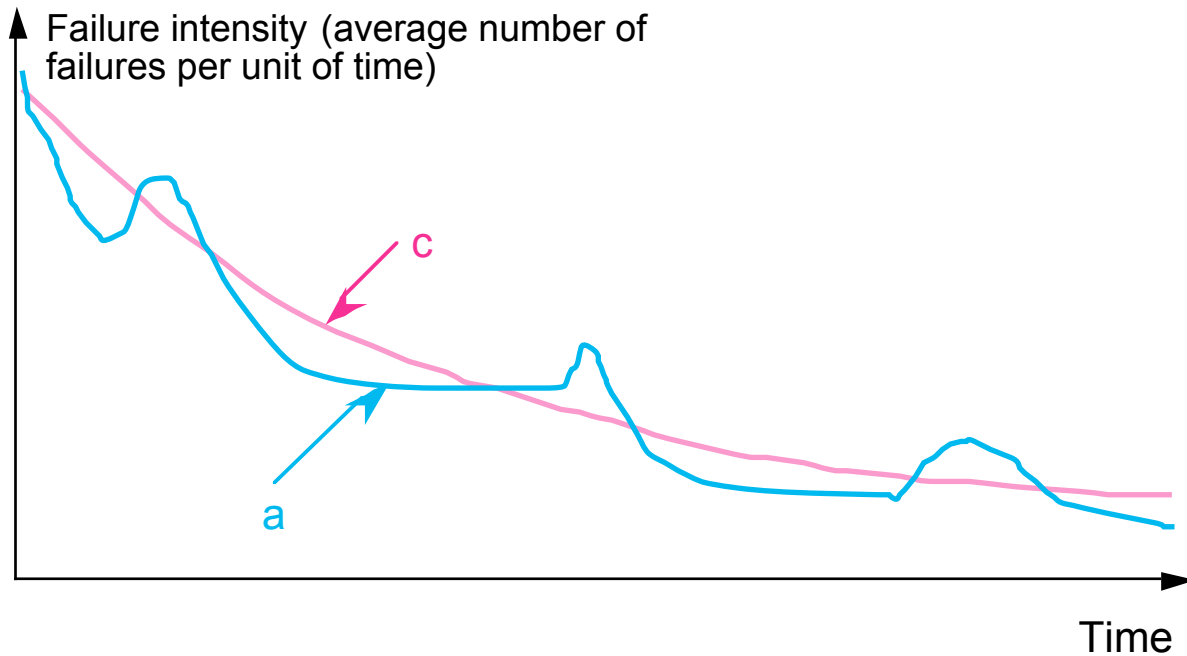
SEE
CHAPTER
4

- ▶ Top-down approach to failure analysis starting with an undesirable event called a top event, and then determining how this top event can be caused by individual or combined lower level failures or events



From qualitative to quantitative

- ▶ Fault forecasting : use of mathematical tools for calculation of reliability and availability
- ▶ Statistics and probabilities



Constant failure intensity

Stable reliability

Preserved ability to deliver correct service
[stochastic equality of times to failure]

Stationary stochastic processes

Decreasing failure intensity

Reliability growth

Improved ability to deliver correct service
[stochastic increase of times to failure]

Increasing failure intensity

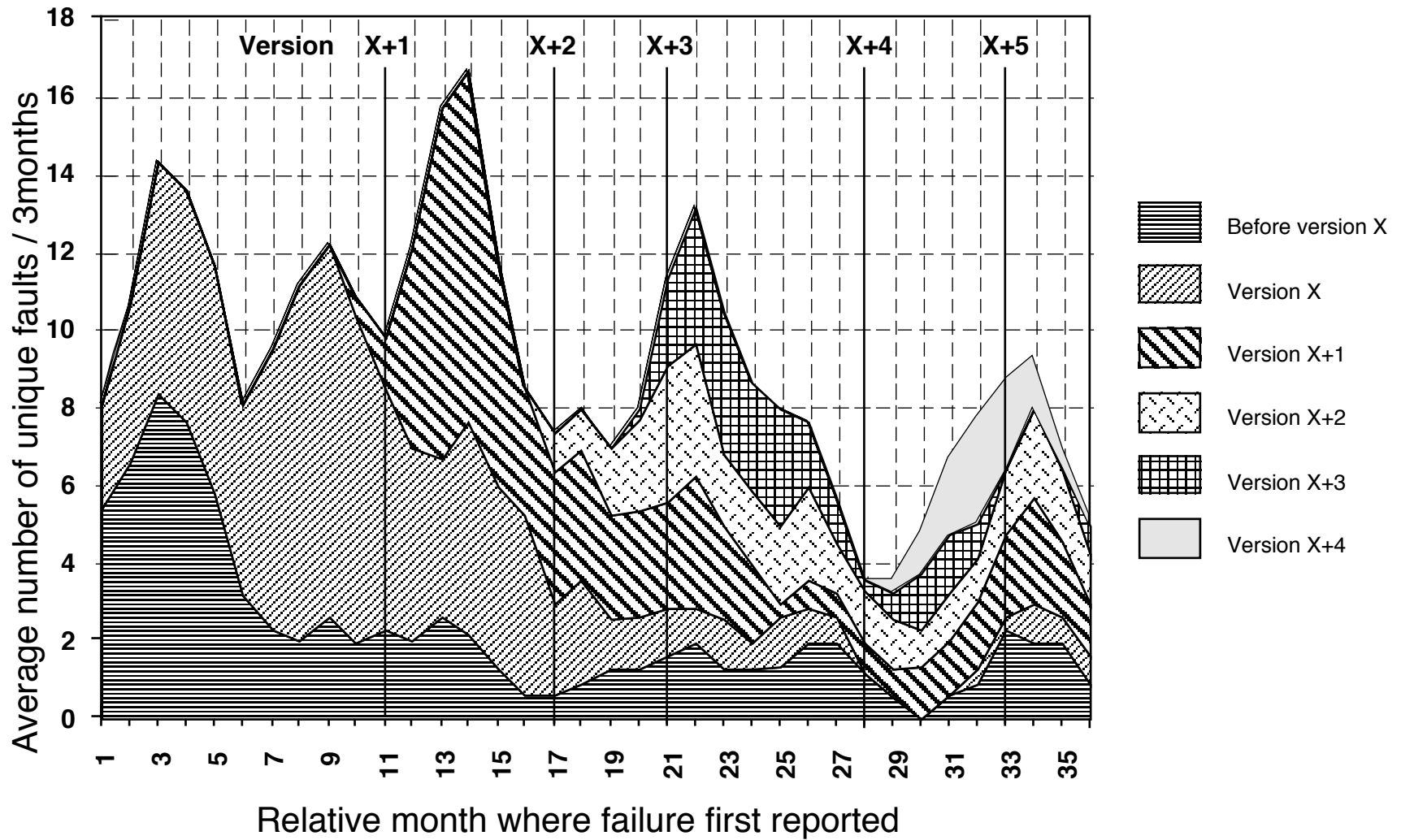
Reliability decrease

Degraded ability to deliver correct service
[stochastic decrease of times to failure]

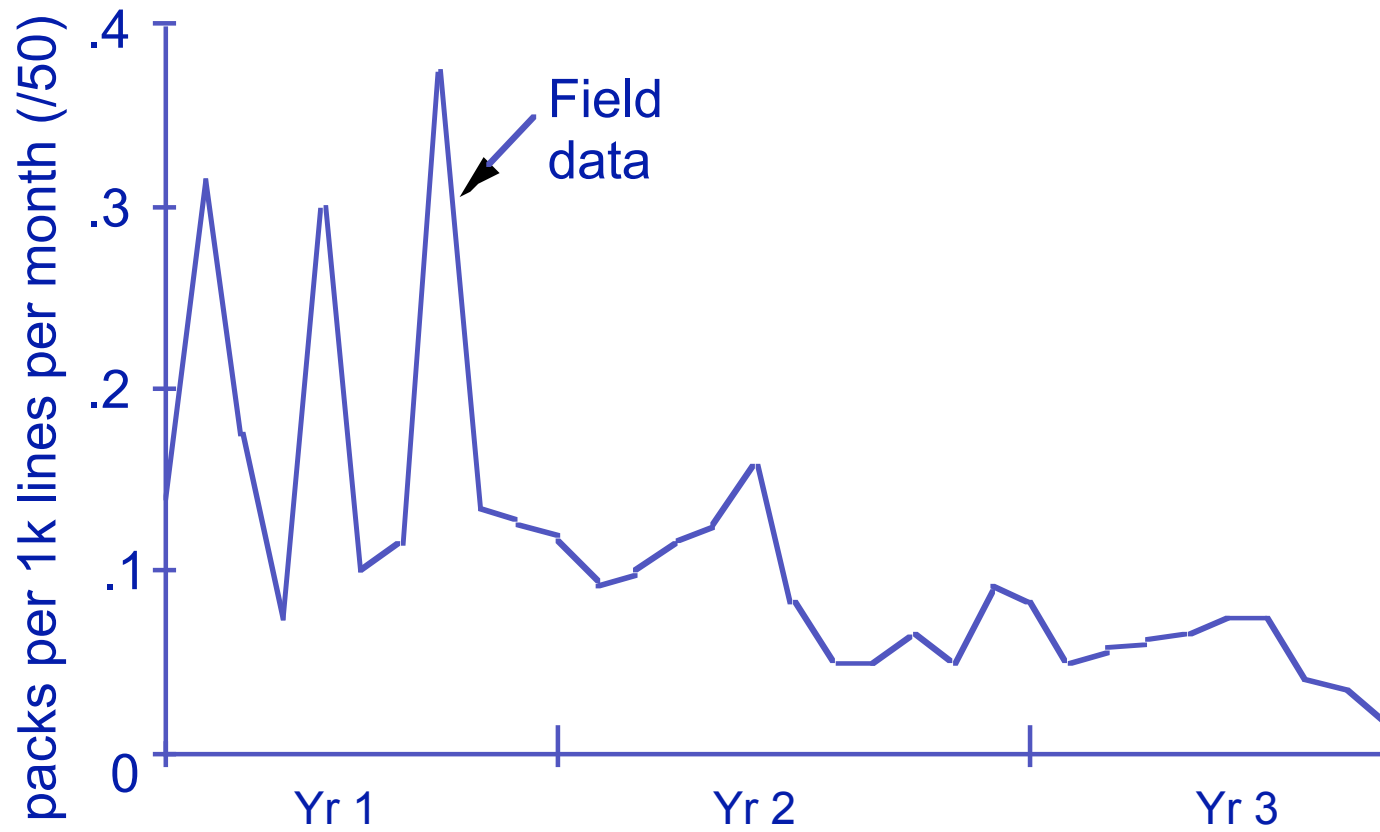
Non-stationary stochastic processes



IBM Data

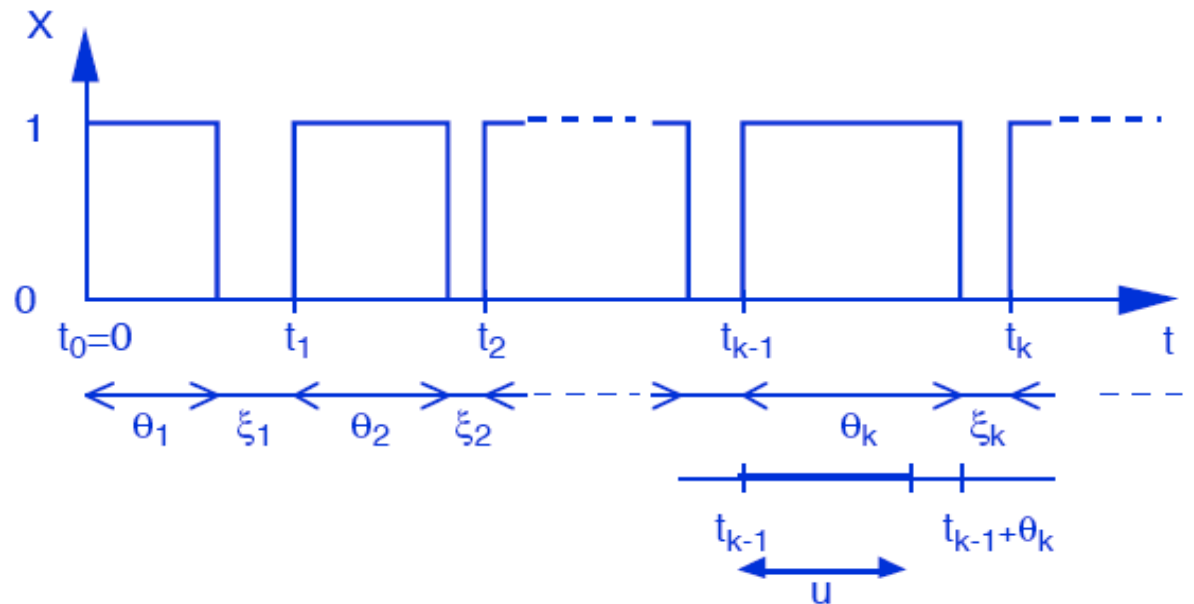


AT&T Data



Hardware replacement rate trend





Reliability

$$R_k(u) = P\{X(v) = 1, \forall v \in [t_{k-1}, t_{k-1} + u]\} = P\{\theta_k > u\} \quad \text{Initial reliability: } k=1$$

Availability

$$A(t) = P\{X(t) = 1\} = E\{X(t)\} = \sum_{k=1}^{\infty} P\{t_{k-1} < t < t_{k-1} + \theta_k\}$$

θ : time of realization until occurrence of event X

Fonction	Symbole	Definition	Statistical estimation
Distribution	$F(t)$	$P\{\theta \leq t\}$	$\frac{N(0) - N(t)}{N(0)}$
Complementary distribution	$\bar{F}(t)$	$P\{\theta > t\}$	$\frac{N(t)}{N(0)}$
Probability of density	$f(t)$	$\frac{P\{\theta \leq t + \Delta t\} - P\{\theta \leq t\}}{\Delta t}$	$\frac{N(t + \Delta t) - N(t)}{N(0) \Delta t}$
Chance rate	$z(t)$	$\frac{P\{\theta \leq t + \Delta t \mid \theta \geq t\}}{\Delta t}$	$\frac{N(t + \Delta t) - N(t)}{N(t) \Delta t}$

Mean time to occurrence of event X :

$$E(\theta) = \int_0^{\infty} t f(t) dt = \int_0^{\infty} \bar{F}(t) dt$$

$N(0)$: cardinality of the sample

$N(t)$: number of systems of the sample where event X does not occur

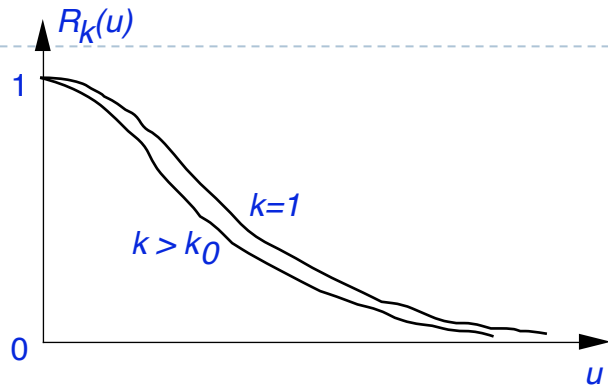
	$F(t)$	$\bar{F}(t)$	$f(t)$	$z(t)$
$F(t)$		$1 - \bar{F}(t)$	$\int_0^t f(v) dv$	$1 - e^{-\int_0^t z(v) dv}$
$\bar{F}(t)$	$1 - F(t)$		$\int_t^\infty f(v) dv$	$e^{-\int_0^t z(v) dv}$
$f(t)$	$\frac{dF(t)}{dt}$	$-\frac{d\bar{F}(t)}{dt}$		$z(t) e^{-\int_0^t z(v) dv}$
$z(t)$	$\frac{1}{1-F(t)} \frac{dF(t)}{dt}$	$-\frac{1}{\bar{F}(t)} \frac{d\bar{F}(t)}{dt}$	$\frac{f(t)}{\int_t^\infty f(v) dv}$	

Hypothesis : exponential distribution of θ :

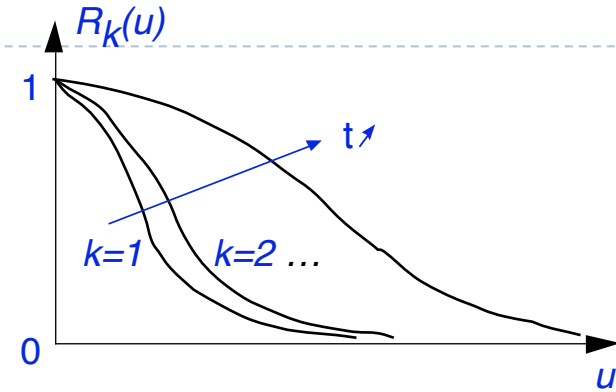
$$F(t) = 1 - e^{-\lambda t} \quad \bar{F}(t) = e^{-\lambda t} \quad f(t) = \lambda e^{-\lambda t} \quad z(t) = \lambda \quad E(\theta) = \frac{1}{\lambda}$$

Reliability

Stable reliability



Growth reliability



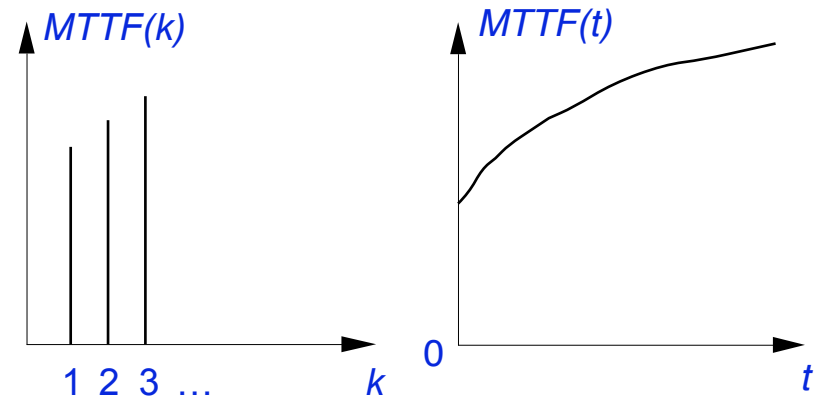
Mean time until next failure

$$MTTF_K = E\{\theta_k\} = \int_0^{\infty} R_K(u) du$$

Mean time to first failure :

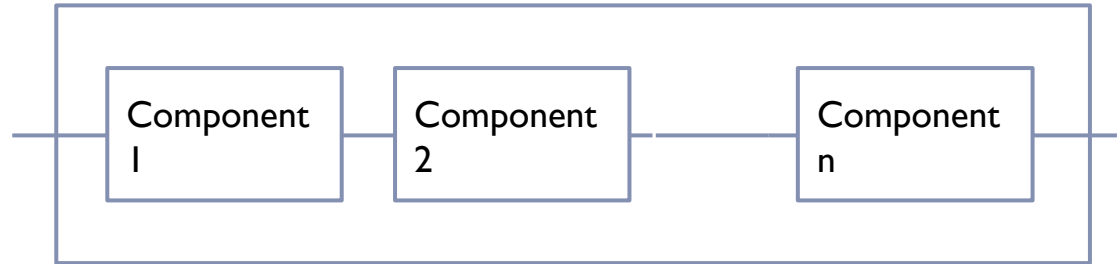
$$MTFF = MTTF_1 = MTTF(0) = \int_0^{\infty} R_1(u) du$$

MTTF : Mean Time To Failure, MTFF :
Mean Time to First Failure,

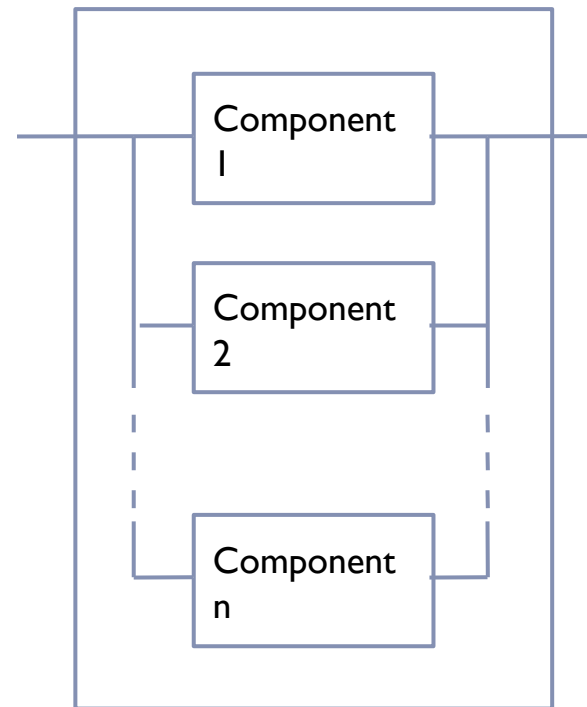


Reliability diagrams

Serial : non redundant systems



Paralell: redundant system



Model calculation

R_k : reliability of the component k ,
 $k=1, \dots, n$

R : reliability of the system

Serial systems

$R = P \{ \text{System without any failure} \}$

$R = P \{ \text{component 1 with no failure and } \dots \text{ and component } n \text{ without failure} \}$

If failure are stochastically independent

$$\longrightarrow R = \prod_{k=1}^n P \{ \text{component } k \text{ without failure} \}$$

$$R = \prod_{k=1}^n R_k$$

$$R_k(t) = \exp \left[- \int_0^t \lambda_k(v) dv \right]$$

$$R(t) = \exp \left[- \sum_{k=1}^n \int_0^t \lambda_k(v) dv \right]$$

$$\lambda(t) = \sum_{k=1}^n \lambda_k(t)$$

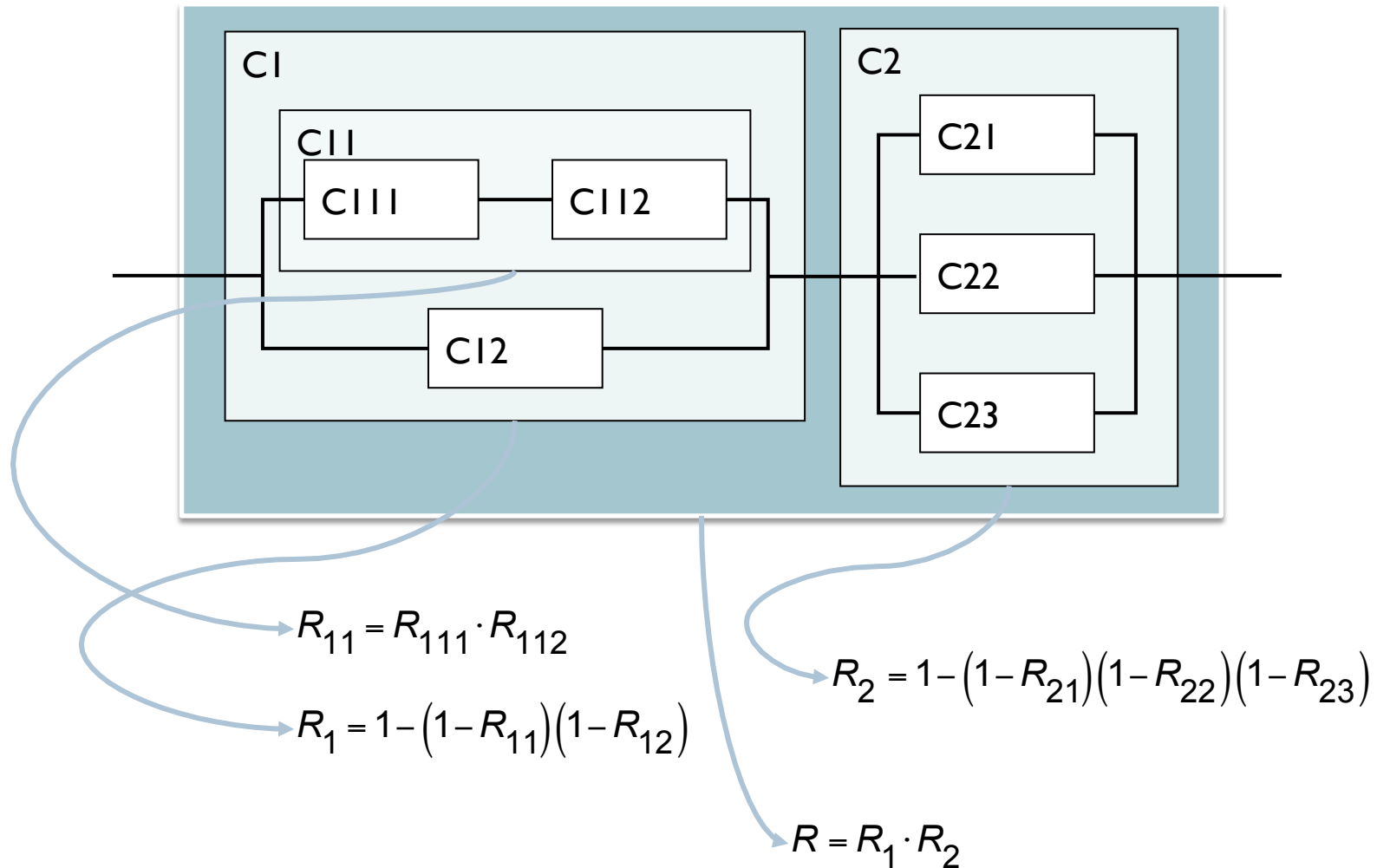
Parallel systems

System failure only if failure of all the components

$$1 - R = \prod_{k=1}^n (1 - R_k)$$

$$R = 1 - \prod_{k=1}^n (1 - R_k)$$

Serial-Parallel systems



Availability estimation

A_k : availability component k, $k=1, \dots, n$

A : availability of the system

Serial Systems

$$A = \prod_{k=1}^n A_k$$

Parallel systems

$$A = 1 - \prod_{k=1}^n (1 - A_k)$$



Fault Tolerance

Delivering service implementing system function in spite of faults

- **Error detection:** identification of error presence
- **System recovery:** transformation of erroneous state in a state free from detected error and from fault that can be activated again
 - **Error handling:** error removal from system state, if possible before failure occurrence
 - **Fault handling:** avoiding fault(s) to be activated again



Error detection

Concurrent detection, during service delivery

Addition of error detection mechanisms in component



Self-checking component

Preemptive detection: service delivery suspended, search for latent errors and dormant faults

Error handling

Rollback: brings the system back into a state saved prior to error occurrence

Saved state: recovery point

Rollforward: new state (free from detected error) found

Compensation: erroneous state contains enough redundancy for enabling error masking



Fault handling

Diagnostics: identifies and records the error cause(s), in terms of localisation and category

Isolation: performs physical or logical exclusion of the faulty component(s) from further contribution to service delivery, i.e., makes the fault(s) dormant

Reconfiguration: either switches in spare components or reassigns tasks among non-failed components

Reinitialization: checks, updates and records the new configuration, and updates system tables and records

👉 Intermittent faults

➤ Isolation and reconfiguration not necessary

➤ Identification

Error handling

Fault diagnosis

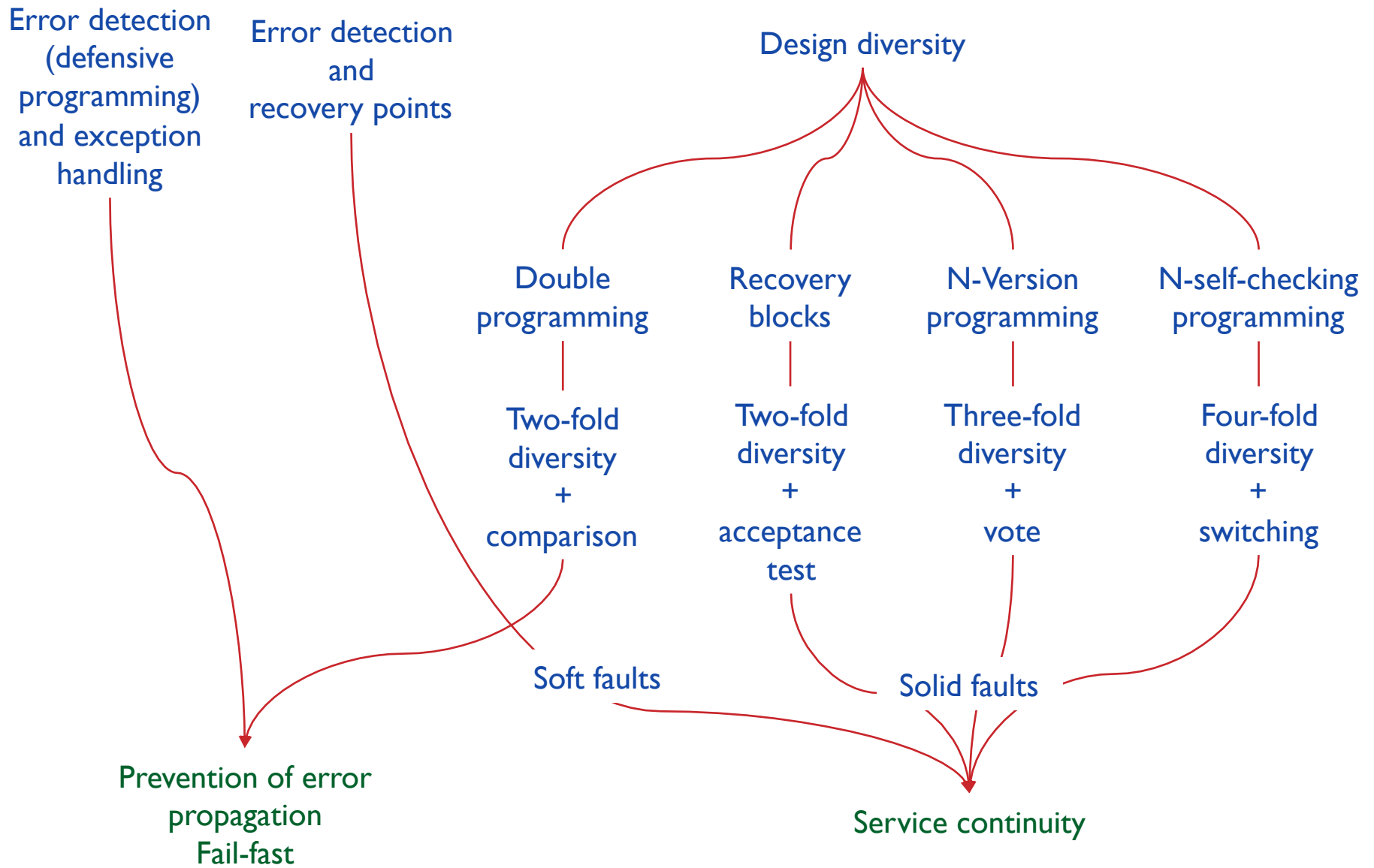
Non recurrence of error

Absence of fault

Intermittent fault



Development of fault tolerance



Design diversity

➤ Aim: failure independency

👉 Obstacles: common specification, common difficulties, inter-variant synchronizations and decisions

➤ Operational use

👉 Civil avionics: generalized, at differing levels

👉 Railway signalling: partial

👉 Nuclear control: partial

➤ Dependability improvement

👉 Gain (max for physical fault tolerance compared to SW)

👉 Contribution to verification of specification



Example 1 Redundancy

- ▶ A fail-safe dual channel robot control for surgery applications

- ▶ U. Laible et al. / Safety Science 4 (2004) 423–436

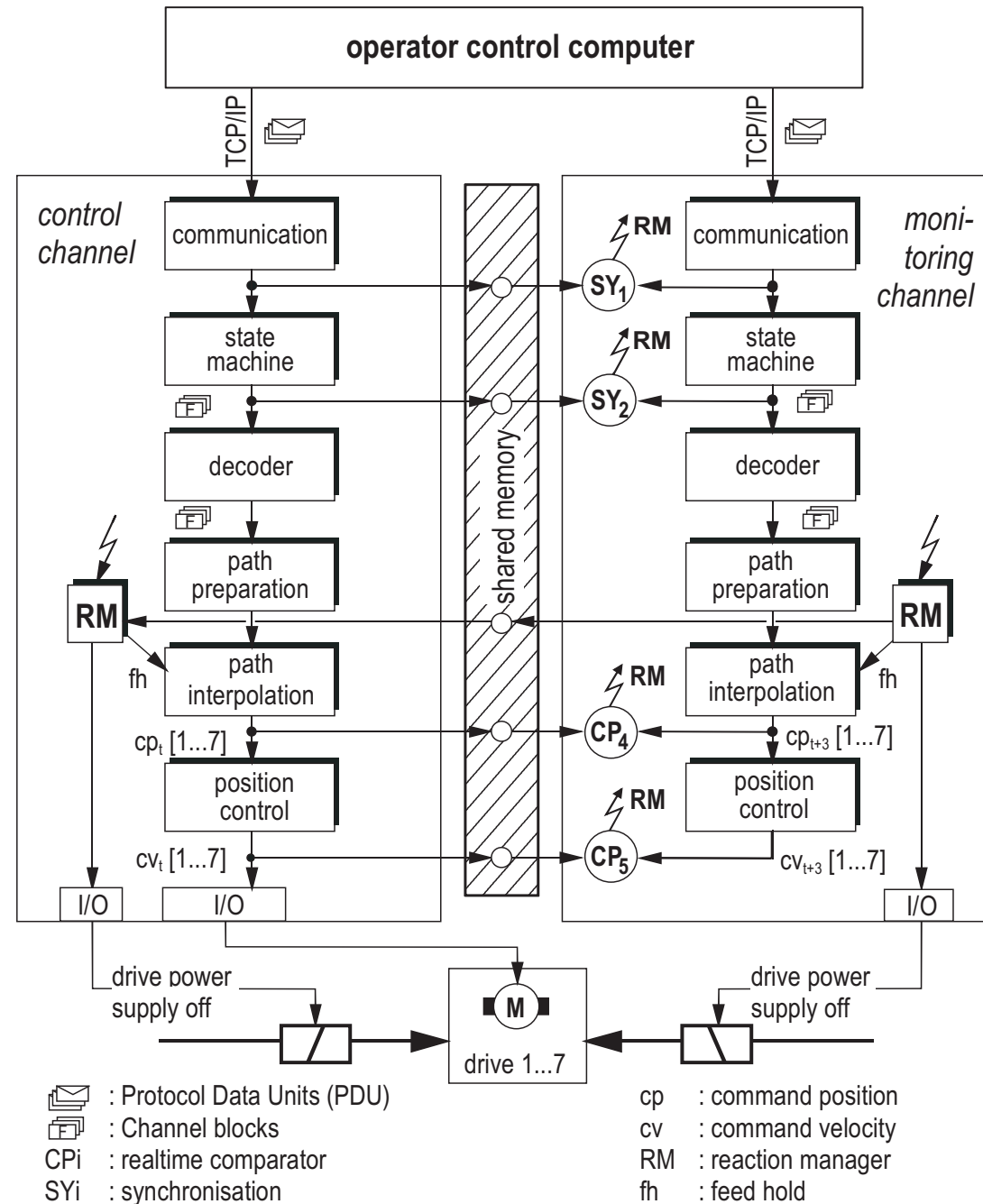
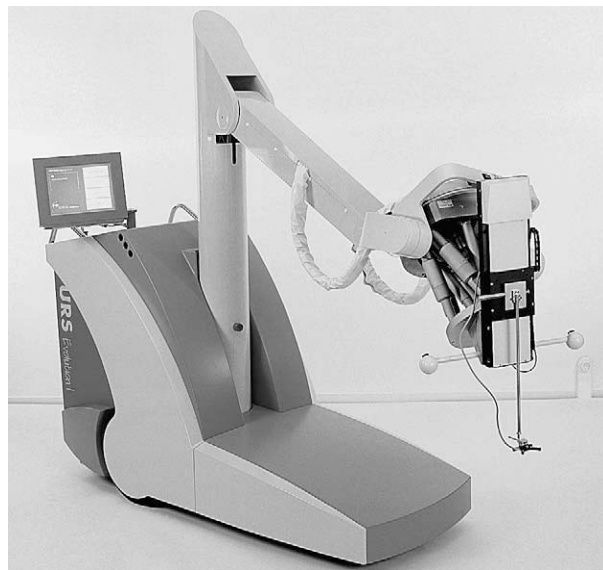
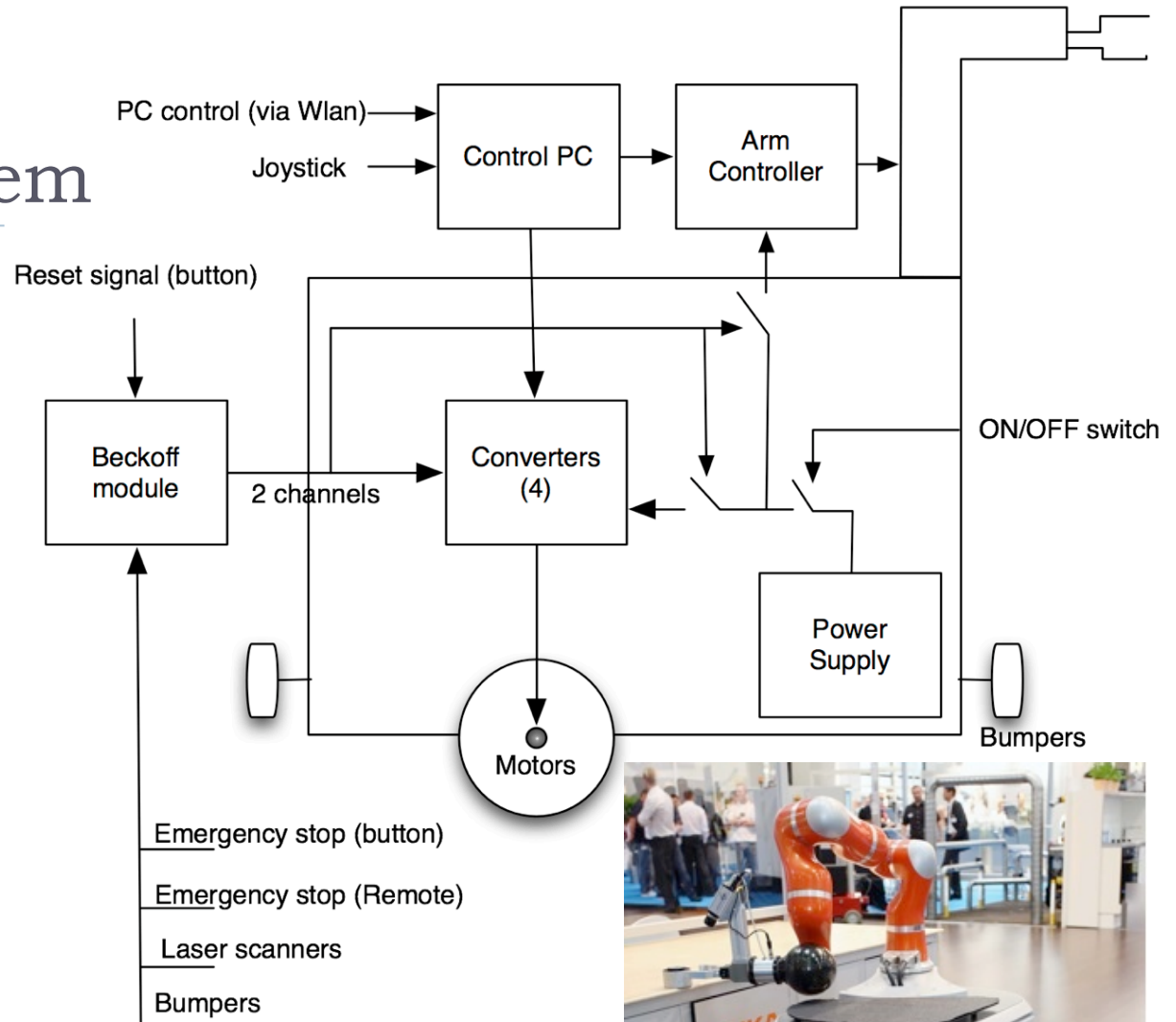


Fig. 7. Redundant RC compares command position values.

Example 2

Protection System

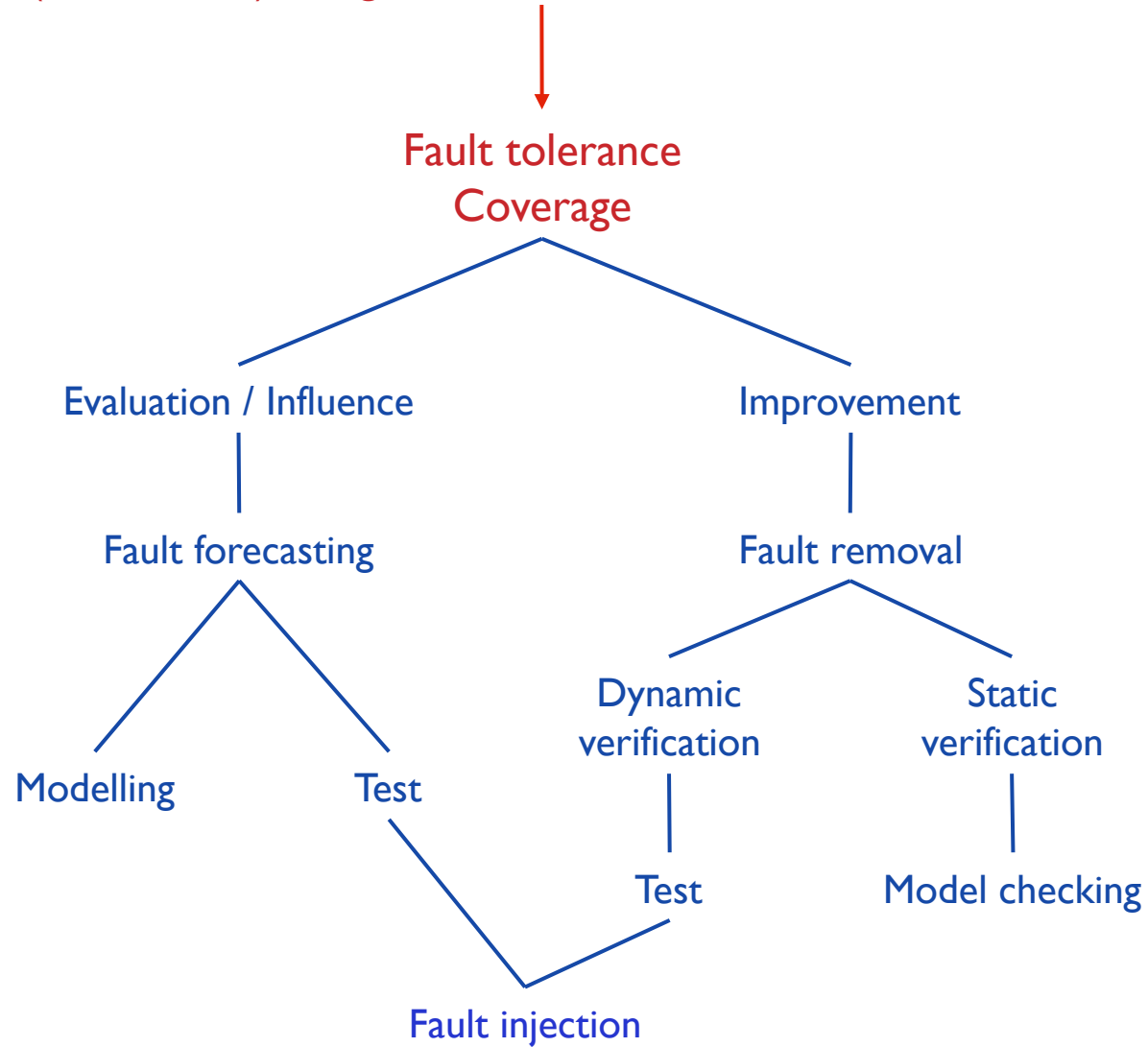
- ▶ Safety PLC (Programmable Logical Controller)
- ▶ Cut power of the robot arm => no power : the brakes are engaged in the robot arm.
- ▶ Command the converters of the motors to slow down.
- ▶ After a delay, the power going towards the motor via the converters is cut => no power on that line : the brakes on the wheel motors are engaged.



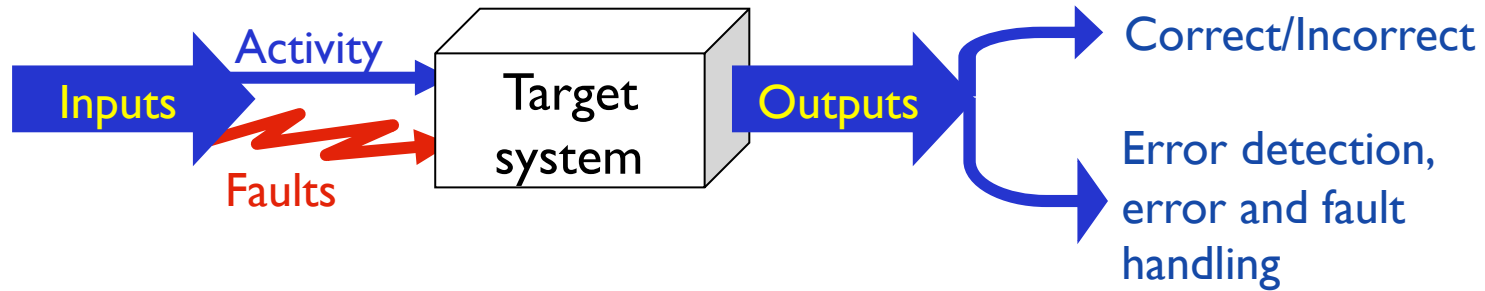
KUKA omnirob© concept

Verification and evaluation of fault tolerance

Faults (deficiencies) in algorithms and mechanisms of fault tolerance



Fault injection



		Target system		
		Simulation model	Prototype or actual system	
Injection	Informational	In simulation	By software <ul style="list-style-type: none"> •Memory •Executive •Processor 	❖ Representativity of faults
	Physical		By hardware <ul style="list-style-type: none"> •Radiations •Interferences •Pins 	





Fault prevention

Prevention techniques

- ▶ **Requirement/specification/analysis/design methods**
 - ▶ Semi-formal (UML)
 - ▶ Formal (Petri nets, automata, ESTEREL, LUSTRE)
 - ▶ Certified technologies and tools (High integrity components, code generation, safety kernels)
- ▶ **Application of development process methods**
 - ▶ Unified Process (Rational Unified Process)
 - ▶ Agile methods
- ▶ **Preventive maintenance**

Too many technologies and methods to be exhaustive
→ fault prevention is all actions aiming at increase quality of the system development and life

References

(in addition to those mentioned in the slides)

A. Avizienis, J.C. Laprie, B. Randell, C. Landwehr, *Basic Concepts and Taxonomy of Dependable and Secure Computing*, IEEE Transactions on Dependable and Secure Computing, Vol. 1, no. 1, pp. 11-33, January-March 2004

J.C. Laprie, *From Dependability to Resilience*, 38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks, Anchorage, Alaska, June 2008, Sup. Vol., pp. G8-G9



Chapter 3. Risk Management



Definitions and concepts of risk management



Unwanted effects: harm

- ▶ **Harm:** *physical injury or damage to the health of people, or damage to property or the environment.*
- ▶ Three attributes of a harm are usually defined the **nature** of the harm, its **severity** and its **probability of occurrence**.



Nature of harm

Human part	Harm nature
Skeletal/Muscular System	Break large bones in primary skeletal structure Break small bones in extremities Break protective bone structure (e.g., skull, ribs) Cut muscle tissue Pierce muscle tissue Crush muscle tissue
Organs	Brain concussion Pierce lungs, heart, stomach, kidneys, etc. Cardiac arrest from electric shock Groin trauma
Circulatory System/Breathing	Arterial/vascular puncture Prevent diaphragm action Block airways
Mouth/Nose/Eyes/Ears	Pierce eyes Crushing trauma to eyes Break teeth Cut/tear facial/head tissue Break nose
Life Support/Ambient Conditions (non robot specific)	Asphyxiate Toxic substances High/low ambient temperatures High sonic energy

Table 1 - Nature of harm (from [KAR95])

Severity – Table Example

AIS*	Severity	Type of Injury
0	None	None
1	Minor	Superficial injury
2	Moderate	Recoverable
3	Serious	Possibly recoverable
4	Severe	Not fully recoverable without care
5	Critical	Not fully recoverable with care
6	Fatal	Not survivable

Table 1 - The Abbreviated Injury Scale from [AAAM 98]



Probability of occurrence or likelihood

Likelihood	Indicative probability (per year)
Frequent	>1
Probable	$1 - 10^{-1}$
Occasional	$10^{-1} - 10^{-2}$
Rare	$10^{-2} - 10^{-6}$
Almost Impossible	$<10^{-6}$

Table 1 – Indicative probability values for likelihood estimation



Risk

Risk: combination of the probability of occurrence of harm and the severity of that harm.

Tolerable risk: risk which is accepted in a given context based on the current values of society.

Likelihood	Severity						
	6 Fatal	5 Critical	4 Severe	3 Serious	2 Moderate	1 Minor	0 None
Frequent	H	H	H	H	H	I	N
Probable	H	H	H	H	I	L	N
Occasional	H	H	H	I	L	N	N
Rare	H	H	I	L	N	N	N
Improbable	I	I	L	N	N	N	N
Almost Impossible	L	L	N	N	N	N	N

H -: High, I - Intermediate, L - Low, N -Negligible

Other Risk Estimation

$$R = N \times C \times F \times Q$$

- ▶ *R: risk related to the considered hazard*
- ▶ *Q: probability of occurrence of harm*
- ▶ *F: frequency and duration of exposure*
- ▶ *C: severity of possible harm that can result*
- ▶ *N: number of exposed people*



Safety

- ▶ **Safety:** freedom from unacceptable risk.

Likelihood	Severity						
	6	5	4	3	2	1	0
	Fatal	Critical	Severe	Serious	Moderate	Minor	None
Frequent	H	H	H	H	H	I	N
Probable	H	H	H	H	I	L	N
Occasional	H	H	H	I	L	N	N
Rare	H	H	I	L	N	N	N
Improbable	I	I	L	N	N	N	N
Almost Impossible	L	L	N	N	N	N	N

H -: High, I - Intermediate, L - Low, N -Negligible



Council Directive 93/42/EEC of 14 June 1993 concerning medical devices

- ▶ **Medical device** means any instrument,[..] material or other article, whether used alone or in combination, including the software necessary for its proper application intended [...] to be used for human beings for the purpose of:
 - ▶ diagnosis, prevention, monitoring, treatment or alleviation of disease,
 - ▶ diagnosis, monitoring, treatment, alleviation of or compensation for an injury or handicap,
 - ▶ investigation, replacement or modification of the anatomy or of a physiological process,
 - ▶ [...];



Council Directive 93/42/EEC of 14 June 1993 concerning medical devices

- ▶ **Active medical device** : Any medical device operation of which depends on a source of electrical energy or any source of power other than that directly generated by the human body or gravity and which acts by converting this energy.

Many rehabilitation robots can be considered as active medical device

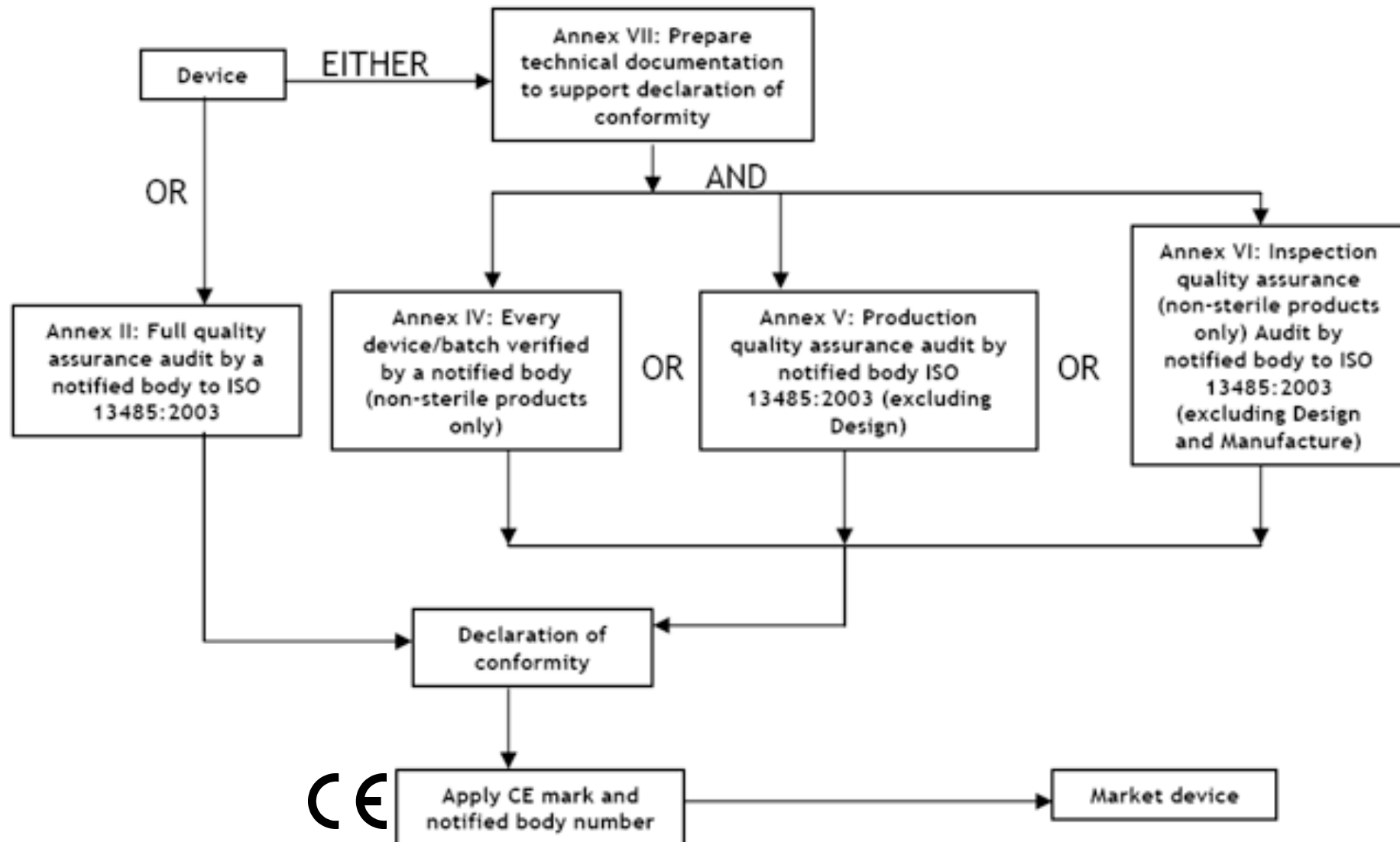
Directive 93/42/EEC classes

I. Determine system class :

- ▶ The classification rules are based on the vulnerability of the human body taking account of the potential risks associated with the technical design and manufacture of the devices
 - ▶ Class I : non invasive devices, unless specific rules applied...
 - ▶ Class IIa : invasive devices intended for short-term, unless...
 - ▶ Class IIb : implantable devices and long-term surgically invasive devices
 - ▶ Class III : implantable devices and long-term surgically invasive devices near heart or of the central circulatory system
- ▶ *Rule 9 : All active therapeutic devices intended to administer or exchange energy are in Class IIa*



CLASS IIa MEDICAL DEVICES - CE MARKING ROUTES



Other system classes ([PHRIENDS project](http://www.phriends.org) www.phriends.org) - **NON MEDICAL**

- ▶ **Class I - Far (no pHRI possible):** Human and robot do not share the same workspace so direct physical contact is not possible. (Remote control, pendant, network, etc.)
 - ▶ **Class IIa - Close (accidental pHRI possible):** Human and robot share the same workspace. (e.g. the programming of the robot system while the programmer is within the robot's work cell, or exchange of an object through a table)
 - ▶ **Class IIb - Touching without simultaneous movement (direct or indirect pHRI):** The robot shares its workspace with the human., But physical contact with the moving robot is avoided. In this category, interaction only takes place when the robot stops. (e.g. the system approaches the human, the robot (safely) stops temporarily when the human reaches for the object and only starts moving again after the interaction is completed)
-



Other system classes (PHRIENDS project) – (2)

- ▶ **Class IIc - Touching with simultaneous movement (direct or indirect pHRI):** The robot shares its workspace with the human. Both are moving simultaneously and physical interaction is possible and intended. (robot that is programmed by being manually guided through the workspace / robot assisting the human with its greater force and/or precision)
- ▶ **Class III - Supporting (direct pHRI):** physical interaction occurs continuously over extended periods of time, usually in the form of exoskeletons which are worn by the user, or when the robot is carrying a human (for example, in entertainment or healthcare applications, or rescue operations).



Causes of harm: hazards

- ▶ **Hazard: potential source of harm**

- ▶ Hazardous inherent characteristics (e.g., a cutting edge, a toxic substance, etc.)
- ▶ Hazardous controllable states of the system (e.g., hazardous motion, suspended mass)
- ▶ Failure of hardware or software components
- ▶ Human errors
- ▶ Unspecified external events
- ▶ The term **hazardous motion** is defined in the standard [ISO 10218:2006] to be “*any motion that is likely to cause personal physical injury or damage to health*”



Causes of harm: hazards (cont'd)

- ▶ **Hazardous situation:** *circumstance in which people, property or the environment are exposed to one or more hazards*
- ▶ **Harmful event or accident:** *occurrence in which a hazardous situation results in harm*
- ▶ **Incident:** *event that does not lead to harm, but which has the potential to create harm in other circumstances*



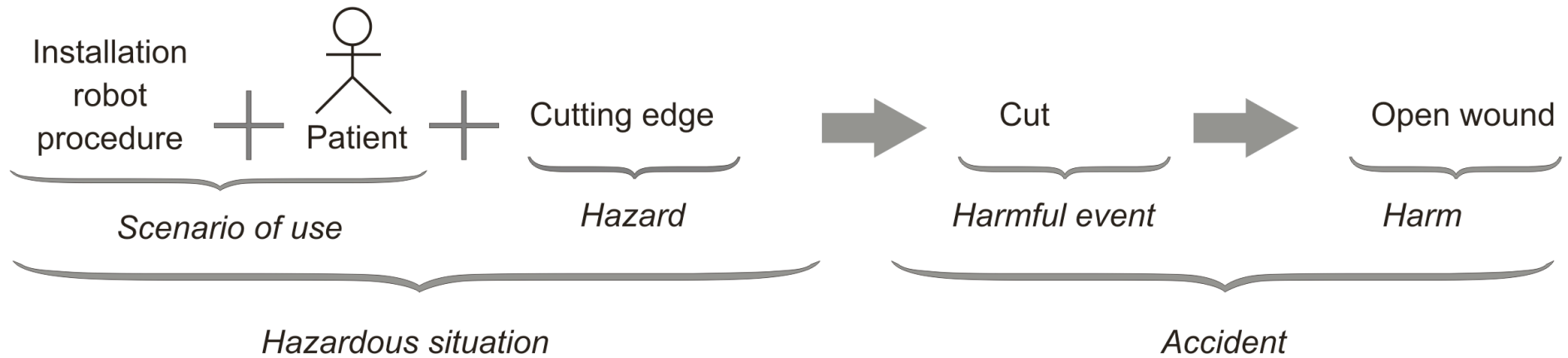
Subset of list of significant hazards (Extracted from ISO 10218 Annex A, Table A.1 – List of significant hazards which is itself based on Annex A of ISO 14121:1999).

No.	Description	Example(s) of related hazardous situations	Related danger zone
1	Mechanical hazards		
1.1	Crushing	Movement (normal or singularity) of any part of the robot arm or additional axes	Restricted space
1.2	Shearing	Movement of additional axes	Around accessory equipment
1.3	Cutting or severing	Movement or rotation creating scissors action	Restricted space
1.4	Entanglement	Rotation of wrist or additional axes	Restricted space
1.5	Drawing-in or trapping	Between robot arm and any fixed object	Around fixed objects close to restricted space
1.6	Impact	Movement (normal or singularity) of any part of the robot arm	Restricted space
2	Electrical hazards		
2.1	Contact of persons with live parts (direct contact)	Contact with live parts or connections	Electrical cabinet, terminal boxes, control panels at machine



8	Hazards generated by neglecting ergonomic principles in the design process		
8.1	Unhealthy postures or excessive effort (repetitive strain)	Poorly designed teach pendant	Teach pendant
8.2	Inadequate consideration of hand-arm or foot-leg anatomy	Inappropriate location of controls	At load/unload work piece and tool mounting or setting positions
8.7	Inadequate design, location or identification of manual controls	Inadvertent operation of controls	At or near robot cell
8.8	Inadequate design or location of visual display units	Misinterpretation of displayed information	At or near robot cell
10	Unexpected start-up, unexpected overrun/over speed		
10.1	Failure/disorder of the energy source	Mechanical hazards associated with robot additional axes	At or near robot cell
10.2	Restoration of energy supply after an interruption	Unexpected movements of robot or additional axes	At or near robot cell
10.3	External influences on the electrical equipment	Unpredictable behavior of electronic controls due to electromagnetic interference	At or near robot cell
13	Failure of the power supply (external power sources)	Malfunctions of the control with consequent release of robot arm brake; release of brake causes robot elements to move under residual forces (inertia, gravity, spring/energy storage means) unexpectedly	At or near robot cell where robot elements retained in a safe condition by the application of power or fluid pressure
14	Failure of the control circuit (hardware or software)	Unexpected movements of robot or additional axes	At or near robot cell
18	Loss of stability, overturning of machinery	Unrestrained robot or additional axes (maintained in position by gravity), falls or overturns	At or near robot cell

Example of use of terminology



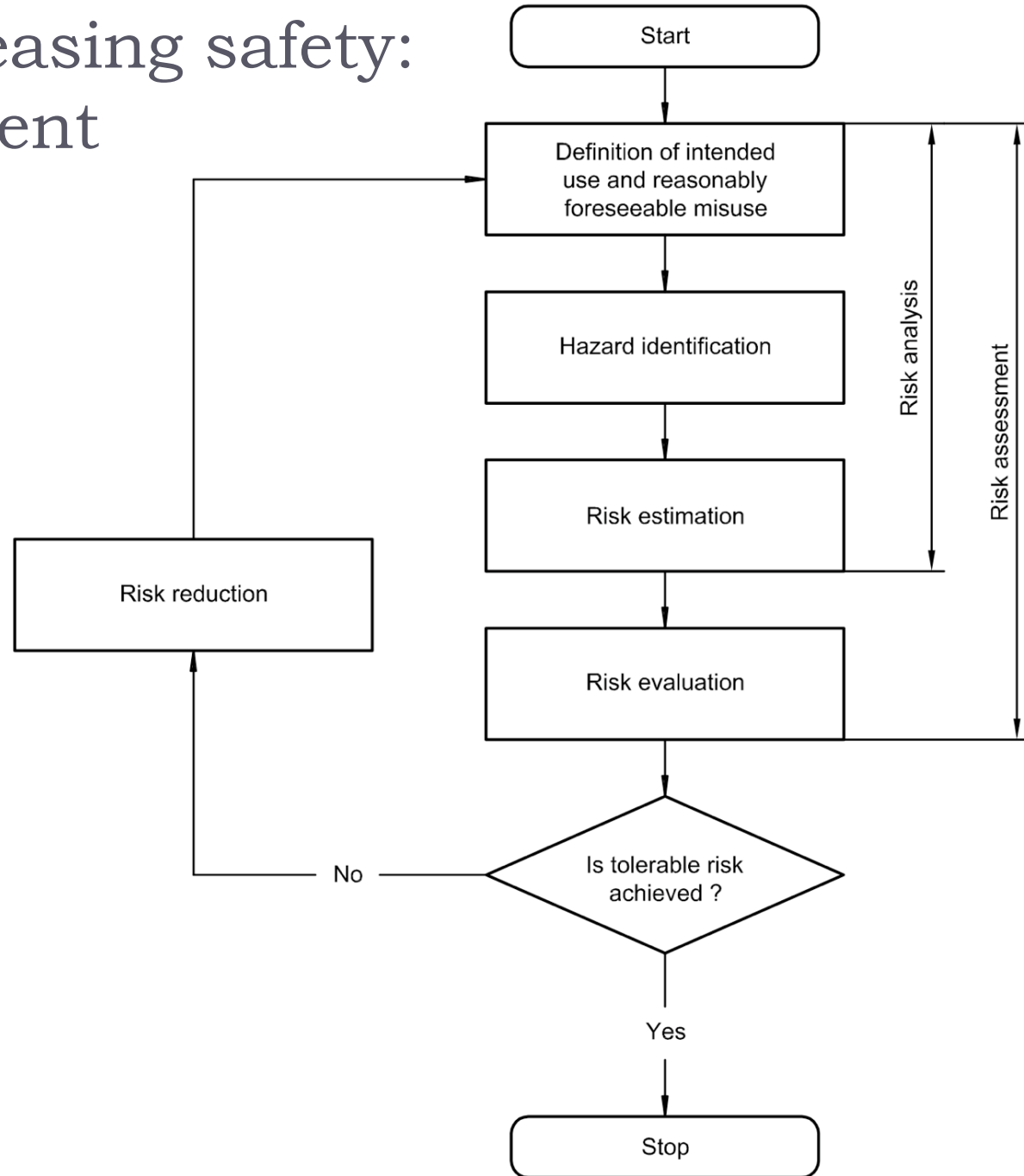


Risk management process



Means of increasing safety: risk management

- ▶ Risk management overview

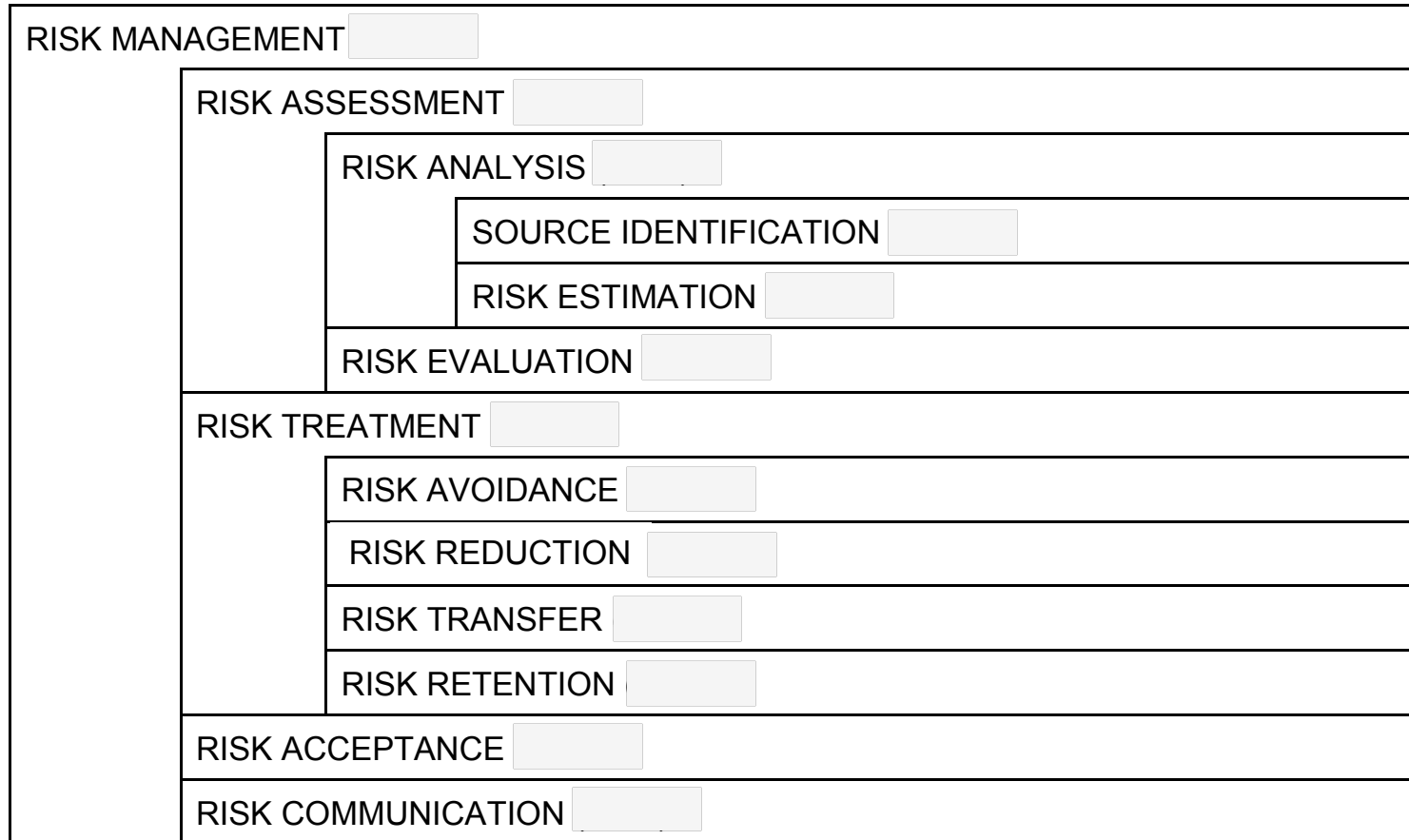


Risk management activities

- ▶ Risk management : coordinated activities to direct and control an organization with regard to risk
 - ▶ **Risk analysis** : systematic use of available information to identify hazards and to estimate the risk
 - ▶ **Risk Evaluation** : process of comparing the estimated risk against given risk criteria to determine the significance of the risk
 - ▶ **Risk treatment** : process of selection and implementation of measures to modify risk
 - ▶ Risk treatment measures can include reducing, avoiding, optimizing, transferring or retaining risk.
 - ▶ Risk reduction : actions taken to lessen the probability, negative consequences, or both, associated with a risk
 - ▶ (Risk communication, transfer, etc.)



Relationship between terms, based on their definitions regarding “Risk” (ISO Guide 73)



Summary of process to achieve tolerable risk

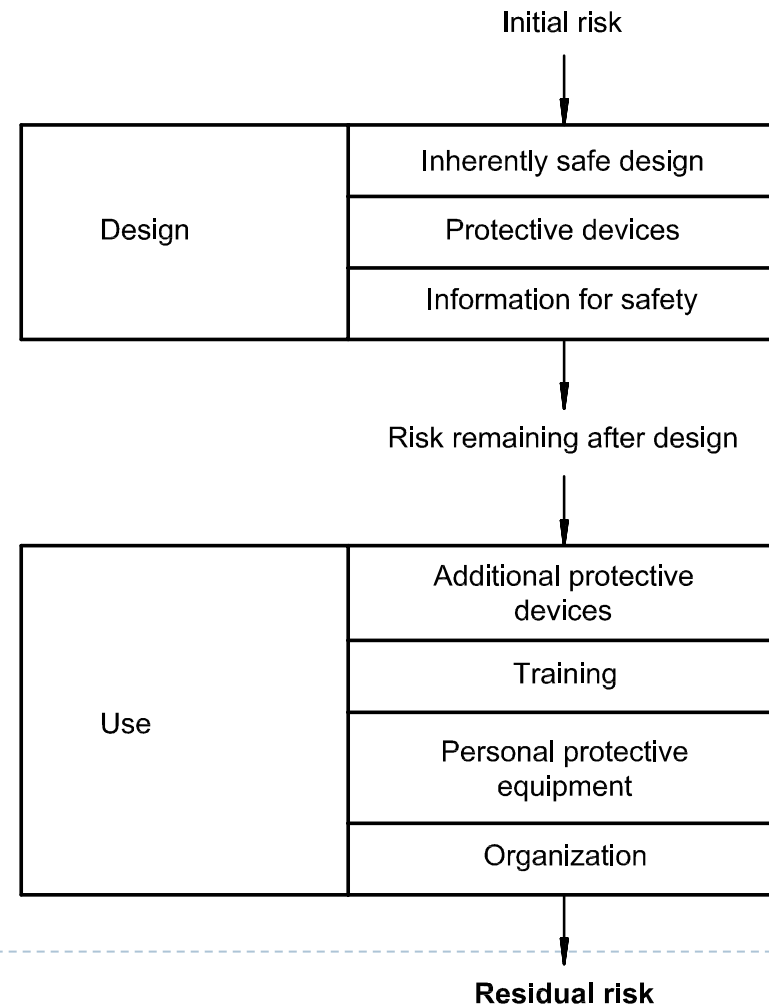
- ▶ **The following procedure should be used to reduce risks to a tolerable level:**
 - ▶ a) identify the likely user group(s) for the product, process or service (including those with special needs and the elderly), and any known contact group (e.g. use/contact by young children);
 - ▶ b) identify the intended use and assess the reasonably foreseeable misuse of the product, process or service;
 - ▶ c) identify each hazard (including any hazardous situation and harmful event) arising in all stages and conditions for the use of the product, process or service, including installation, maintenance, repair and destruction/disposal;
 - ▶ d) estimate and evaluate the risk to each identified user/contact group arising from the hazard(s) identified;
 - ▶ e) judge if the risk is tolerable (e.g. by comparison with similar products, processes or services);
 - ▶ f) if the risk is not tolerable, reduce the risk until it becomes tolerable.
-



Summary of process to achieve tolerable risk (cont'd)

▶ When reducing risks the order of priority should be as follows:

- ▶ 1) inherently safe design;
- ▶ 2) protective devices;
- ▶ 3) information for users.



Risk management activities

Risk analysis, Evaluation and Reduction

Risk analysis methods

- ▶ During risk analysis, various methods can be used to handle functional and technological issues, for example:
 - ▶ Preliminary Hazard Analysis (PHA)
 - ▶ HAZard OPerability (HAZOP)
 - ▶ Failure Modes Effects and Criticality Analysis (FMECA)
 - ▶ Fault Tree Analysis (FTA)
 - ▶ Event Tree Analysis (ETA)
- ▶ Widely used in many domains and particularly in industrial robotics
- ▶ They are also recommended in many standards on dependability
- ▶ See chapter 4 “Three risk analysis techniques”



Preliminary Hazard Analysis (PHA)

System:

Operating mode:

Ref.	Hazard	Accidental event (what, where, when)	Probable causes	Contingencies/ Preventive actions



Failure Modes and Effects Criticality Analysis (FMECA)

Failure mode number	Identification of Item or Function	a. Failure mode b. Failure cause	Failure effects a. Local or Subsystem b. Next Higher Level c. Real word effect (Mission)	Risk assessment			Remarks a. Failure detection method b. Compensating Features/actions c. Other
				O	S	R	



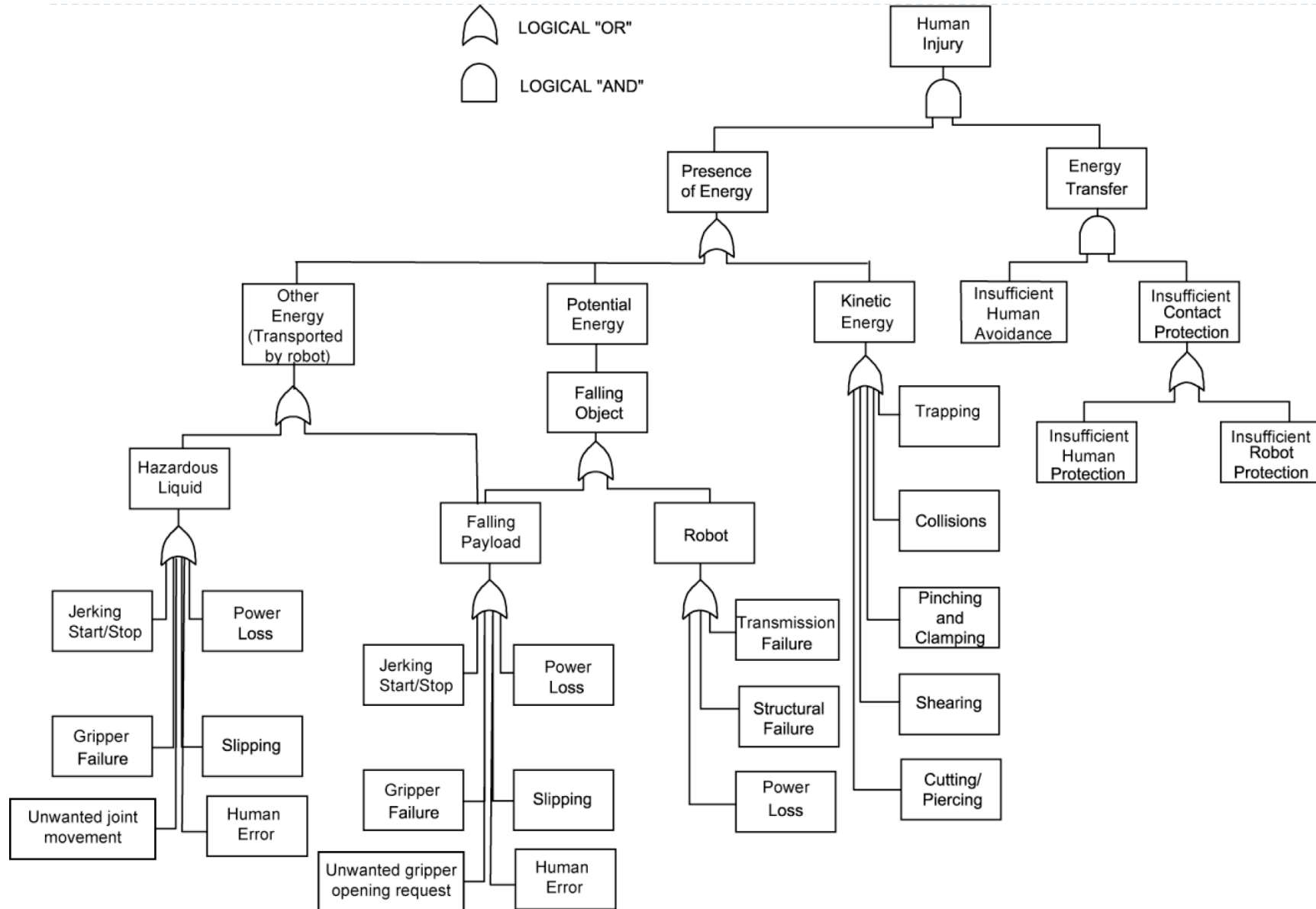
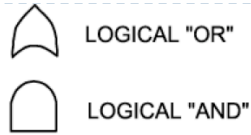
Hazard Operability (HAZOP)

No/None	Complete negation of the design intention No part of the intention is achieved and nothing else happens
More	Quantitative increase
Less	Quantitative decrease
As Well As	All the design intention is achieved together with additions
Part of	Only some of the design intention is achieved
Reverse	The logical opposite of the design intention is achieved
Other than	Complete substitution, where no part of the original intention is achieved but something quite different happens
Early	Something happens earlier than expected relative to clock time
Late	Something happens later than expected relative to clock time
Before	Something happens before it is expected, relating to order or sequence
After	After Something happens after it is expected, relating to order or sequence

Study title:						Page: of			
Drawing no.:			Rev no.:			Date:			
HAZOP team:						Meeting date:			
Part considered:									
Design intent:			Material: Source:			Activity: Destination:			
No.	Guide-word	Element	Deviation	Possible causes	Consequences	Safeguards	Comments	Actions required	Action allocated to

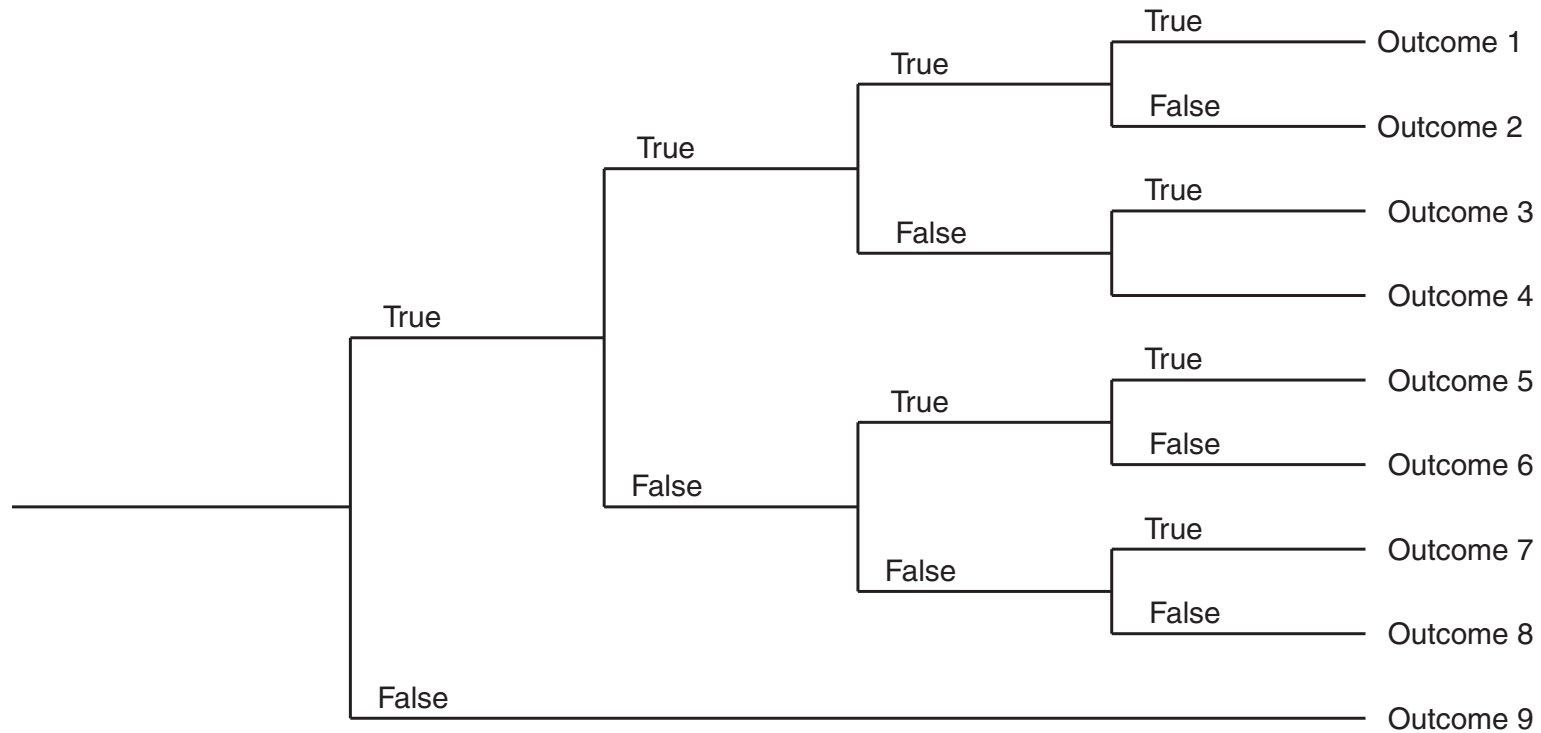
– Source: IEC 61882

Fault Tree Analysis (FTA)



Event Tree Analysis

	B ₁	B ₂	B ₃	B ₄	
Accidental event	Additional event I occurs	Barrier I does not function	Barrier II does not function	Additional event II occurs	Outcome / consequence



Qualitative Vs Quantitative

- ▶ Qualitative risk estimation is based on qualitative enumerations (e.g., minor, negligible, etc.)
- ▶ Quantitative estimation is based on numerical estimation (e.g., $F=10^{-2}$)
- ▶ The Grail : quantitative estimation -> NOT ALWAYS POSSIBLE (e.g., software faults) -> « reasonable worst-case estimate of probability, ... it is convenient to set this default value of the probability to one » (ISO 14971)



Quantitative Risk estimation : Significant disadvantages

- ▶ **The risk estimation depends on users / cared person.**
In case of in-home care, a caretaker or a cared person has to operate a medical / rehabilitation robot by himself / herself. Most of caretakers or cared persons are not familiar with their operation. Then the probability of occurrence of harm becomes large caused by their incorrect operation or misuse and a generic calculation becomes impossible.
 - ▶ **There is little judgment material for determining the probability of occurrence of harm and severity of possible harm :** Compared with machinery, there are few statistics data about the accident report of medical treatment and rehabilitation apparatus. In the present circumstances, these values are estimated experimentally or subjectively by the risk assessor.
-



Risk acceptability criteria

- ▶ Risk acceptance principles from industrial safety

ALARP (UK)

As Low As Reasonably Prac^ticable

reduce unacceptable risks to acceptable level

effort ("price") must be reasonable and practicable

GAMAB (FR)

Globalement Au Moins Aussi Bon (globally at least as good)

allows for trade-offs

similar to ALARP

MEM (DE)

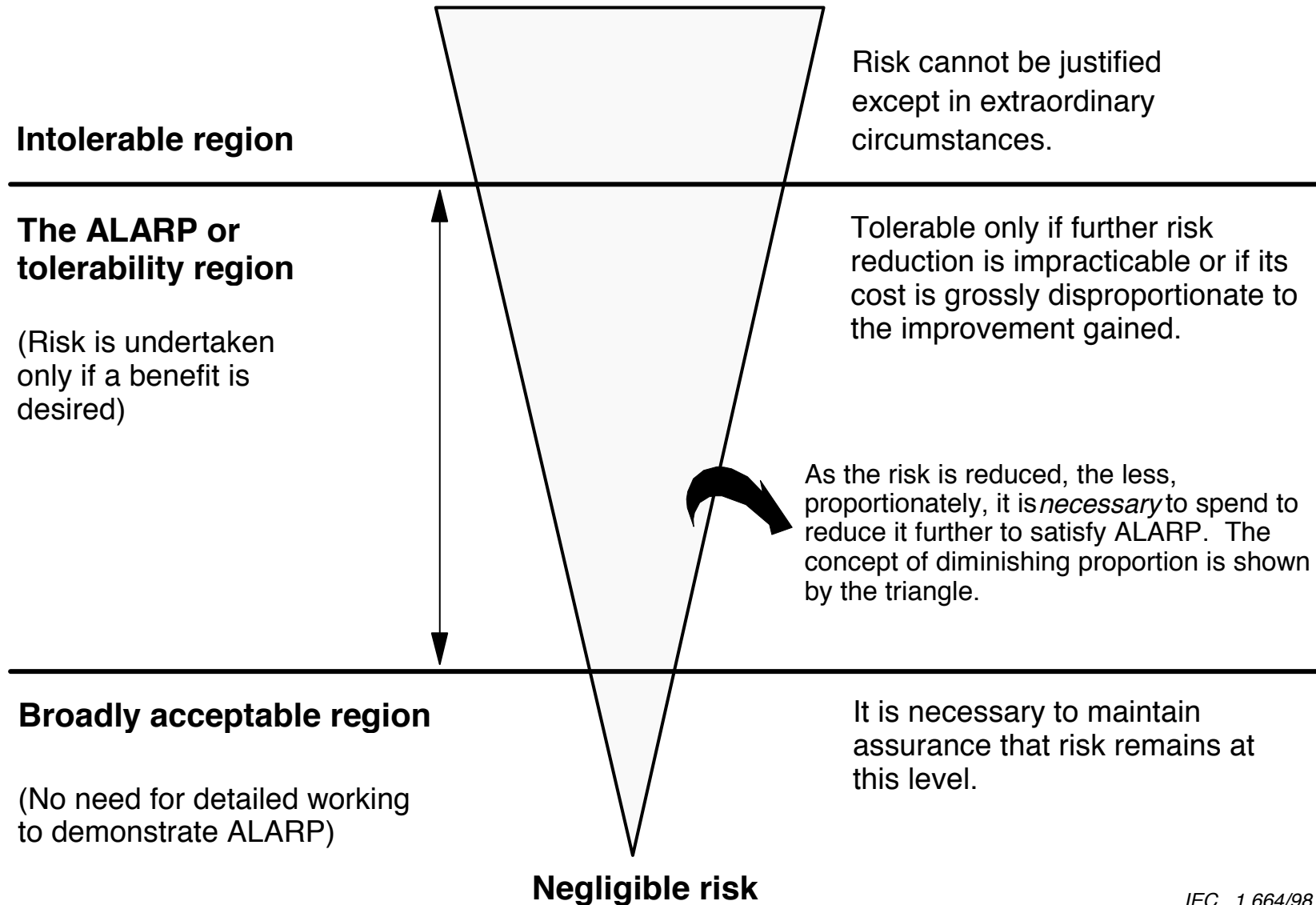
Minimum Endogenous Mortality

objective principle, but risk is subjective

controversial, difficult to apply



Risk acceptability criteria



Risk acceptability criteria

- ▶ **Difficulties to define generic process**
 - ▶ There is no set of generally accepted risk acceptance principles
 - ▶ Harmonisation is necessary for interoperability
 - ▶ difficult, because risk acceptance is a political question
 - ▶ Standards propose principles
 - ▶ leave the details to the legislative bodies

MEM is not generally accepted

GAMAB appears to be most widespread

results are not uniform

ALARP has greatest potential to bring improvements



Tolerable Risk for Robots

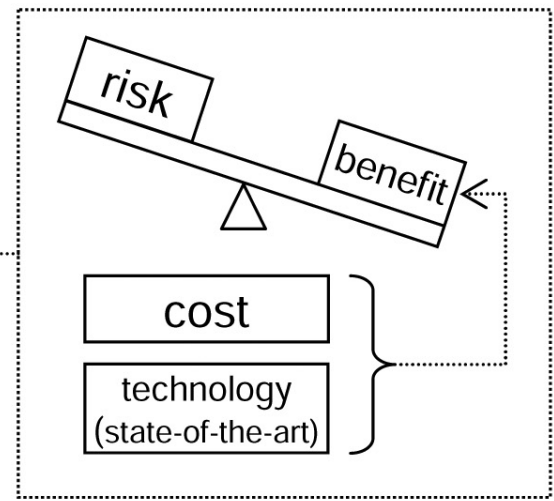
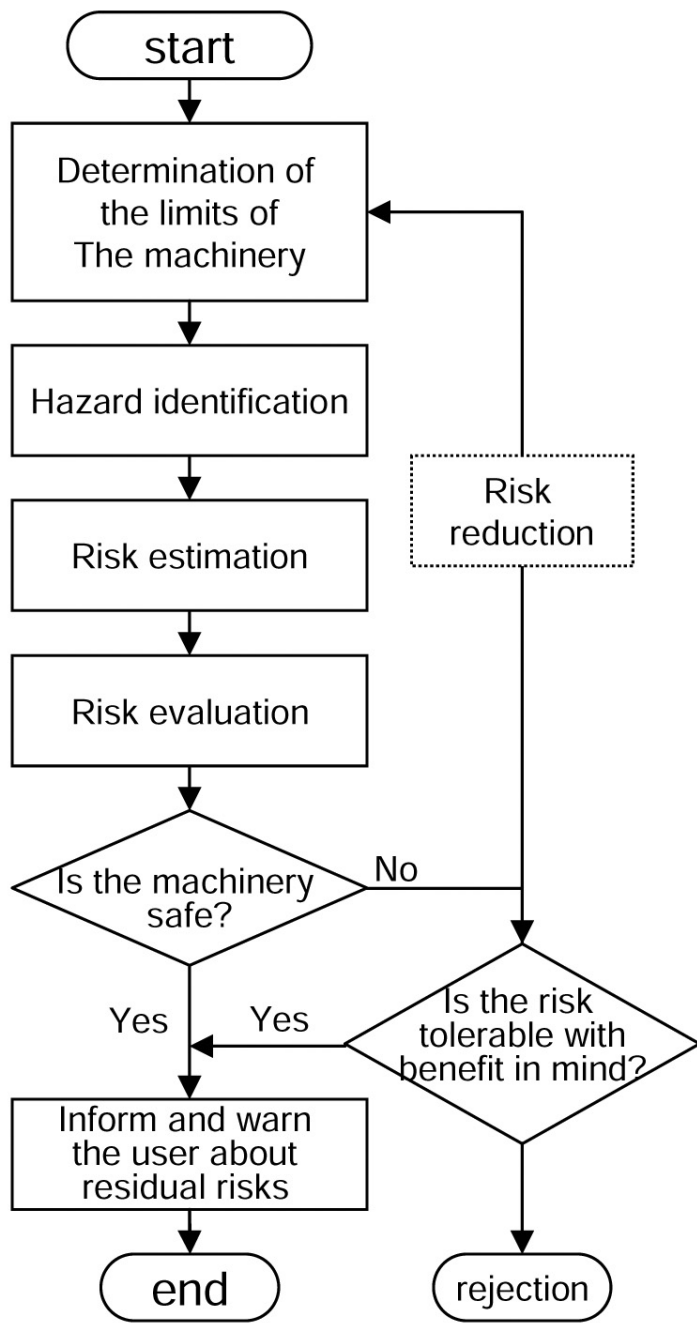
▶ Machinery/Industrial robots

- ▶ Level of tolerable risk should be decided according to the standard which risk assessor declares.
- ▶ Applicable to industrial robots -> workers whom they may injure do not directly benefit from their use, their safety should be certified objectively

▶ Rehabilitation robots

- ▶ Users may accept their use on account of these benefits even when the designer cannot reduce their associated risks sufficiently.
- ▶ For example, surgical robots are highly beneficial if the patients' outcome is successful, however, the operative outcomes are not always a success. the residual risks were clearly detailed to the patients and that the patients consented to their use





ISO 14971 : Application of risk management to medical devices

A contribution : Quantitative Benefit Estimation for rehabilitation robots

- ▶ Most of benefits must be quantified by use of QOL (Quality Of Life), ROL (Respect Of Living) and ADL (Activities of Daily Living), but it is too difficult to quantify them objectively. For examples, benefit of "mobility" is changed according to the extent of gait disorder, so it is necessary to subjectively consider the daily life of targeted cared person.

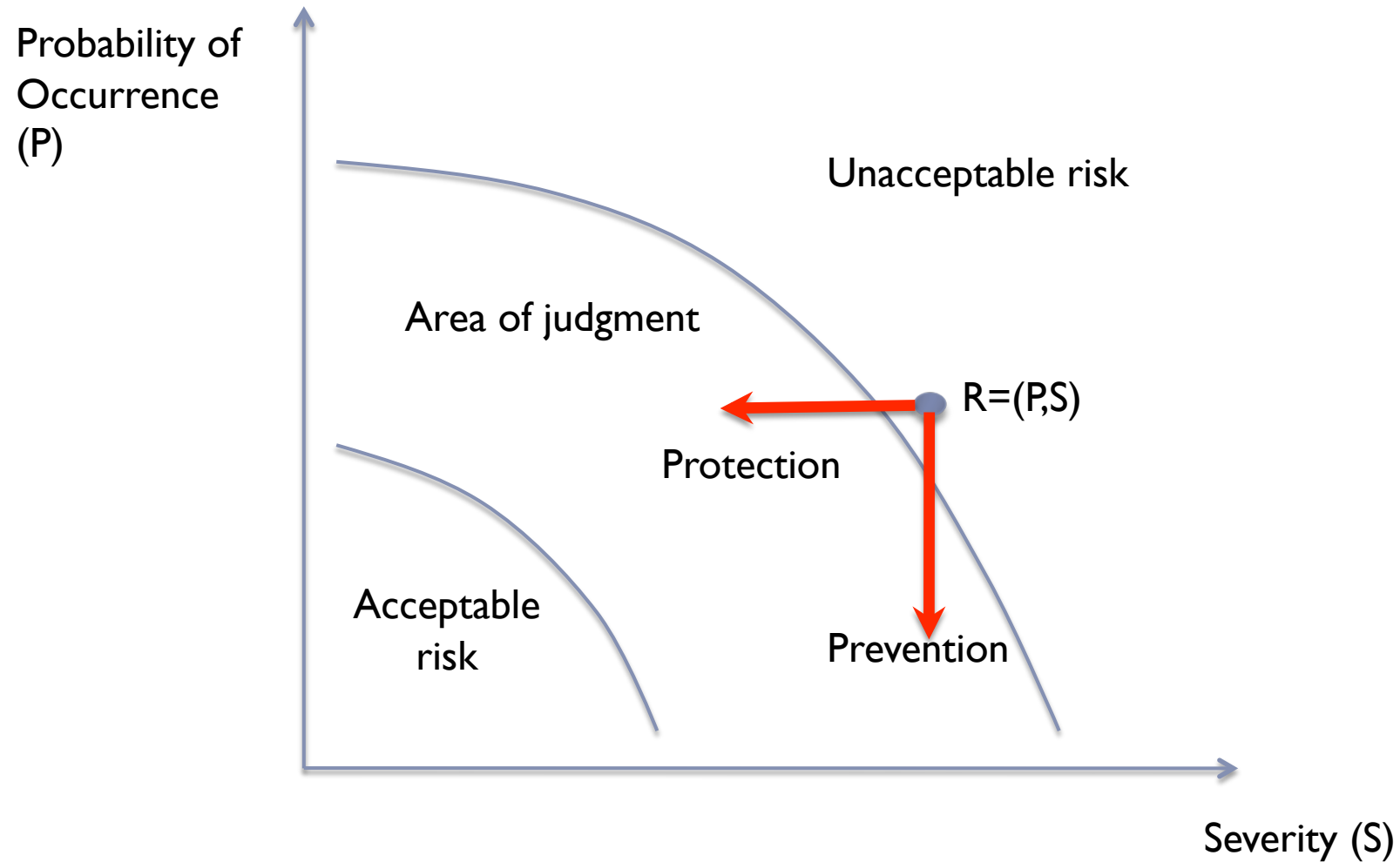
Table 10.1. Benefits of using rehabilitation robots and the quantification factors

	Benefit	Quantification factors
User (Mostly carer)	Improvement of working condition (ex. Reduction of lumbago generating)	QOL, ROL, Tiredness, Working time, Cut-down medical expenses for lumbago
Cared person	Acquisition of an independence life Expansion of a life space Mentally relieved	QOL, ROL, ADL

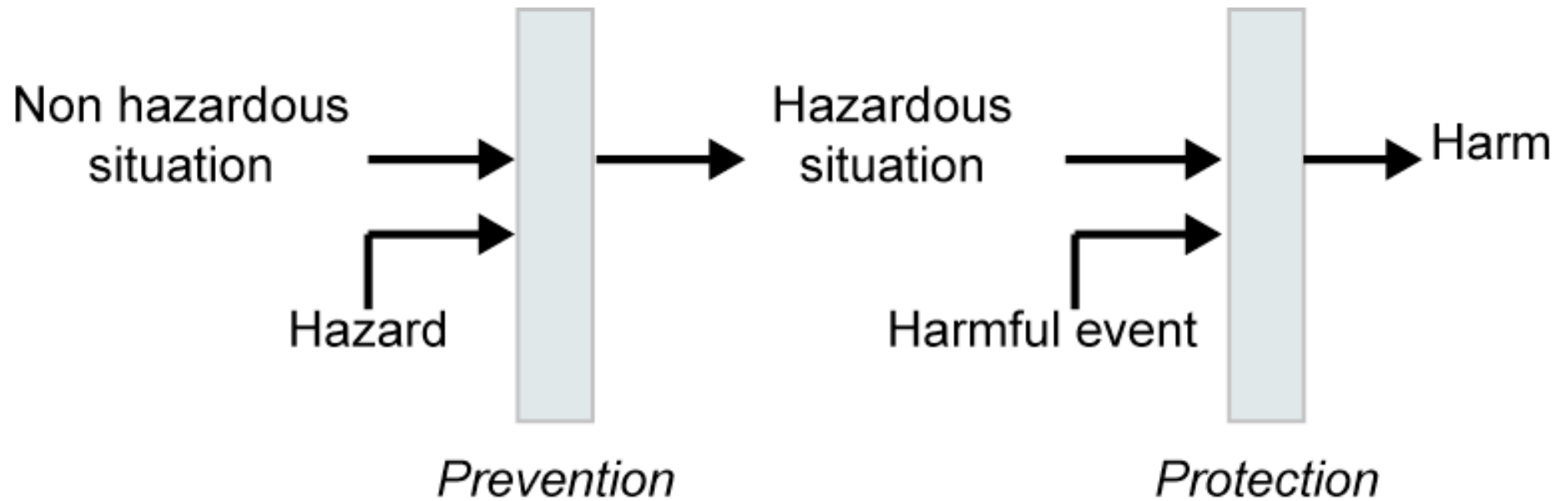
From : *A Safety Strategy for Rehabilitation Robots*, Makoto Nokata and Noriyuki Tejima

Risk reduction technologies

Prevention and protection



Prevention and protection as safety barriers



Risk reduction technologies

- ▶ **Prevention:**
 - ▶ Fault avoidance
 - ▶ Software development fault avoidance (e.g., use of software **fault prevention** and **fault elimination** methods and tools)
 - ▶ Hardware fault avoidance (e.g., preventive maintenance)
 - ▶ Human error avoidance (e.g., human robot interface analysis, cognitive aspects to avoid human error, Human legible motion planning and reactive planning for collision avoidance)
 - ▶ Performance limitations
 - ▶ Mechanical architecture limits (e.g., limits of weight, restriction of degrees of freedom)
 - ▶ Working area, force, acceleration, and speed limits
- ▶ **Protection:**
 - ▶ Compliance of robot movements (Passively safe actuators, control of active compliance)
 - ▶ **Fault tolerance** mechanisms :
 - ▶ Independent safety systems (Safety bag)
 - ▶ Detection and reaction to human presence or contact (Collision detection and reaction)
 - ▶ Detection of robot failure (through redundancy)
 - ▶ Emergency stop and controlled stop controlled by hardware devices (sensors, dead-man switches, etc.)



Example 1 Redundancy

- ▶ A fail-safe dual channel robot control for surgery applications

- ▶ U. Laible et al. / Safety Science 4 (2004) 423–436

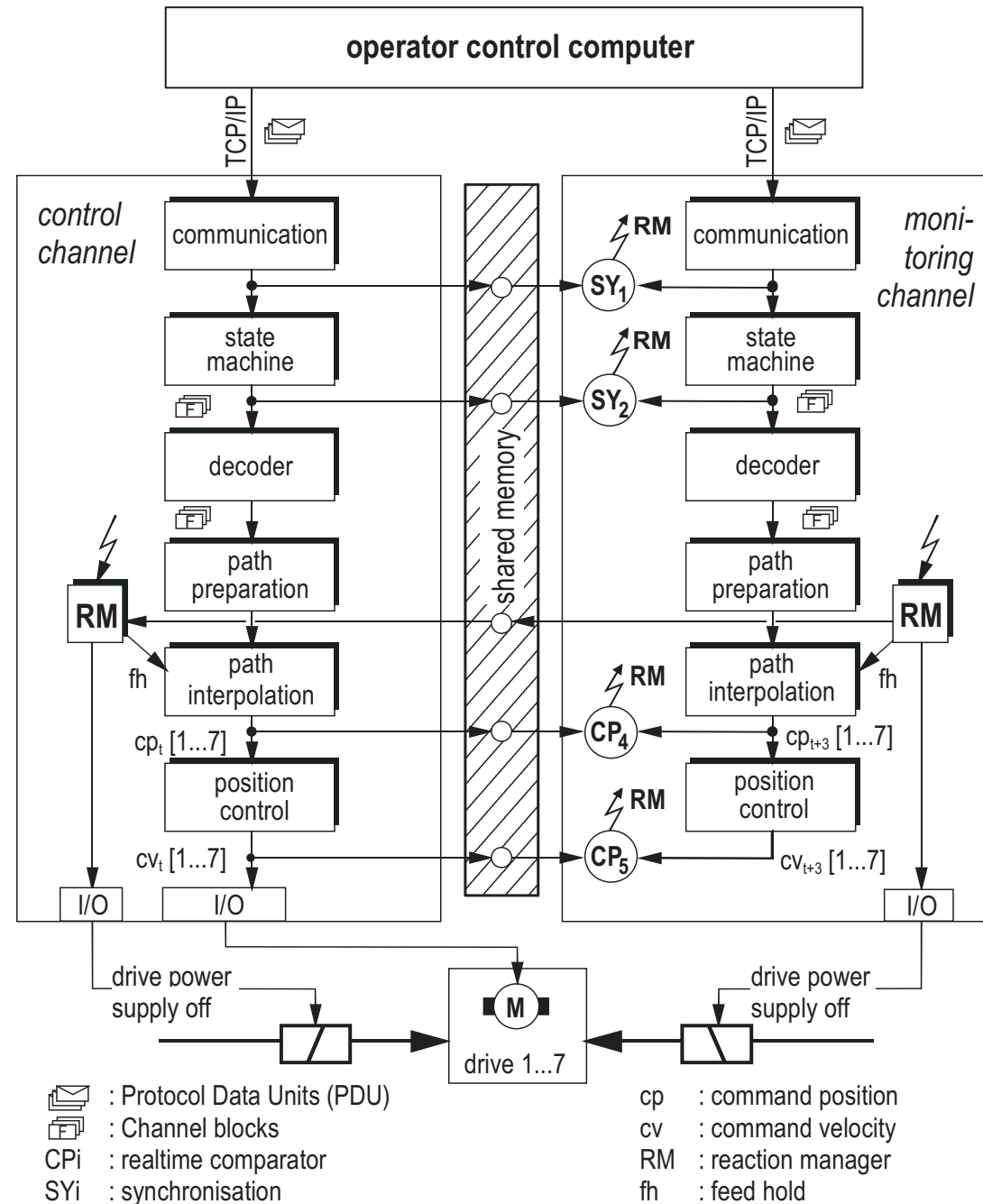
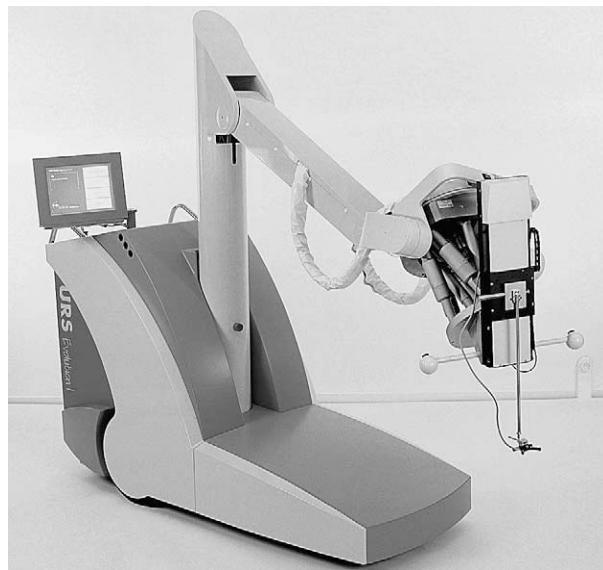
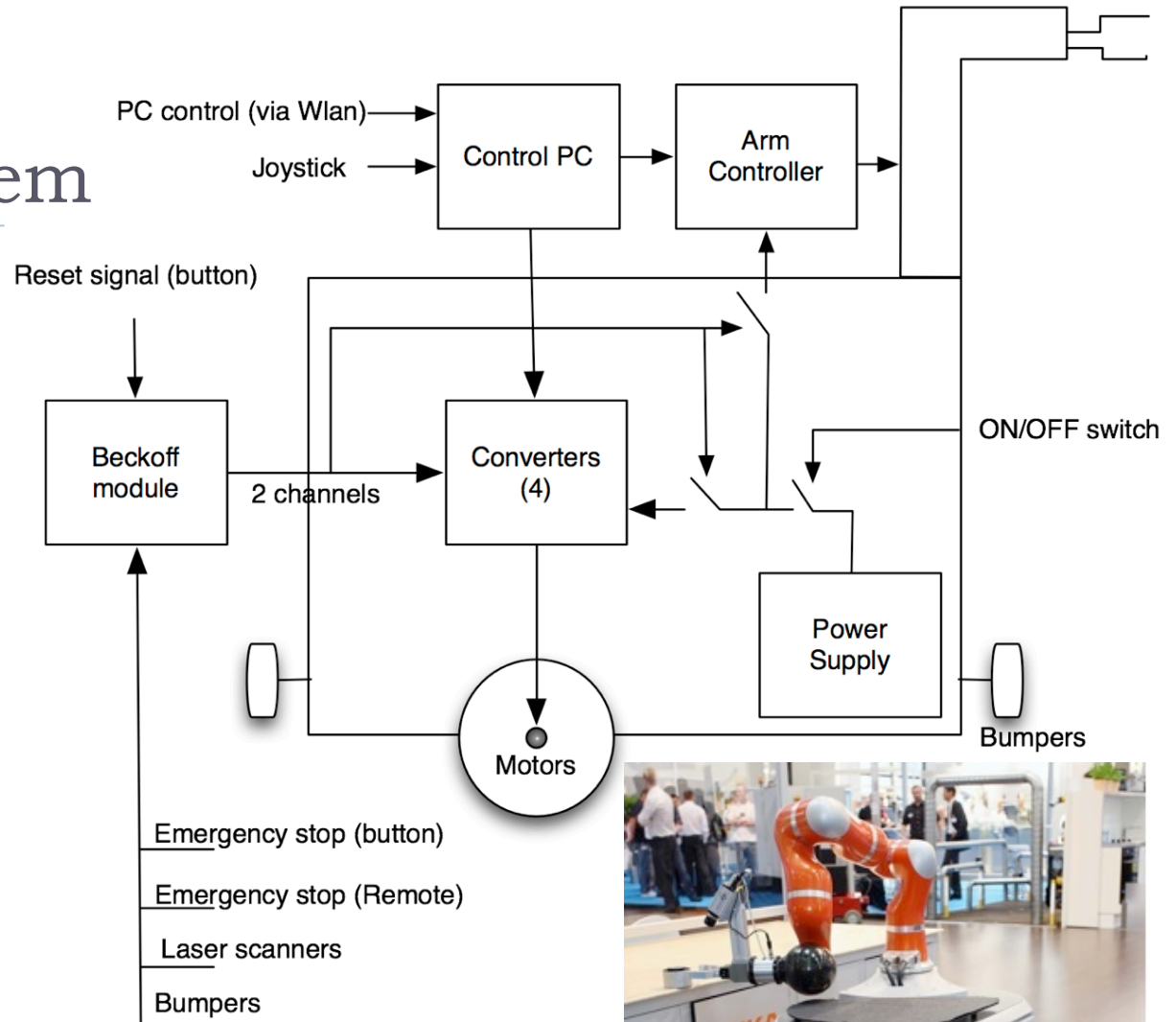


Fig. 7. Redundant RC compares command position values.

Example 2

Protection System

- ▶ Safety PLC (Programmable Logical Controller)
- ▶ Cut power of the robot arm => no power : the brakes are engaged in the robot arm.
- ▶ Command the converters of the motors to slow down.
- ▶ After a delay, the power going towards the motor via the converters is cut => no power on that line : the brakes on the wheel motors are engaged.



KUKA omnirob© concept

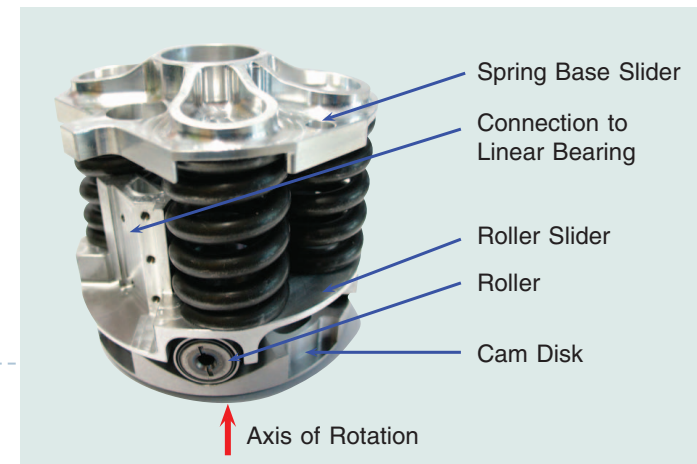
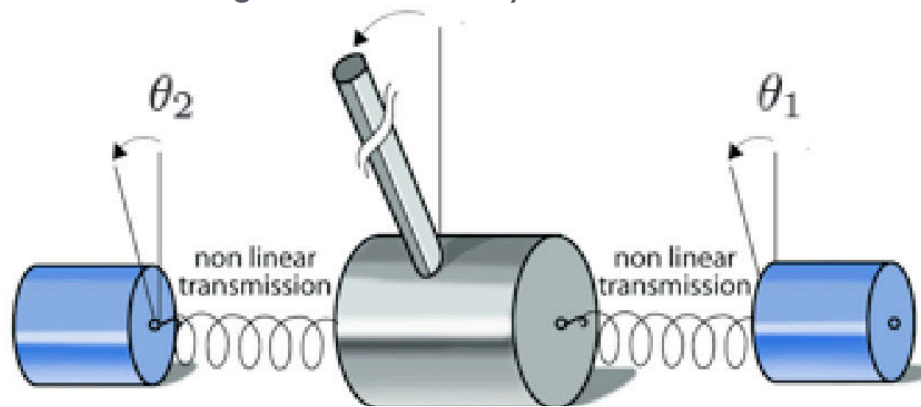
Example 3 – A new generation of actuators

▶ MacKibben Muscles

- ▶ TONDU B., IPPOLITO S., GUIOCHET J., DAIDIE A., "A Seven-degrees-of-freedom Robot-Arm Driven by Pneumatic Artificial Muscles for Humanoid Robots", International Journal of Robotics Research, vol. 24, num. 4, MIT Press, pp. 257-274, April 2005

▶ Variable stiffness actuation :VSA

- ▶ A. De Luca, F. Flacco, R. Schiavi, A. Bicchi, "Nonlinear decoupled motion-stiffness control and collision detection/reaction for the VSA-II variable stiffness device", IEEE/RSJ International Conference on Intelligent Robots and Systems, USA, 2009



Human factors in risk analysis

“To err is human, but to really foul things up requires a computer”

Or a weak system...



Human factors in risk management

- ▶ Human factor studies in robotics usually focus on the design of operator or user interfaces to enhance operator performance and decrease potential human errors => use of guidelines / checklists / best practices
- ▶ No sufficient for innovative system (guidelines not applicable)
- ▶ Important human factors activities for risk management :
 - ▶ Task analysis (and function allocation)
 - ▶ Human error analysis



Task analysis and function allocation

- ▶ **Task analysis** : identify the details of specified tasks, including the knowledge, skills, attitudes, and personal characteristics required for successful task performance.
- ▶ Linked to the process of **function allocation**, which aims to determine the distribution of work between human and technical actors.
- ▶ **Human error** as an hazard should be identified and analyzed as other hazards


All those human factors activities are strongly linked with the two first step of risk analysis



Methods for task analysis & human error analysis

- ▶ Many methods from human factors:
 - ▶ Models of tasks / activities
 - ▶ Stanton, N., P. Salmon, G. Walker, C. Baber, and Daniel P. Jenkins. *Human Factors Methods: A Practical Guide for Engineering and Design*. Ashgate Publishing, 2006
- ▶ Most of those methods for human error analysis are closed to risk analysis methods such as FMECA or HAZOP
- ▶ For task analysis/function allocation : based on models of tasks => see section 5 for an example

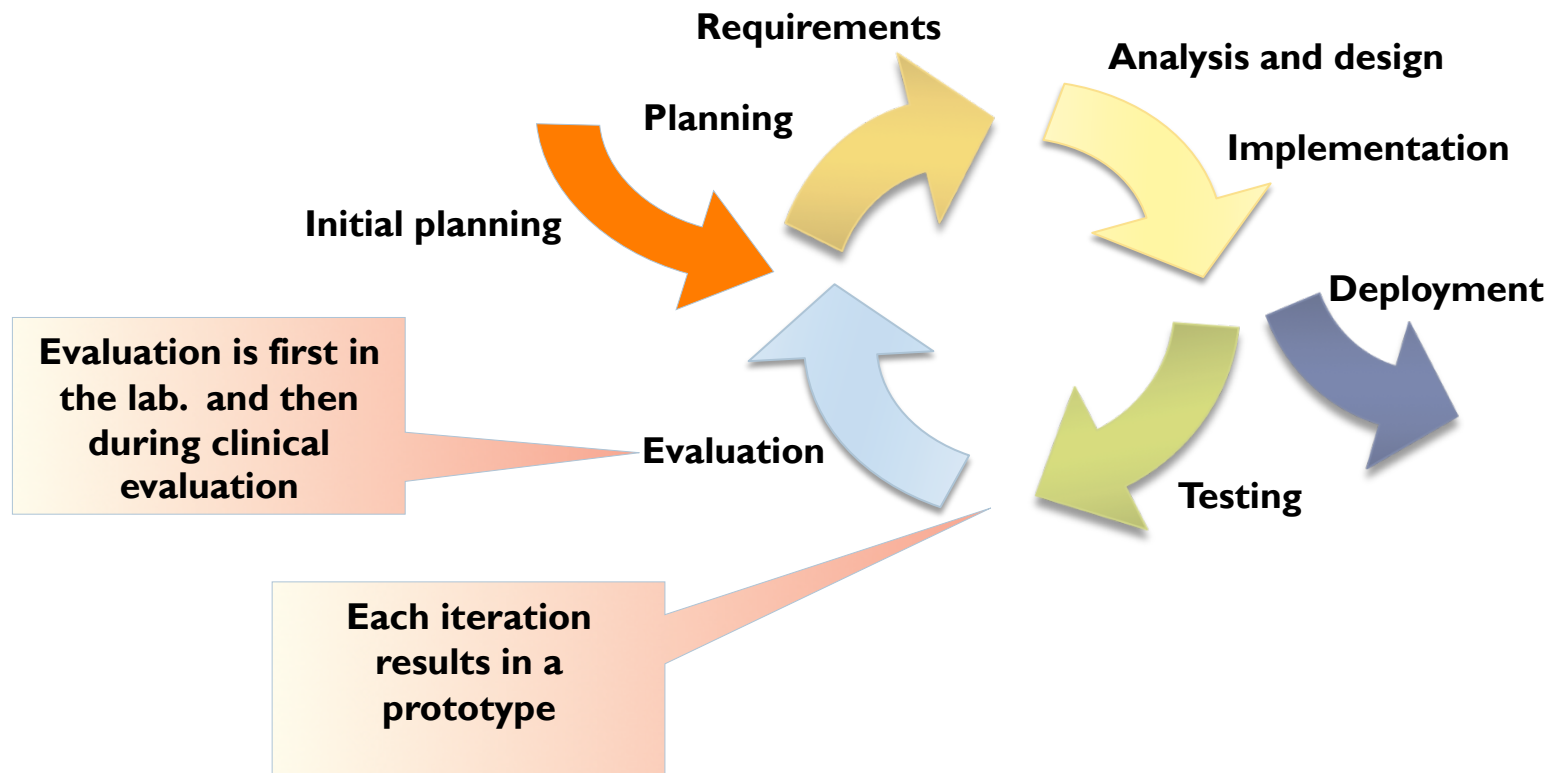




Development process and risk management



Iterative and incremental process for rehabilitation systems



Activities in this process

- ▶ Initial planning- develop a the concept and a vision of the system and produce a plan
- ▶ Requirements - Requirements analysis for an application, such as writing scenarios of use and identifying non-functional requirements.
- ▶ Analysis - Refine the requirements to describe with models **what** the system has to do according to requirements
- ▶ Design - Describe **how** the system performs analysis description (overall architecture, objects, SW and HW choices)
- ▶ Implementation
- ▶ Testing - functional testing but also robustness, reliability, performance, integrity, benchmark, installation, etc.
- ▶ Evaluation – Considering the tests and the requirements, (re) evaluate if objectives are reached
- ▶ Deployment – Deploy the system for final users (install, training course, support, etc.)

Clinical Evaluation process

Activity organization

- ▶ Activities in a project should be organised to produce tangible outputs for management to judge progress.
- ▶ *Milestones* are the end-point of a process activity.
- ▶ *Deliverables* are project results delivered to customers.
- ▶ The waterfall process allows for the straightforward definition of progress milestones.
- ▶ Risk management is one activity of the overall process
 - ▶ Necessity to define milestones
 - ▶ And deliverables

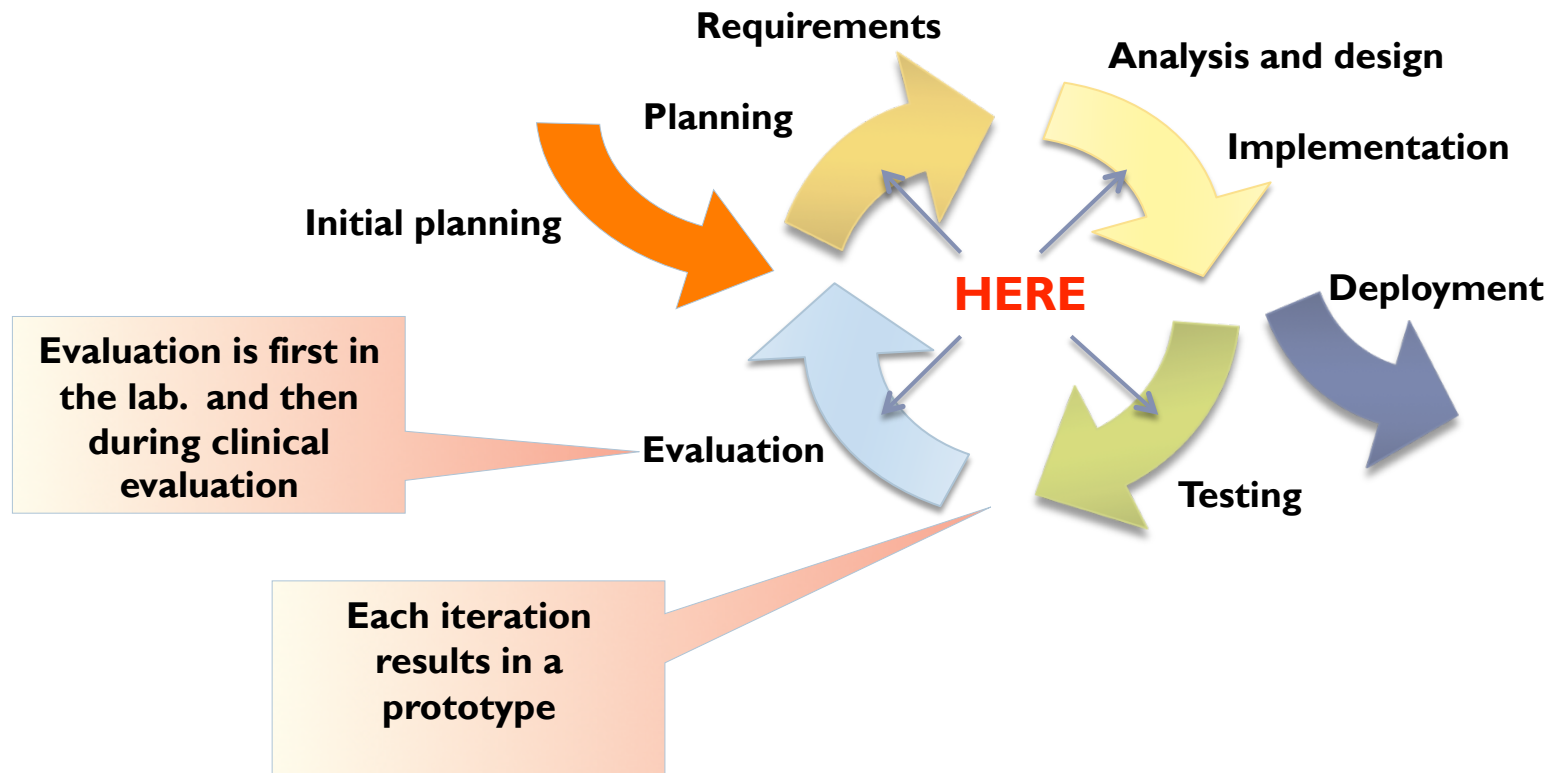


When manage risk ?

- ▶ Risk management is an iterative and incremental process
 - ▶ Iterative :
 - ▶ At the very beginning of the development process (initial planning) => risk management plan and preliminary hazard identification
 - ▶ Number of iterations depends on the project, the team, the objectives => adapt it to the project
 - ▶ Incremental
 - ▶ Studies are refined and level of details increase
 - ▶ Hazard : combinatory explosion of the risk analysis results => decide which level of granularity
 - ▶ Outputs of risk management:
 - ▶ Modification of use, specification, and design
 - ▶ => results must be inputs of the development process, nevertheless in many project safety analysis are performed after the design !



So when manage risk ? (2)



Chapter 4 – Three risk analysis techniques HAZOP, FMECA, and FTA

Hazard Operability (HAZOP)

Credits : Marvin Rausand

What is HAZOP?

- ▶ A Hazard and Operability (HAZOP) study is a structured and systematic examination of a planned or existing process or operation in order to identify and evaluate problems that may represent risks to personnel or equipment, or prevent efficient operation.
- ▶ The HAZOP technique was initially developed to analyze chemical process systems, but has later been extended to other types of systems and also to complex operations and to software systems.
- ▶ A HAZOP is a qualitative technique based on guide-words and is carried out by a multi-disciplinary team (HAZOP team) during a set of meetings.



When to perform a HAZOP?

- ▶ The HAZOP study should preferably be carried out as early in the development phase as possible - to have influence on the design. On the other hand; to carry out a HAZOP we need a rather complete description of the system. As a compromise, the HAZOP is also sometimes carried out as a final check when the detailed design has been completed.
- ▶ A HAZOP study may also be conducted on an existing facility to identify modifications that should be implemented to reduce risk and operability problems.



When to perform a HAZOP? - (2)

- ▶ HAZOP studies may also be used more extensively, including:
 - ▶ At the initial concept stage when design drawings are available
 - ▶ When the analysis models are available
 - ▶ During implementation and deployment to ensure that recommendations are implemented
 - ▶ During test and evaluation
 - ▶ During operation to ensure that emergency and operating procedures are regularly reviewed and updated as required



HAZOP background

- ▶ The basis for HAZOP was laid by ICI in 1963 and was based on so-called “critical examination” techniques
- ▶ First guide: “A Guide to Hazard and Operability Studies”, ICI and Chemical Industries Associations Ltd. 1977.
- ▶ First main textbook: Kletz, T.A.: “Hazop and Hazan - Identifying and Assessing Process Industry Hazards” , Institution of Chemical Engineers.
- ▶ See also: Kletz, T.A.: “Hazop – past and future”. Reliability Engineering and System Safety, 55:263-266, 1997.



Standards and guidelines

- ▶ IEC 61882. “Hazard and operability studies (HAZOP studies) – Application guide” . International Electrotechnical Commission, Geneva.
- ▶ Crawley, F., M. Preston, and B. Tyler: “HAZOP: Guide to best practice. Guidelines to best practice for the process and chemical industries” . European Process Safety Centre and Institution of Chemical Engineers, 2000
- ▶ Kyriakdis, I.: “HAZOP - Comprehensive Guide to HAZOP in CSIRO” , CSIRO Minerals, National Safety Council of Australia, 2003



Types of HAZOP

- ▶ **Process HAZOP**
 - ▶ The HAZOP technique was originally developed to assess plants and process systems
- ▶ **Human HAZOP**
 - ▶ A “family” of specialized HAZOPs. More focused on human errors than technical failures
- ▶ **Procedure HAZOP**
 - ▶ Review of procedures or operational sequences Sometimes denoted SAFOP - SAFE Operation Study
- ▶ **Software HAZOP**
 - ▶ Identification of possible errors in the development of software



HAZOP team and meetings

- ▶ **HAZOP team leader Responsibilities:**
 - ▶ Define the scope for the analysis
 - ▶ Select HAZOP team members
 - ▶ Plan and prepare the study
 - ▶ Chair the HAZOP meetings
 - ▶ → Trigger the discussion using guide-words and parameters
 - ▶ → Follow up progress according to schedule/agenda
 - ▶ → Ensure completeness of the analysis
- ▶ The team leader should be independent (i.e., no responsibility for the process and/or the performance of operations)



Team members and responsibilities (2)

- ▶ **HAZOP secretary Responsibilities:**
 - ▶ Prepare HAZOP worksheets
 - ▶ Record the discussion in the HAZOP meetings
 - ▶ Prepare draft report(s)



Team members

▶ HAZOP team members

The basic team for a process plant will be:

- ▶ Project engineer
- ▶ Commissioning manager
- ▶ Process engineer
- ▶ Instrument/electrical engineer
- ▶ Safety engineer

Depending on the actual process the team may be enhanced by:

- ▶ Operating team leader
 - ▶ Maintenance engineer
 - ▶ Suppliers representative
 - ▶ Other specialists as appropriate
-



How to be a good HAZOP participant?

- ▶ Be active! Everybody's contribution is important
- ▶ Be to the point. Avoid endless discussion of details
- ▶ Be critical in a positive way - not negative, but constructive
- ▶ Be responsible. He who knows should let the others know



HAZOP meeting

- ▶ **Proposed agenda:**
 - ▶ 1. Introduction and presentation of participants
 - ▶ 2. Overall presentation of the system/operation to be analyzed
 - ▶ 3. Description of the HAZOP approach
 - ▶ 4. Presentation of the first node or logical part of the operation
 - ▶ 5. Analyze the first node/part using the guide-words and parameters
 - ▶ 6. Continue presentation and analysis (steps 4 and 5)
 - ▶ 7. Coarse summary of findings
- ▶ **Focus should be on potential hazards as well as potential operational problems**
- ▶ **Each session of the HAZOP meeting should not exceed two hours.**



HAZOP recording

- ▶ The findings are recorded during the meeting(s) using a HAZOP work-sheet, either by filling in paper copies, or by using a computer connected to a projector (recommended).
- ▶ The HAZOP work-sheets may be different depending on the scope of the study - generally the following entries (columns) are included:
 - ▶ 1. Ref. no.
 - ▶ 2. Guide-word
 - ▶ 3. Deviation
 - ▶ 4. Possible causes
 - ▶ 5. Consequences
 - ▶ 6. Safeguards
 - ▶ 7. Actions required (or, recommendations)
 - ▶ 8. Actions allocated to (follow-up responsibility)

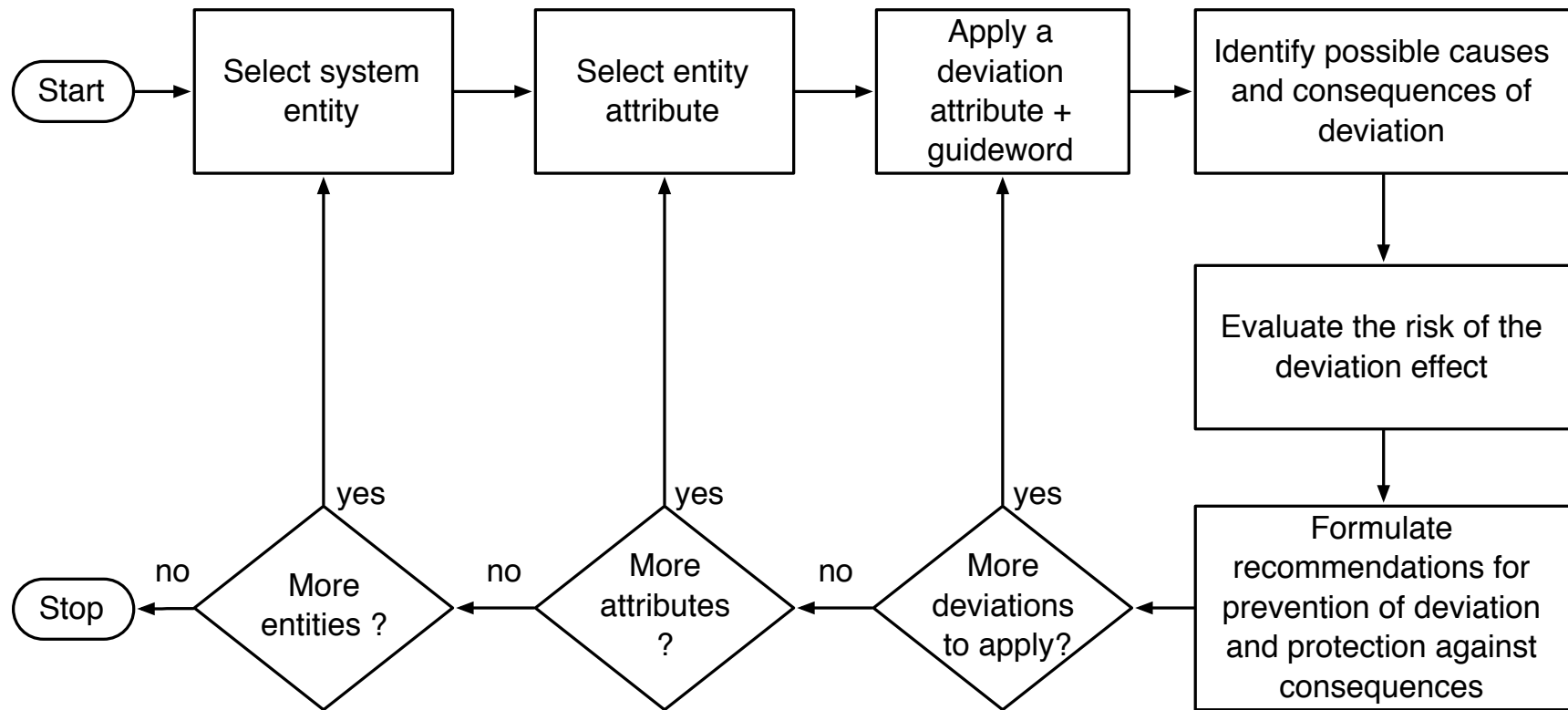


HAZOP procedure

1. Divide the system into entities (i.e., reactor, storage)
2. Choose an entity and an attribute (i.e., line, vessel, pump, operating instruction)
3. Apply a guide-word
4. Determine cause(s)
5. Evaluate consequences/problems
6. Recommend action: What? When? Who?
7. Record information
8. Repeat procedure (from step 2)



HAZOP procedure (2)



Example of HAZOP worksheet

Study title:						Page: of			
Drawing no.:			Rev no.:			Date:			
HAZOP team:						Meeting date:			
Part considered:									
Design intent:			Material: Source:			Activity: Destination:			
No.	Guide-word	Element	Deviation	Possible causes	Consequences	Safeguards	Comments	Actions required	Action allocated to

– Source: IEC 61882



Worksheet entries (1)

▶ Modes of operation

The following modes of system operation should be considered for each entity:

- ▶ Normal operation
- ▶ Reduced throughput operation
- ▶ Routine start-up
- ▶ Routine shutdown
- ▶ Emergency shutdown
- ▶ Special operating modes (e.g. fall back modes)



Worksheet entries (2)

- ▶ **Scenario of use**

- ▶ operation/activity of the system and humans are described

- ▶ **Deviation**

- ▶ A deviation is a way in which the operation conditions may depart from their design intent.

- ▶ **Parameter**

- ▶ The relevant parameter for the condition(s) of the operation (e.g. pressure, speed, acceleration, movements).



Worksheet entries - (3)

▶ **Guidewords**

- ▶ A short word to create the imagination of a deviation of the design/process intent. The most commonly used set of guidewords is: no, more, less, as well as, part of, other than, and reverse. In addition, guidewords like too early, too late, instead of, are used; the latter mainly for batch-like processes. The guidewords are applied, in turn, to all the parameters, in order to identify unexpected and yet credible deviations from the design/process intent.

Guide-word + Parameter → Deviation



Basic HAZOP guide-words

No/None	Complete negation of the design intention No part of the intention is achieved and nothing else happens
More	Quantitative increase
Less	Quantitative decrease
As Well As	All the design intention is achieved together with additions
Part of	Only some of the design intention is achieved
Reverse	The logical opposite of the design intention is achieved
Other than	Complete substitution, where no part of the original intention is achieved but something quite different happens
Early	Something happens earlier than expected relative to clock time
Late	Something happens later than expected relative to clock time
Before	Something happens before it is expected, relating to order or sequence
After	After Something happens after it is expected, relating to order or sequence



Worksheet entries - (4)

▶ Cause

- ▶ The reason(s) why the deviation could occur. Several causes may be identified for one deviation. It is often recommended to start with the causes that may result in the worst possible consequence.

▶ Consequence

- ▶ The results of the deviation, in case it occurs. Consequences may both comprise process hazards and operability problems, like plant shut-down or reduced quality of the product. Several consequences may follow from one cause and, in turn, one consequence can have several causes



Worksheet entries - (5)

▶ Safeguard

- ▶ Facilities that help to reduce the occurrence frequency of the deviation or to mitigate its consequences. Some types of safeguards are:
 - ▶ 1. Detect the deviation (e.g., with sensors, use of alarms)
 - ▶ 2. Compensate for the deviation (e.g., an automatic control)
 - ▶ 3. Prevent the deviation from occurring
 - ▶ 4. Prevent further escalation of the deviation (e.g., by (total) trip of the activity. These facilities are often interlocked with several units in the process, often controlled by computers)



Review meetings

- ▶ Review meetings should be arranged to monitor completion of agreed actions that have been recorded. The review meeting should involve the whole HAZOP team. A summary of actions should be noted and classified as:
 - ▶ Action is complete
 - ▶ Action is in progress
 - ▶ Action is incomplete, awaiting further information



HAZOP Results

- ▶ **Improvement of system or operations**
 - ▶ Reduced risk and better contingency
 - ▶ More efficient operations
- ▶ **Improvement of procedures**
 - ▶ Logical order
 - ▶ Completeness General awareness among involved parties Team building



Advantages

- ▶ Systematic examination
- ▶ Multidisciplinary study
- ▶ Utilizes operational experience
- ▶ Covers safety as well as operational aspects
- ▶ Solutions to the problems identified may be indicated
 - Considers operational procedures
- ▶ Covers human errors
- ▶ Study led by independent person
- ▶ Results are recorded



Success factors

- ▶ Accuracy of drawings and data used as a basis for the study
- ▶ Experience and skills of the HAZOP team leader
- ▶ Technical skills and insights of the team
- ▶ Ability of the team to use the HAZOP approach as an aid to identify deviations, causes, and consequences
- ▶ Ability of the team to maintain a sense of proportion, especially when assessing the severity of the potential consequences.



Pitfalls and objections

- ▶ Time consuming
- ▶ Focusing too much on solutions
- ▶ Team members allowed to divert into endless discussions of details
- ▶ A few of the team members dominate the discussion
 - “This is my design/procedure”
 - ▶ – Defending a design/procedure
 - ▶ – HAZOP is not an audit
- ▶ “No problem”
- ▶ “Wasted time”



Failure Modes Effects and Criticality Analysis (FMECA)

Credits : Marvin Rausand

[FMECA Slides](#)

Fault Tree Analysis (FTA)

Credits : Marvin Rausand

[FTA slides](#)

What is fault tree analysis?

- ❑ Fault tree analysis (FTA) is a top-down approach to failure analysis, starting with a potential undesirable event (accident) called a TOP event, and then determining all the ways it can happen.
- ❑ The analysis proceeds by determining how the TOP event can be caused by individual or combined lower level failures or events.
- ❑ The causes of the TOP event are “connected” through logic gates
- ❑ In this book we only consider AND-gates and OR-gates
- ❑ FTA is the most commonly used technique for causal analysis in risk and reliability studies.



History

- ❑ FTA was first used by Bell Telephone Laboratories in connection with the safety analysis of the Minuteman missile launch control system in 1962
- ❑ Technique improved by Boeing Company
- ❑ Extensively used and extended during the Reactor safety study (WASH 1400)



FTA main steps

- ❑ Definition of the system, the TOP event (the potential accident), and the boundary conditions
- ❑ Construction of the fault tree
- ❑ Identification of the minimal cut sets
- ❑ Qualitative analysis of the fault tree
- ❑ Quantitative analysis of the fault tree
- ❑ Reporting of results

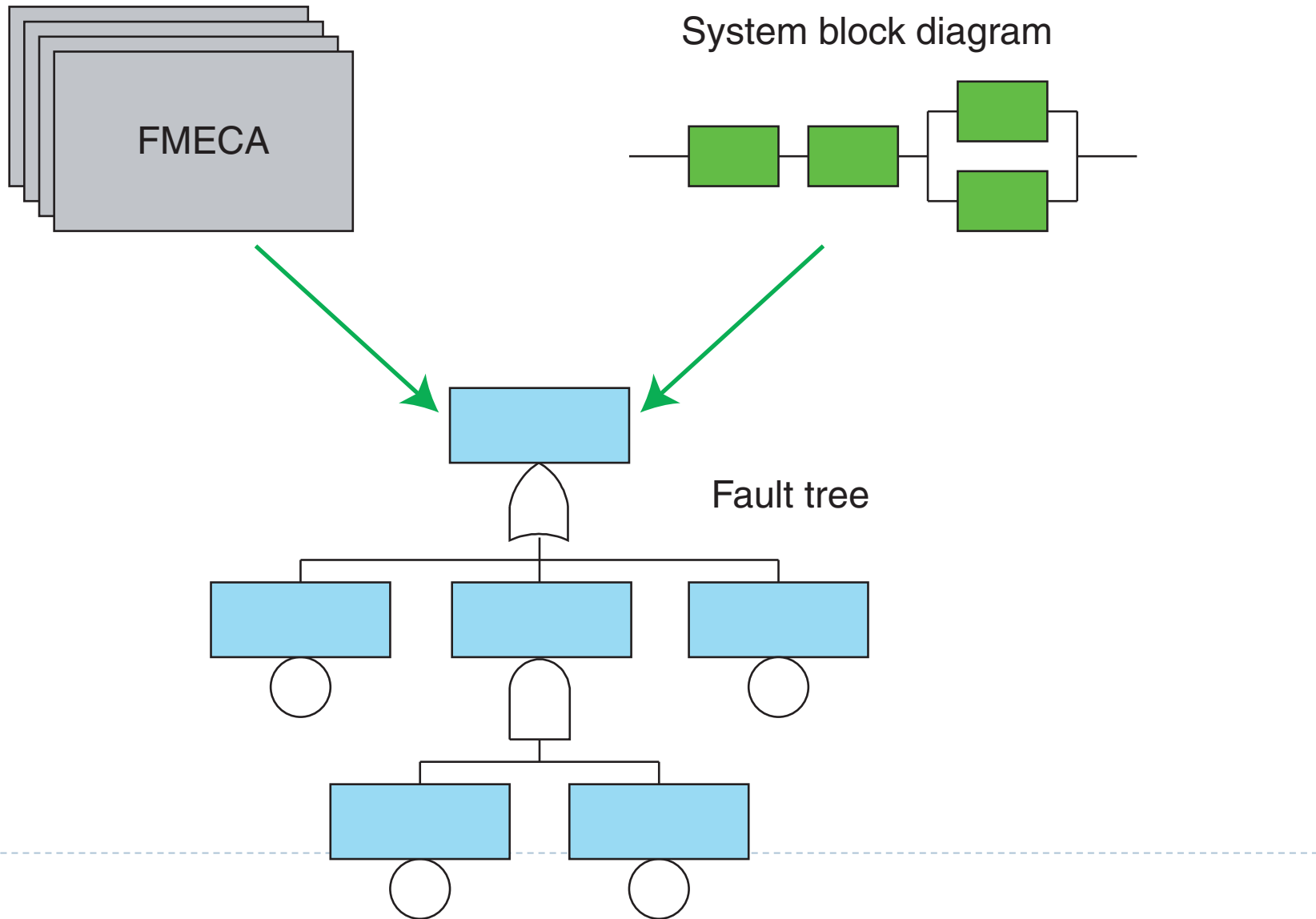


Preparation for FTA

- ❑ The starting point of an FTA is often an existing FMECA and a system block diagram
- ❑ The FMECA is an essential first step in understanding the system
- ❑ The design, operation, and environment of the system must be evaluated
- ❑ The cause and effect relationships leading to the TOP event must be identified and understood



Preparation for FTA



Boundary conditions

- ❑ The physical boundaries of the system (Which parts of the system are included in the analysis, and which parts are not?)
- ❑ The initial conditions (What is the operational stat of the system when the TOP event is occurring?)
- ❑ Boundary conditions with respect to external stresses (What type of external stresses should be included in the analysis – war, sabotage, earthquake, lightning, etc?)
- ❑ The level of resolution (How detailed should the analysis be?)

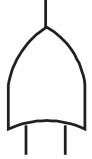
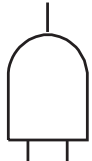


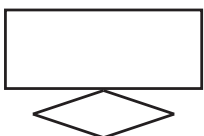
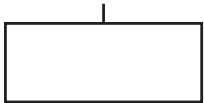




Fault tree construction

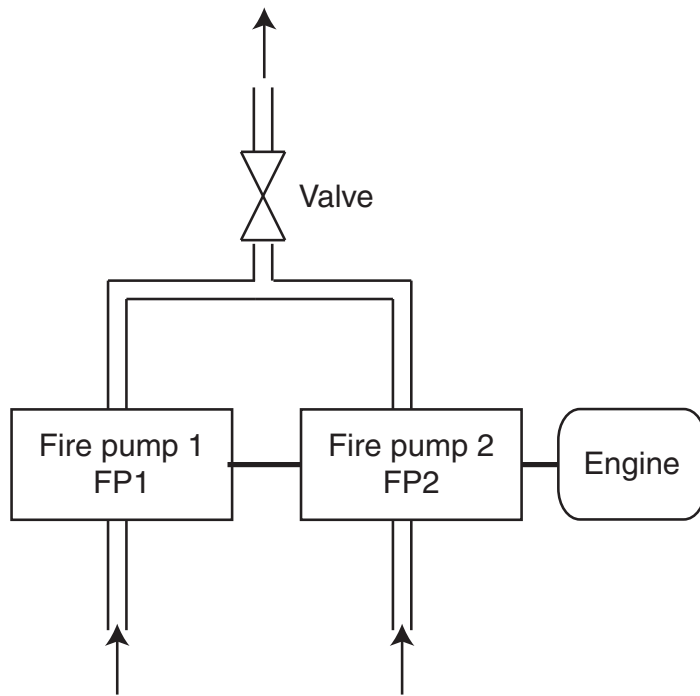
- ❑ Define the TOP event in a clear and unambiguous way.
Should always answer:
 - What** e.g., “Fire”
 - Where** e.g., “in the process oxidation reactor”
 - When** e.g., “during normal operation”
- ❑ What are the immediate, necessary, and sufficient events and conditions causing the TOP event?
- ❑ Connect via AND- or OR-gate
- ❑ Proceed in this way to an appropriate level (= basic events)
- ❑ Appropriate level:
 - ◆ Independent basic events
 - ◆ Events for which we have failure data



Fault tree symbols

Logic gates	 OR-gate	The OR-gate indicates that the output event occurs if any of the input events occur
	 AND-gate	The AND-gate indicates that the output event occurs only if all the input events occur at the same time
Input events (states)	 	The basic event represents a basic equipment failure that requires no further development of failure causes
		The undeveloped event represents an event that is not examined further because information is unavailable or because its consequences are insignificant
Description of state		The comment rectangle is for supplementary information
Transfer symbols	Transfer out  Transfer in 	The transfer-out symbol indicates that the fault tree is developed further at the occurrence of the corresponding transfer-in symbol

Example: Redundant fire pumps



TOP event = No water from fire water system

Causes for TOP event:

VF = Valve failure

G1 = No output from any of the fire pumps

G2 = No water from FP1 G3 = No water from FP2

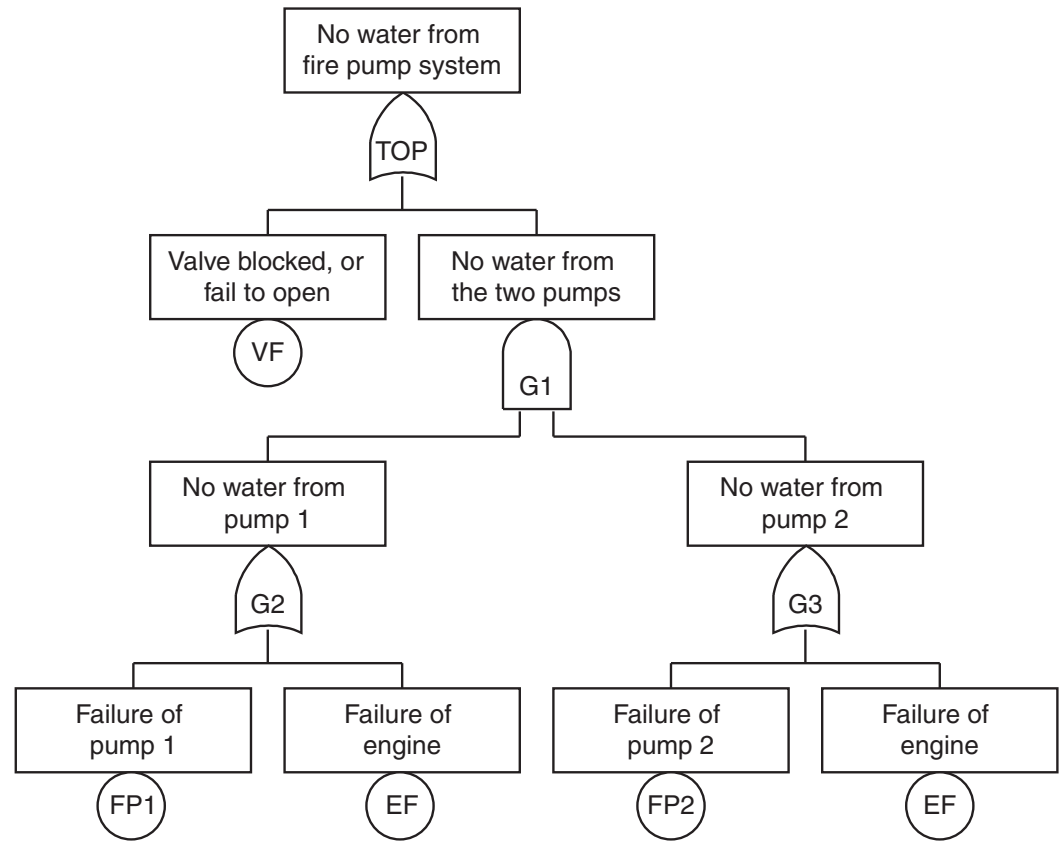
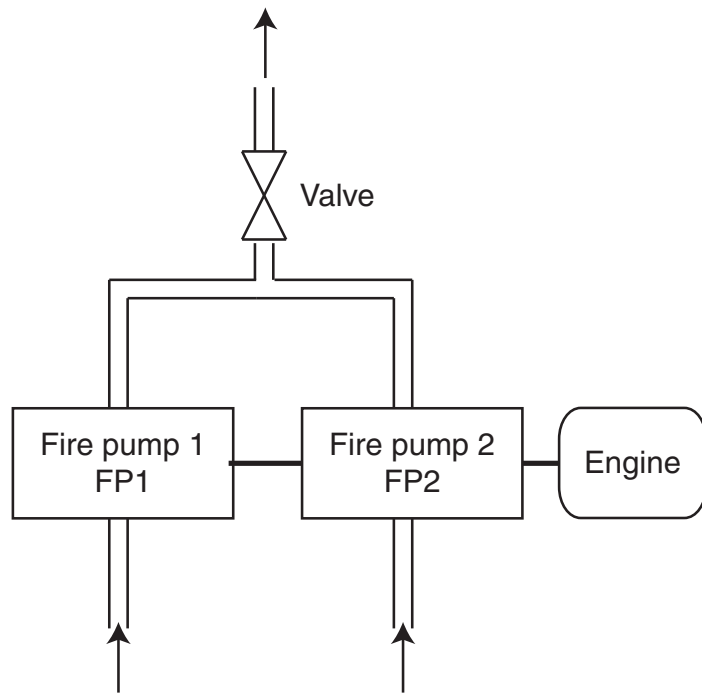
FP1 = failure of FP1

EF = Failure of engine

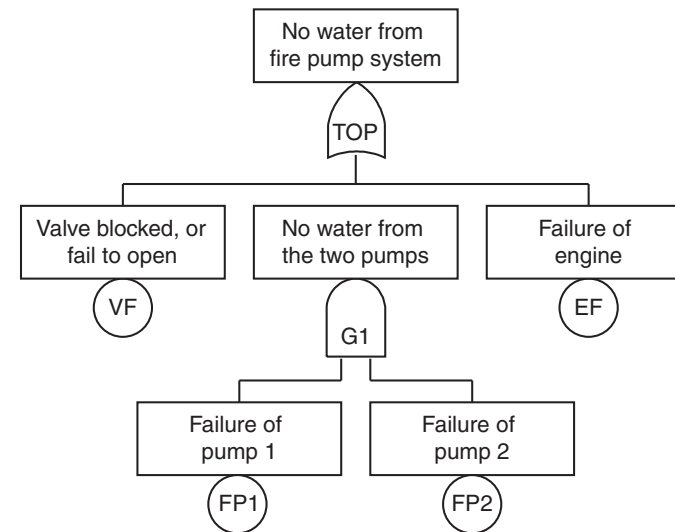
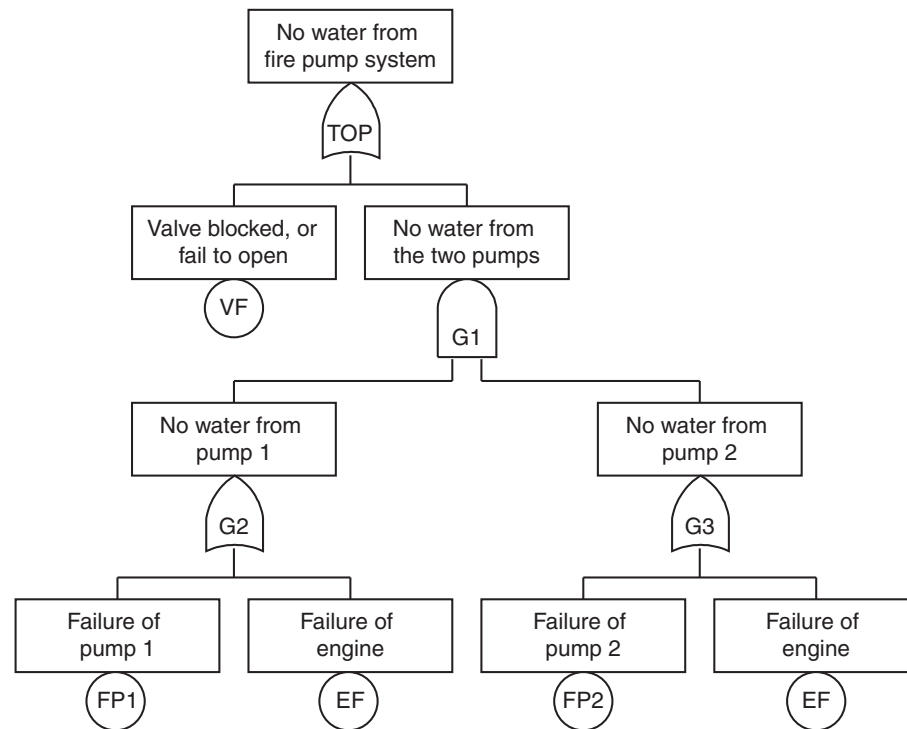
FP2 = Failure of FP2



Example: Redundant fire pumps (2)



Example: Redundant fire pumps (3)



The two fault trees above are logically identical. They give the same information.



Conclusions

- ❑ FTA identifies all the possible causes of a specified undesired event (TOP event)
- ❑ FTA is a structured top-down deductive analysis.
- ❑ FTA leads to improved understanding of system characteristics. Design flaws and insufficient operational and maintenance procedures may be revealed and corrected during the fault tree construction.
- ❑ FTA is not (fully) suitable for modelling dynamic scenarios
- ❑ FTA is binary (fail–success) and may therefore fail to address some problems



Quantitative estimation with independent stochastic events

AND gate Event E occurs when E_1 and E_1 and ... and E_n occurs

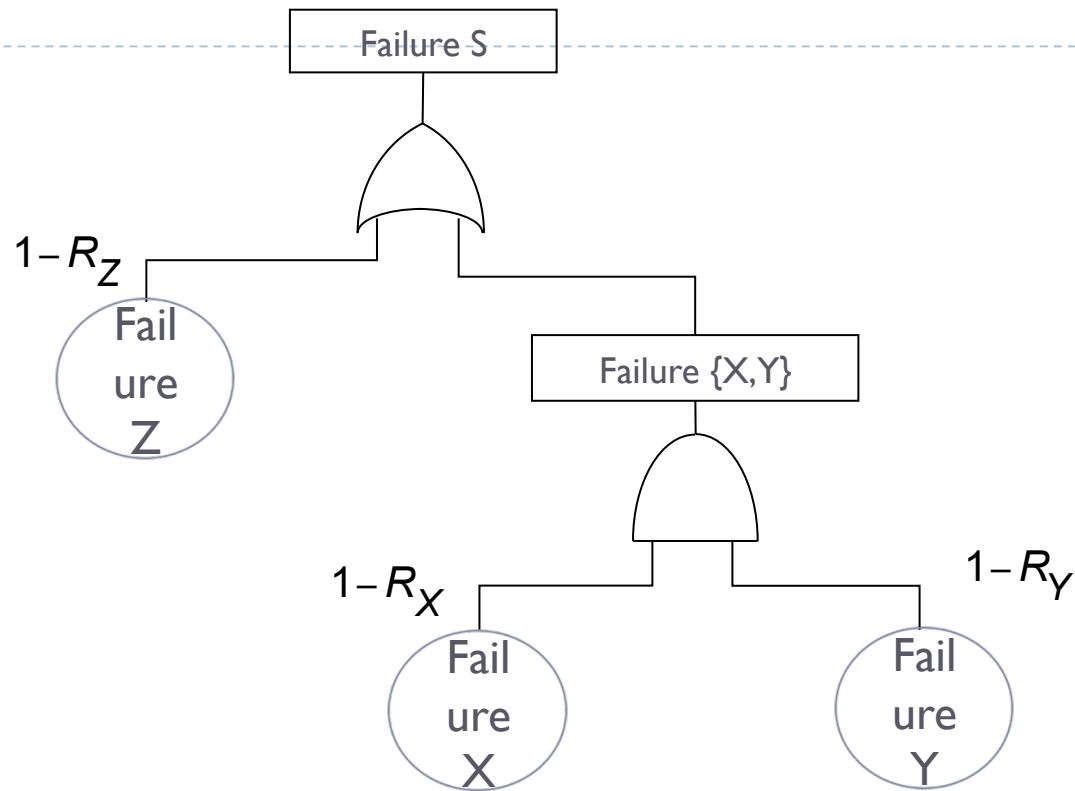
$$E = E_1 \cap E_2 \cap \dots \cap E_n$$
$$P\{E\} = P\{E_1\} \cdot P\{E_2\} \cdot \dots \cdot P\{E_n\}$$

OR gate Event E occurs when E_1 or E_1 or ... or E_n occurs

$$E = E_1 \cup E_2 \cup \dots \cup E_n$$
$$\bar{E} = \bar{E}_1 \cap \bar{E}_2 \cap \dots \cap \bar{E}_n$$
$$P\{E\} = 1 - P\{\bar{E}\} = 1 - (1 - P\{E_1\})(1 - P\{E_1\}) \dots (1 - P\{E_n\})$$

Example for 2 events :

$$E = E_1 \cup E_2$$
$$P\{E\} = P\{E_1\} + P\{E_2\} - P\{E_1\} \cdot P\{E_2\}$$



$$1-R = 1-R_Z + (1-R_X)(1-R_Y) - (1-R_Z)(1-R_X)(1-R_Y)$$

$$R = R_Z(R_X + R_Y - R_X R_Y)$$

Minimal cut sets

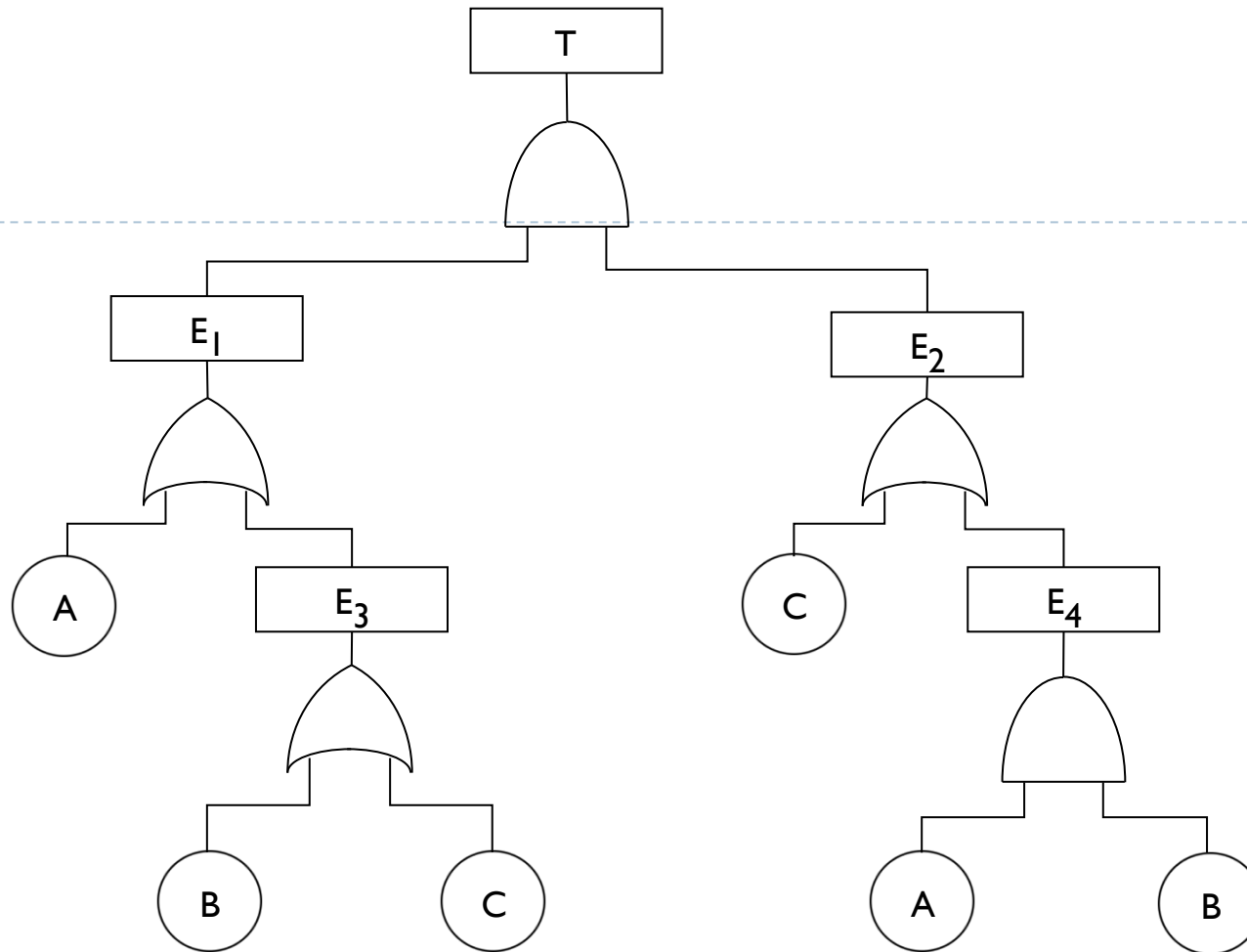
- ▶ A cut set in a fault tree is a set of basic events whose (simultaneous) occurrence ensures that the TOP event occurs
- ▶ A cut set is said to be minimal if the set cannot be reduced without losing its status as a cut set

C_j Minimal cut of order m_j $C_j = B_{1j} \cap B_{2j} \cap \dots \cap B_{jj} \cap \dots \cap B_{m_jj}$

B_{jj} : basic events

R : top event

$$P\{R\} = P\{C_1 \cup C_2 \cup \dots \cup C_m\}$$



$$E_3 = B \cup C$$

$$E_1 = A \cup (B \cup C)$$

$$E_4 = A \cap B$$

$$E_2 = C \cup (A \cap B)$$

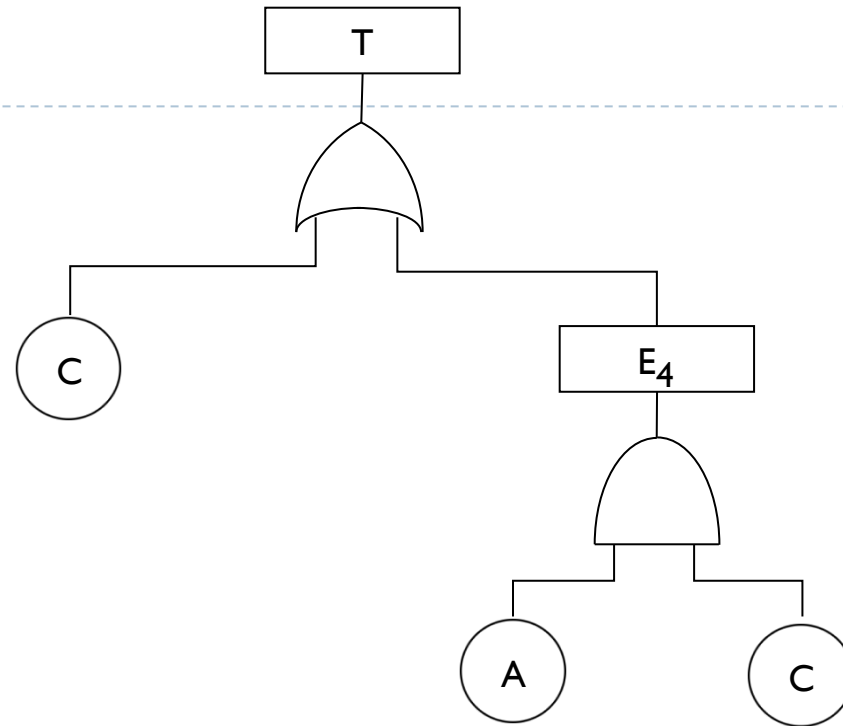
$$T = E_1 \cap E_2 = (A \cup B \cup C) \cap (C \cup (A \cap B))$$

$$T = ((A \cup B \cup C) \cap C) \cup ((A \cup B \cup C) \cap (A \cap B))$$

$$T = (A \cap C) \cup (B \cap C) \cup C \cup (A \cap B) \cup (A \cap B) \cup (A \cap B \cap C)$$

$$T = C \cup (A \cap B)$$

Minimal cut sets : $\{C\}, \{A, B\}$



$$P\{T\} = P\{C \cup (A \cap B)\} = P\{C\} + P\{A\}P\{B\} - P\{A\}P\{B\}P\{C\}$$

From qualitative to quantitative

- ▶ Fault forecasting : use of mathematical tools for calculation of reliability and availability
- ▶ Statistics and probabilities

Chapter 5. A scenario based risk analysis approach

Using UML and HAZOP



From system modelling to UML



System complexity

- ▶ Domain problematic (medical, rehabilitation, transportation, etc.)
- ▶ Development process
- ▶ Software adaptability and modifiability



Complexity outcomes

- ▶ Catastrophic failure probability is high
- ▶ Tuning/adjustment is slow and chaotic
- ▶ Maintainability is out of proportion
- ▶ Cost is high
- ▶ Software crisis (1970)



Complexity management

- ▶ For lack of reducing complexity, one must control it :
 - ▶ Give an illusion of simplicity
 - ➔ modelling
 - ▶ Apply decomposition criteria
 - ➔ break into component parts



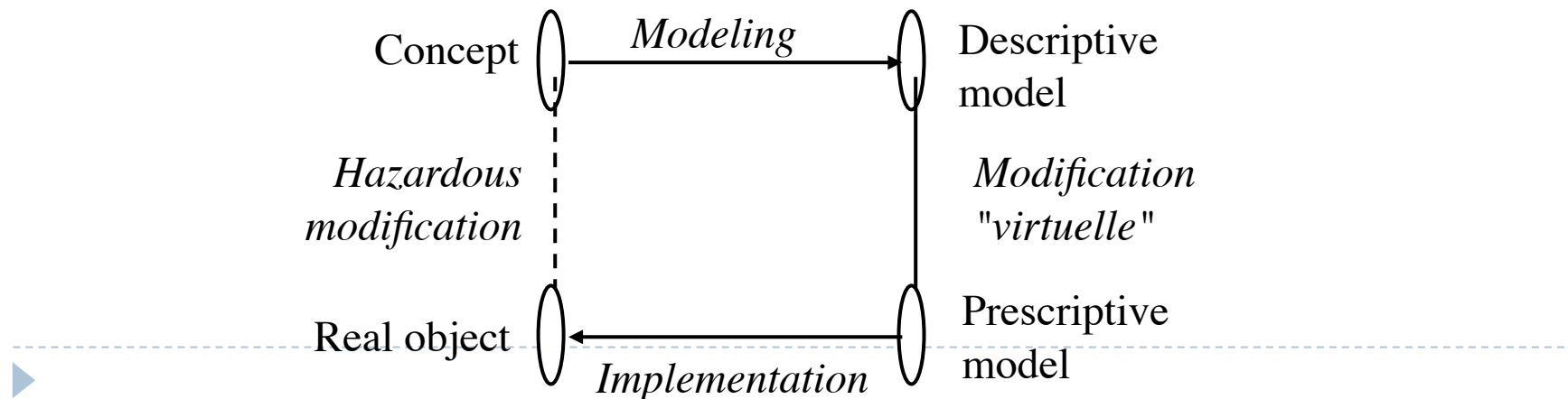
What is a model ?

- ▶ System development → need for concepts manipulation (software, hardware, environment, users, etc.)
- ▶ **Model and modelling**
 - ▶ Could represent something that already exists but also something that does not exist (physical parts of the robot or software entities)
 - ▶ Is an abstraction of the original object of study (a formula or a box is an abstraction)
 - ▶ Only some aspects are considered (e.g. kinematics, thermodynamics, etc.)
 - ▶ Has an objective. It is only considering the objective that the efficiency of the model can be evaluate.

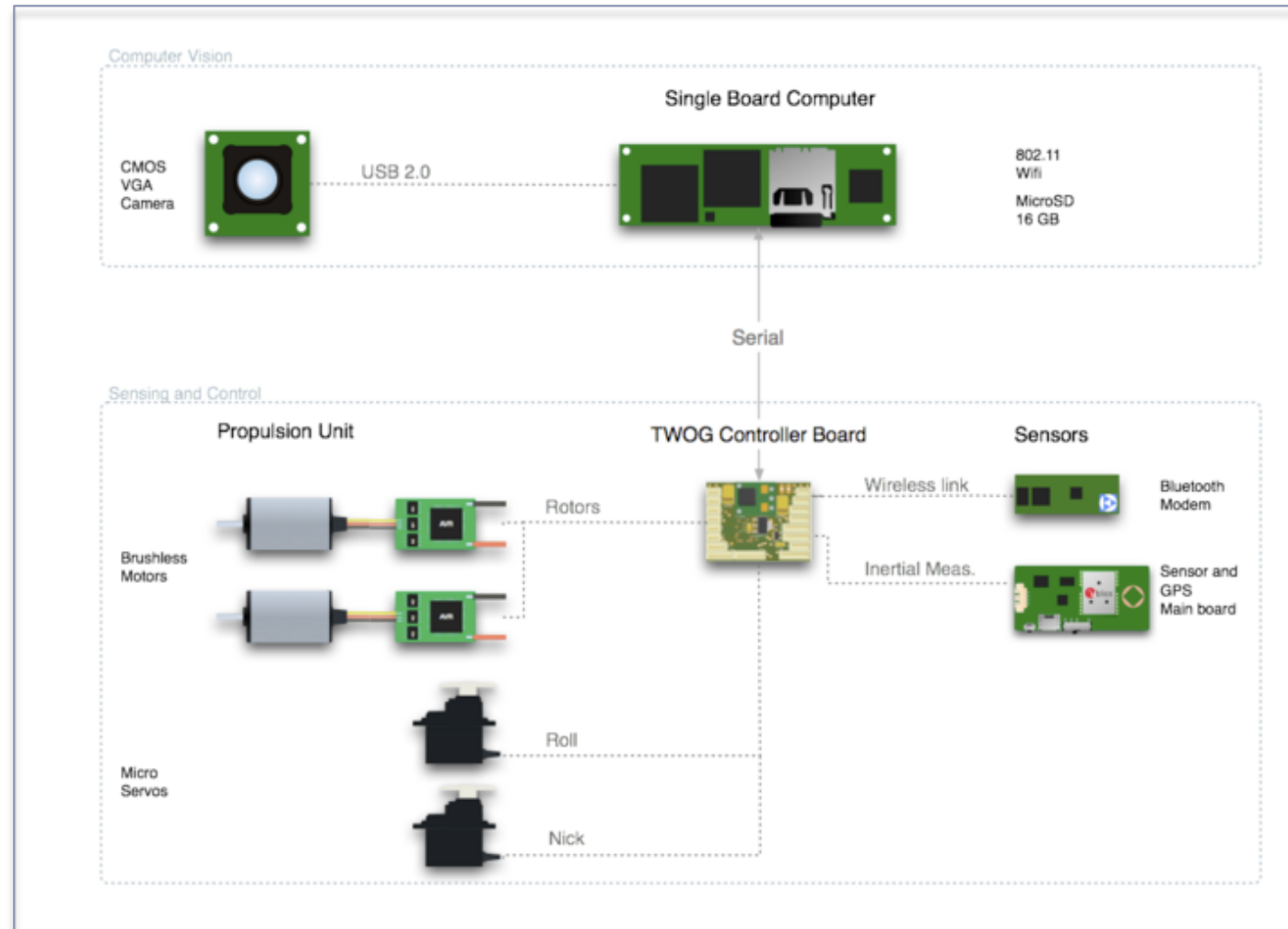


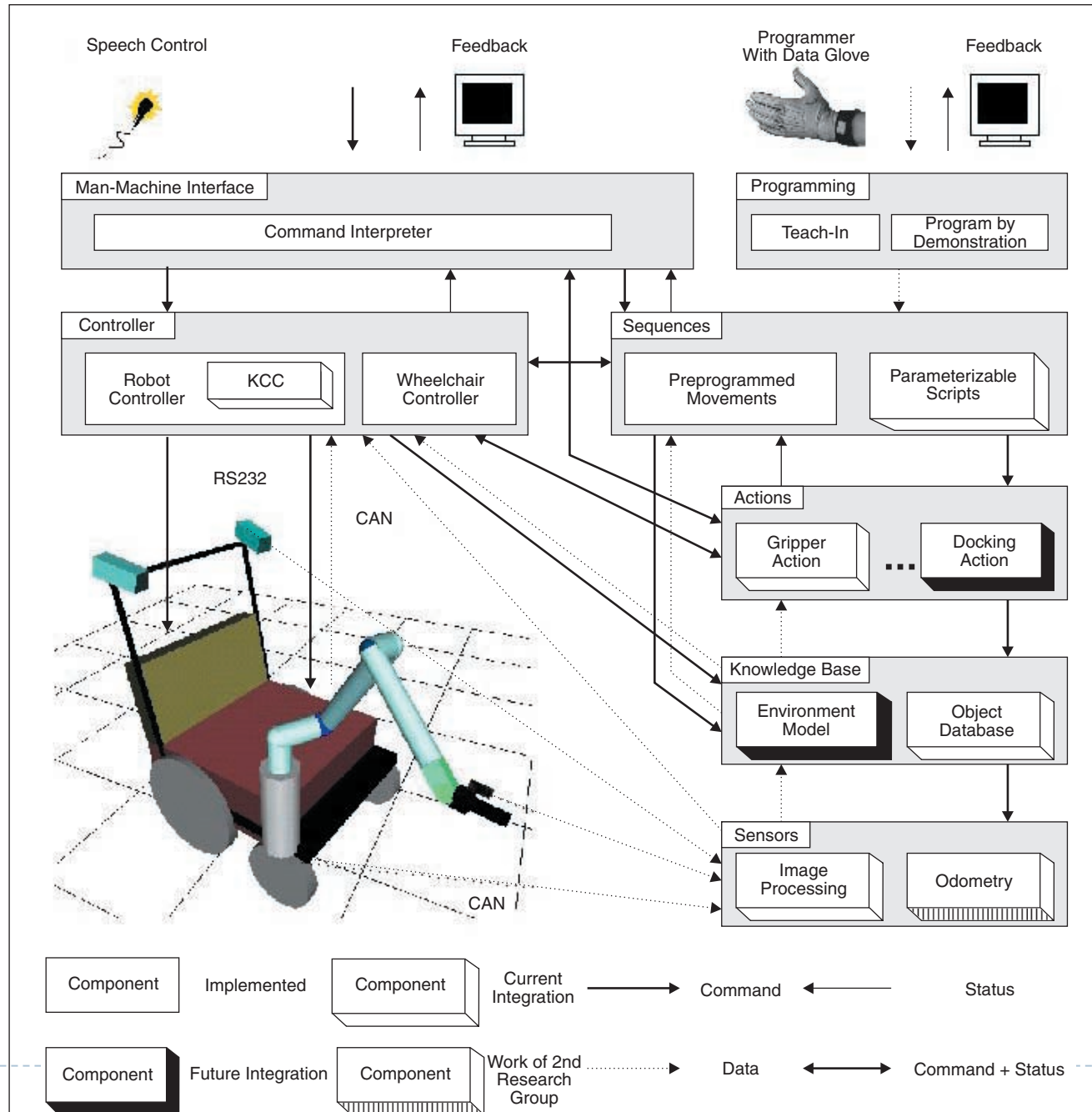
Why use a model?

- ▶ A model is used when:
 - ▶ Reality is too complex (simplification)
 - ▶ A concept is required (abstraction)
 - ▶ Direct modification of the design is too hazardous (representation)
 - ▶ Communicate between developers
 - ▶ Prevent and eliminate errors of specification/design
 - ▶ Guarantee tracability from requirements to implementation



Models for system development





Models

- ▶ List all modeling language that you know:
 - ▶ Modeling of the dynamics
 - ▶ Structural modelling



Division role

- ▶ « Divide and rule »
- ▶ Recursive refinement until reach comprehensive elements
- ▶ Divide system state space

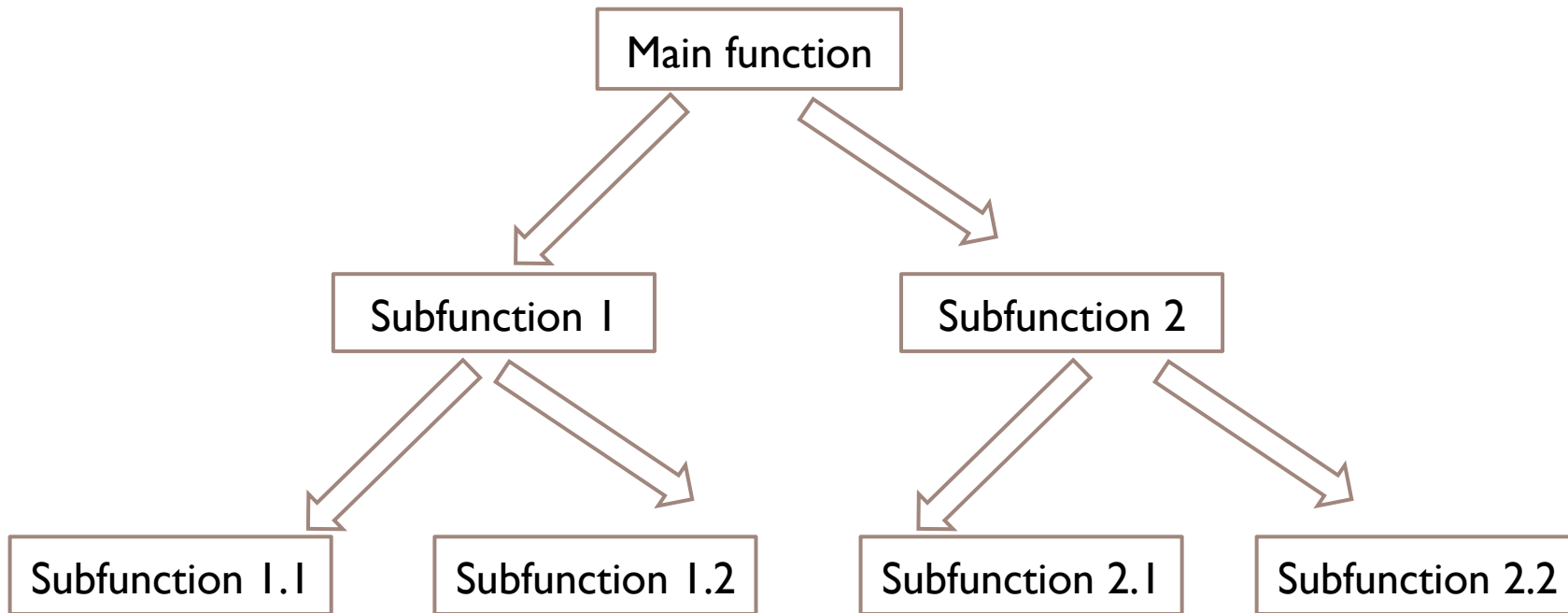


Functional decomposition

- ▶ Traditional approach
- ▶ Each module is a step of the global process
- ▶ Functional division from specification to subprograms



Functional decomposition



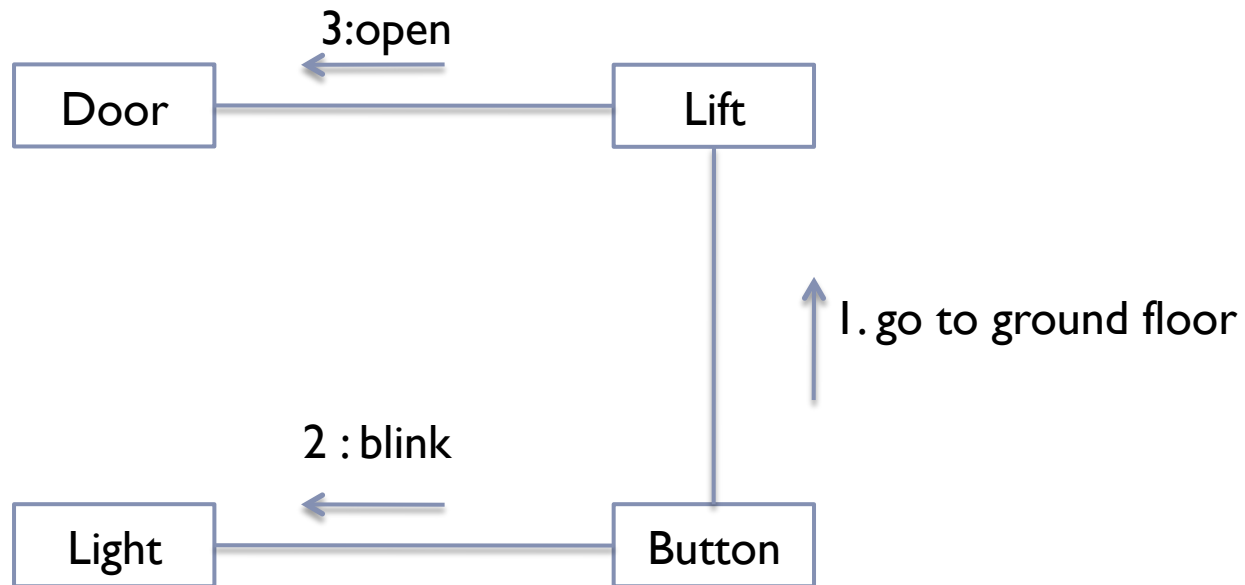
Object decomposition

- ▶ More recent approach (computer systems)
- ▶ Each module is an object of the application
- ▶ Objects are autonomous entities that collaborate to reach a goal



Object division

- ▶ Function is carried with collaborative objects



Decomposition/ Composition

- ▶ Object decomposition is restrictive
- ▶ Object approach is not only top-down
- ▶ Top-down, bottom-up, recursive, iterative, incremental



Comparison functions / objects

- ▶ Both are interesting but really different
- ▶ One must be chosen, to start to decompose



Functional approach

- ▶ More intuitive
- ▶ Focus on “DO”
- ▶ Suits when all is known in advance
- ▶ **BUT**
 - ▶ Stiff Architecture
 - ▶ Evolvability is limited
 - ▶ Not suitable to discovery



Object approach

- ▶ Focus on “BE”
- ▶ Simple (small number of concepts)
- ▶ Reasoning on abstraction (object of the domain)
- ▶ Suitable for discovery and evolvability
- ▶ BUT
 - ▶ Hard to understand for people used to functional approach.



Object Oriented advantages

- ▶ **Lead to more stable model**
 - ▶ Based on real world
- ▶ **Independancy from fuctions**
 - ▶ Evolvability
- ▶ **Encapsulate complexity**
 - ▶ Suitable for reuse



System complexity : conclusion

- ▶ Computer systems are complex by nature
- ▶ Necessity to manage this complexity
- ▶ Systems can be decomposed according to what they DO or what they ARE?
- ▶ The object approach manage with more efficiency the complexity
 - ▶ Reuse, evolvability, stability



What do we need ?

- ▶ **A modelling language**
 - ▶ Clear notation
 - ▶ Usability
 - ▶ Not too complex
 - ▶ Exchange data between developers, and stakeholders
 - ▶ Completeness and consistency semantics
- ▶ **A development process**

Method = Language + Process

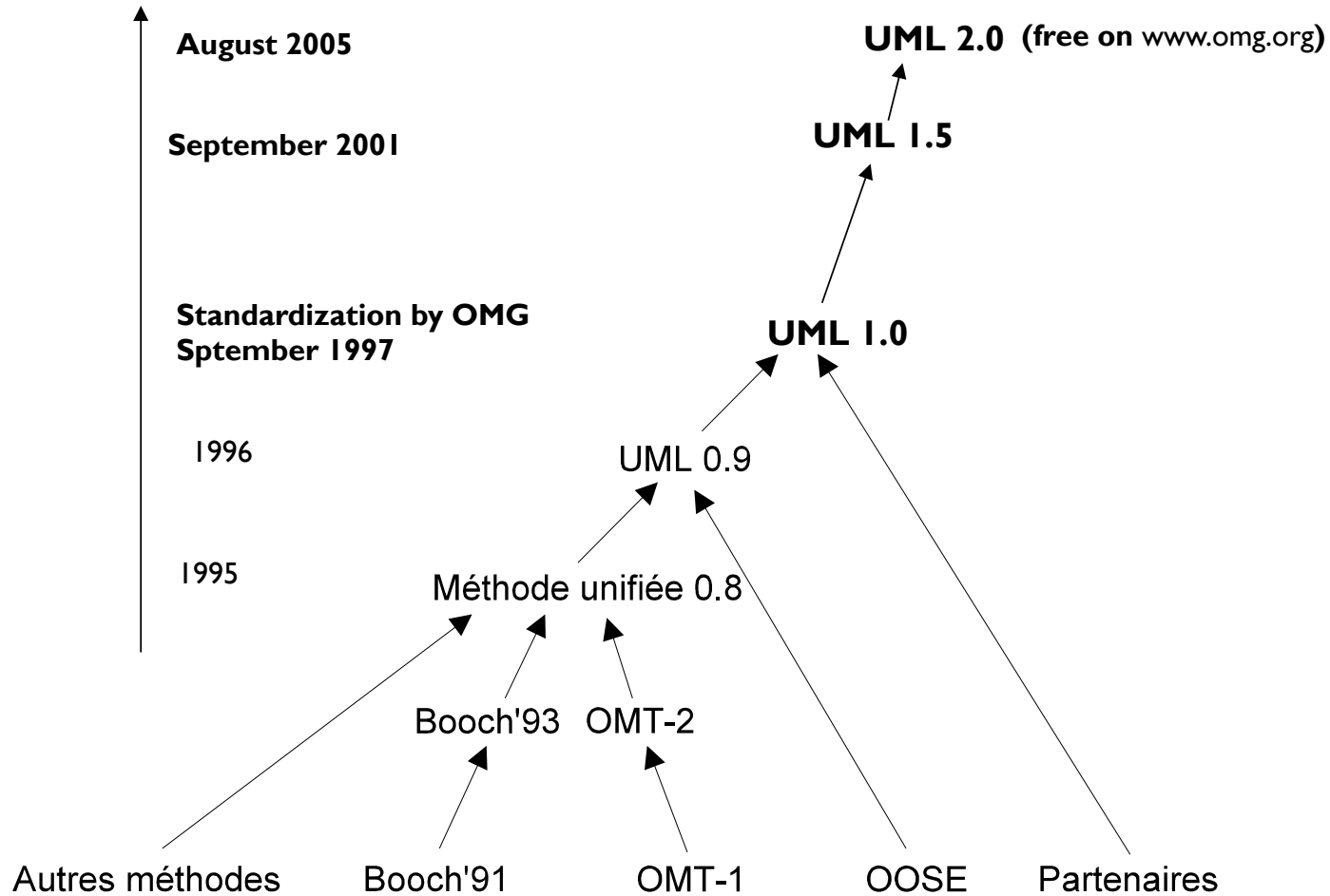


The unified notation UML

- ▶ Comes from BOOCH, OMT and OOSE
- ▶ And take good ideas from other methods
- ▶ Convergence of notations
- ▶ A unique example of standard notation which is a de facto standard (in computer science)



UML development



Summary

- ▶ UML is a notation not a method
- ▶ UML is an object modelling language
- ▶ UML is suitable for all object development
- ▶ UML is free

UML is a de facto standard for the notation of object oriented development

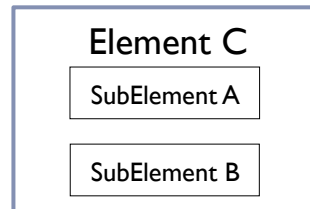




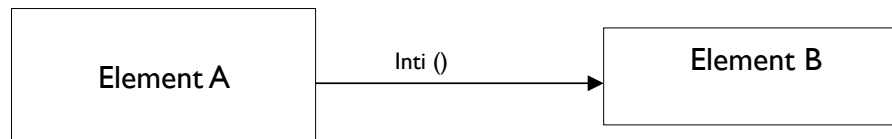
UML diagrams

Two types : structural and dynamic

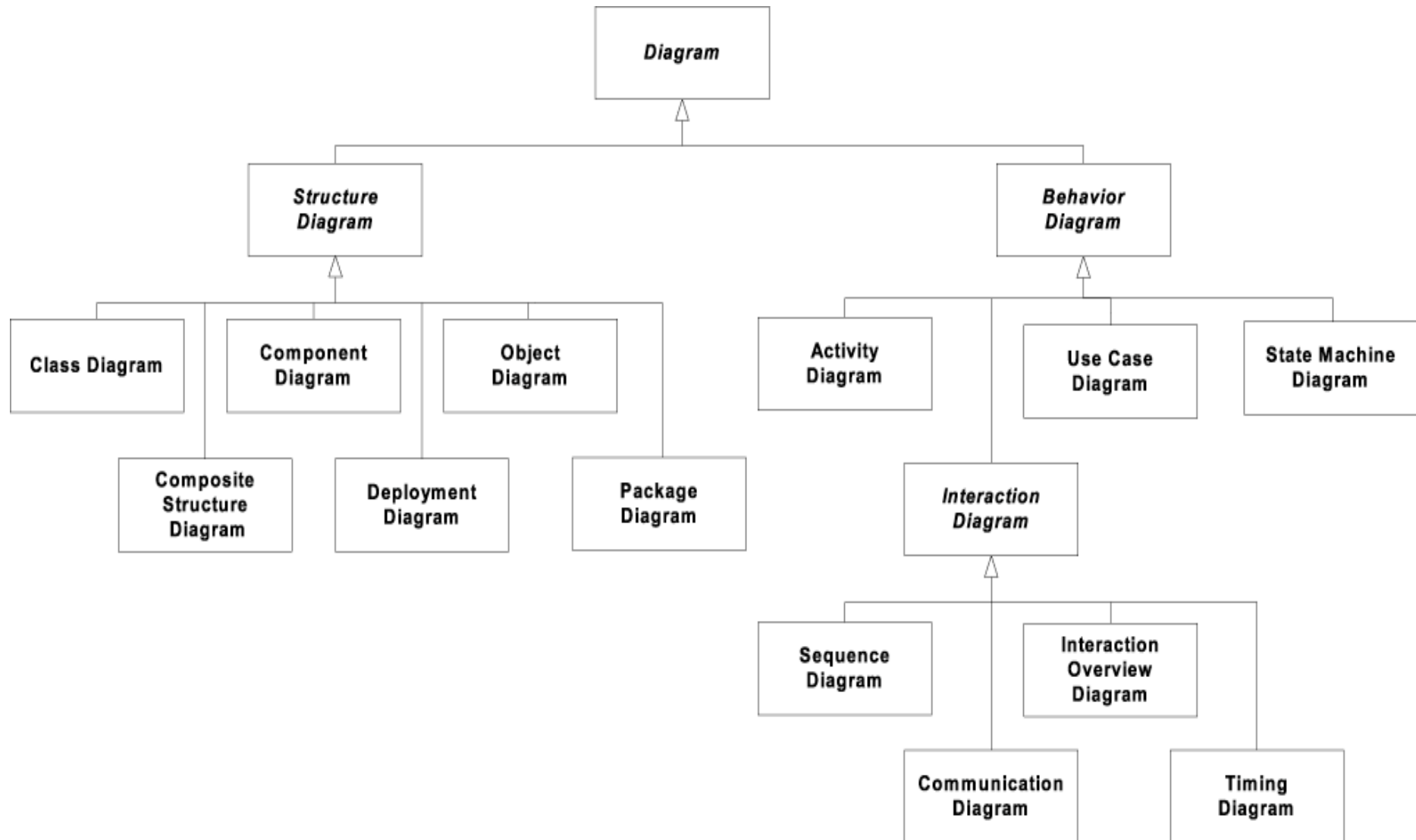
- ▶ **Structural representation fo an element**
 - ▶ Internal structure (composition) et external (relationships and dependencies wth other elements)



- ▶ **Dynamic representation**
 - ▶ Behavior considering time : interaction with other elements, modification of its internal state...

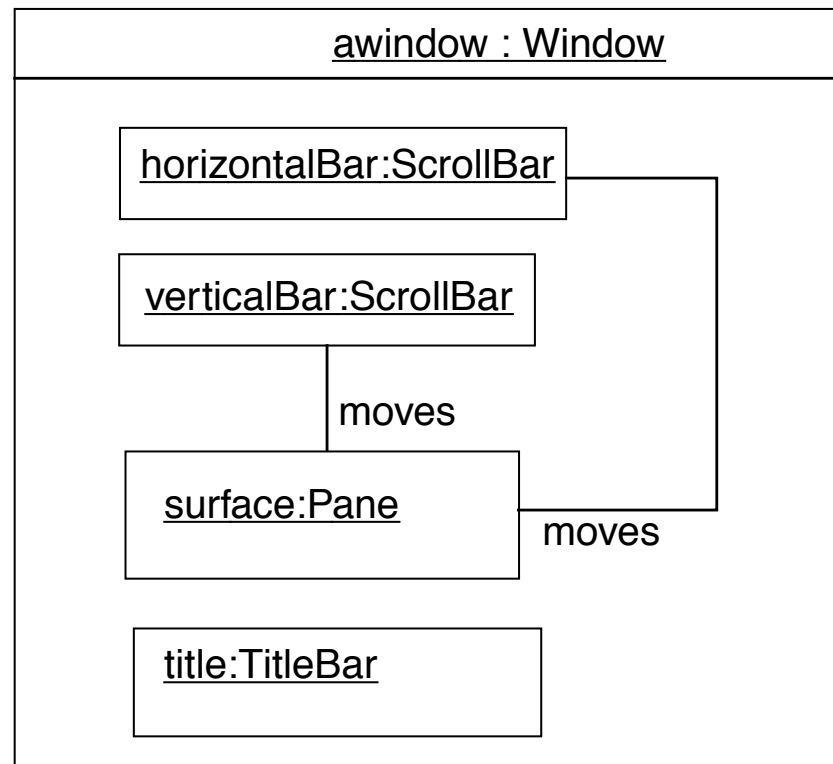


UML 2 diagrams



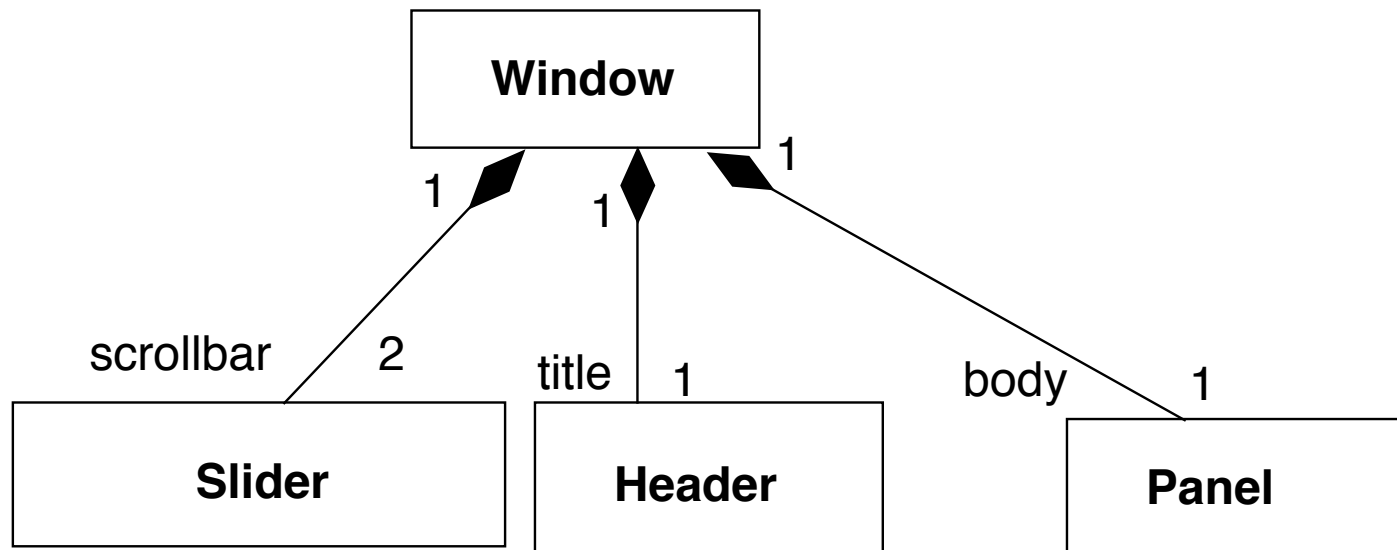
Object diagram

- ▶ Represents *objects and their relationships*



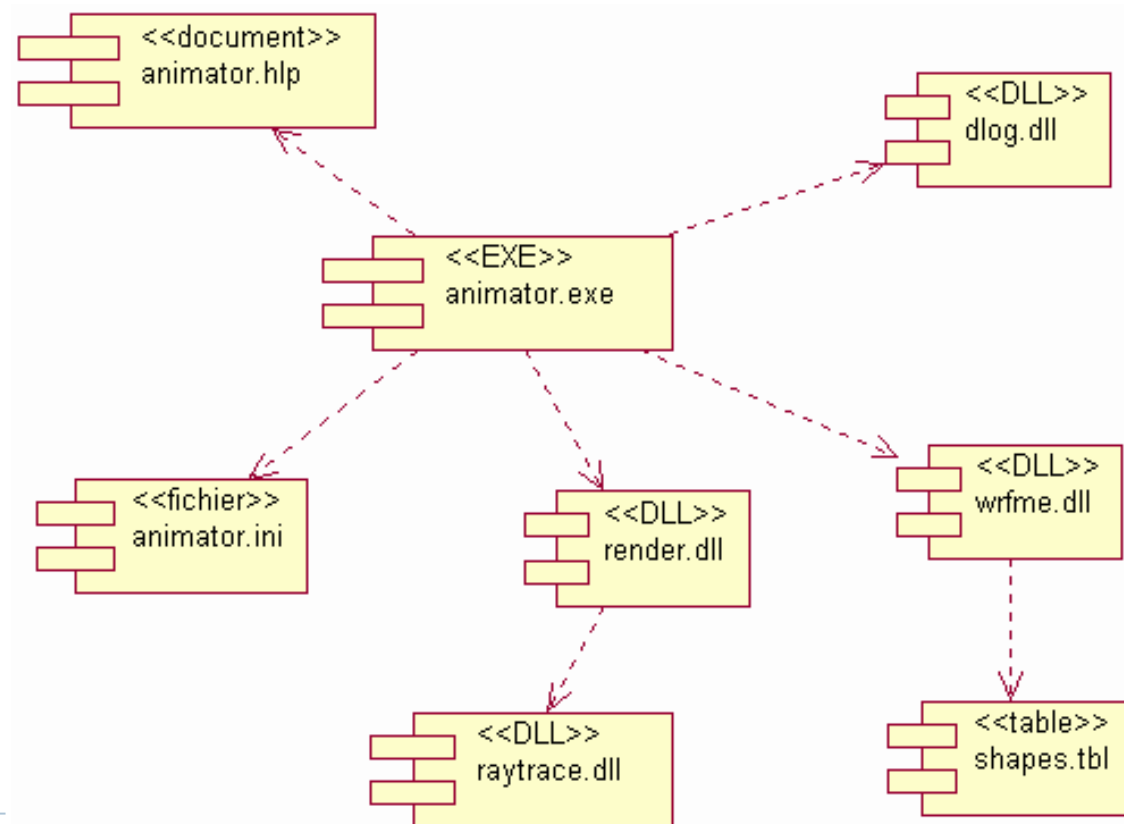
Class diagram

- ▶ *Represents static structure with classes and their relationships*



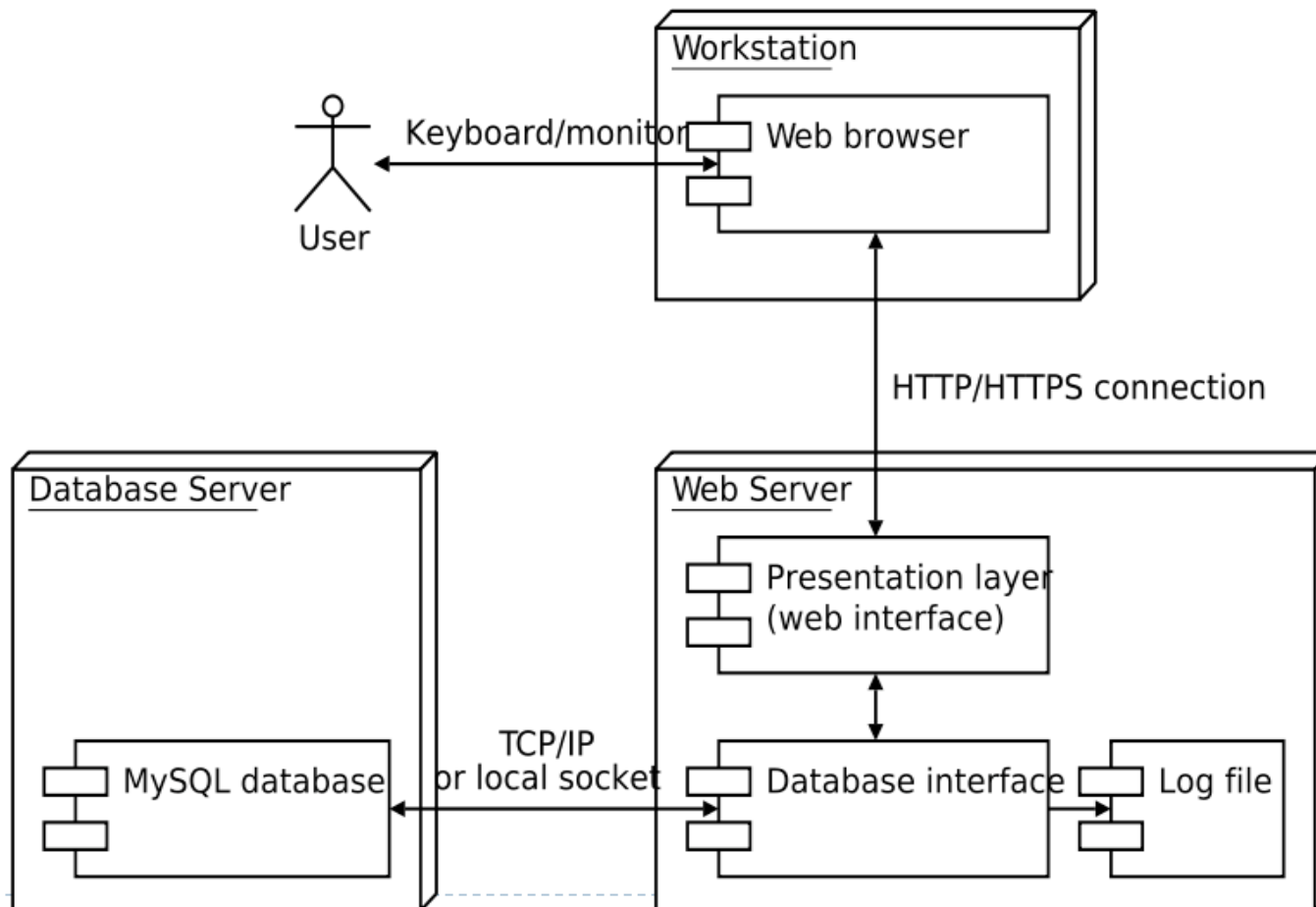
Component diagram

- Represents *physical components* of a system



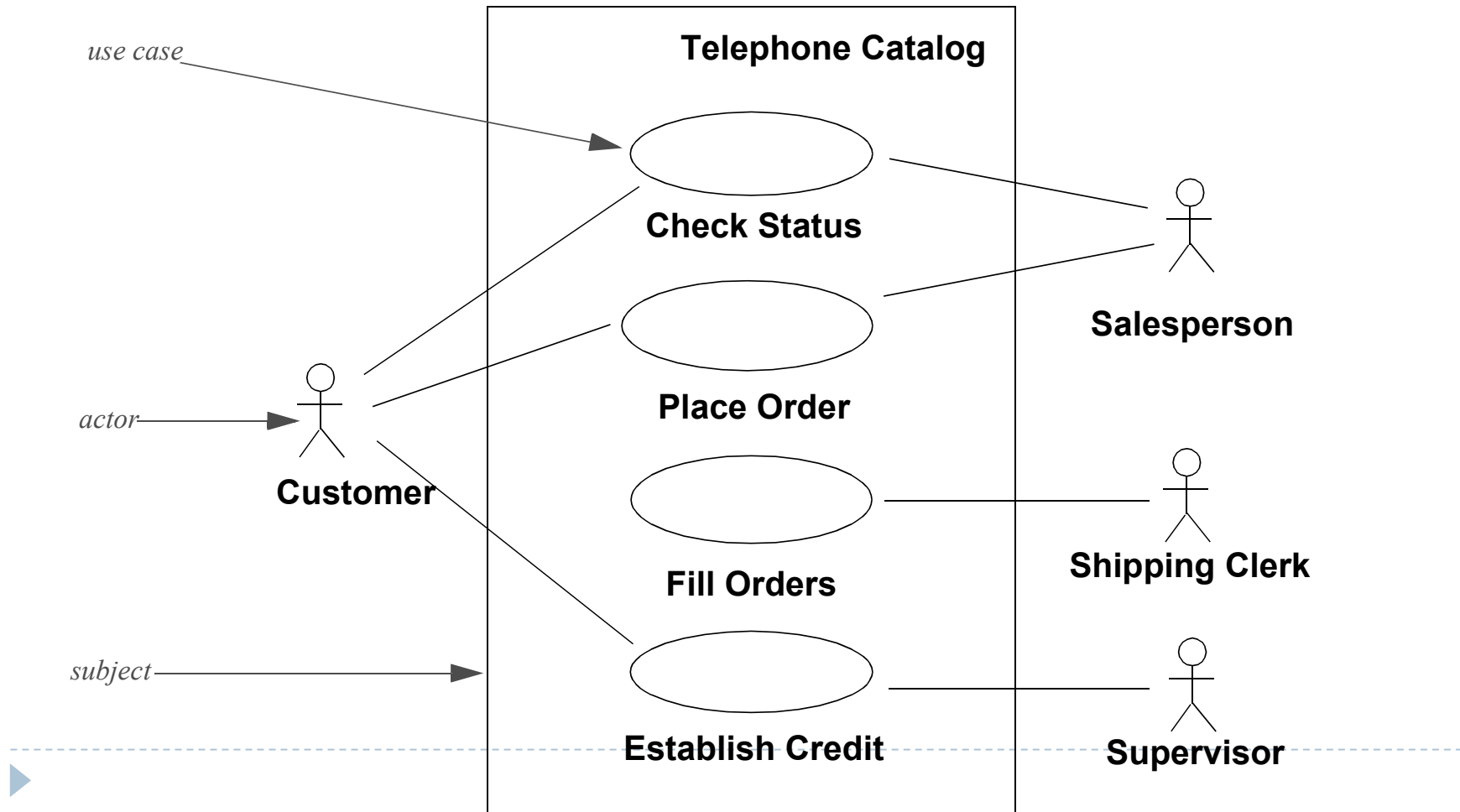
Deployment diagram

- ▶ Represents the *deployment of the components on hardware devices*



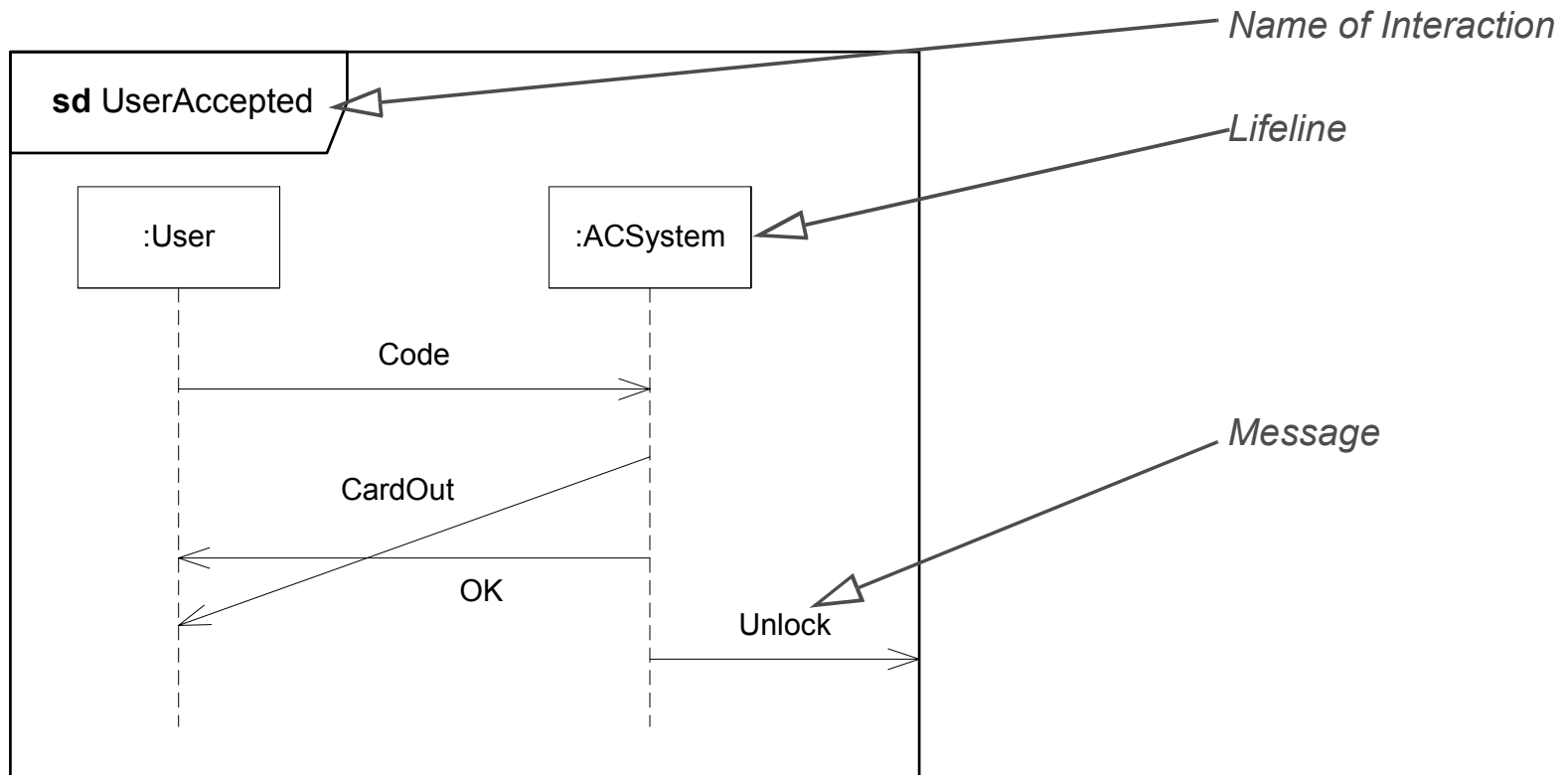
Use case diagram

- ▶ Represents objectives of the use of the system according to actors view point



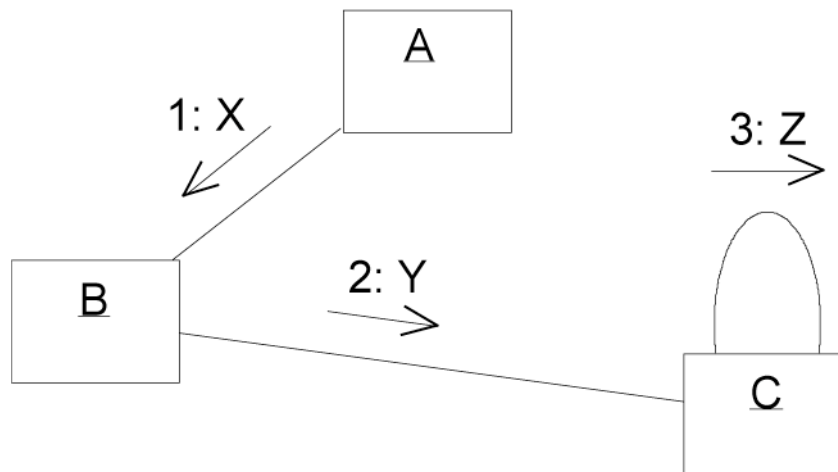
Sequence diagram

- ▶ Represents interactions between objects according to *time*.

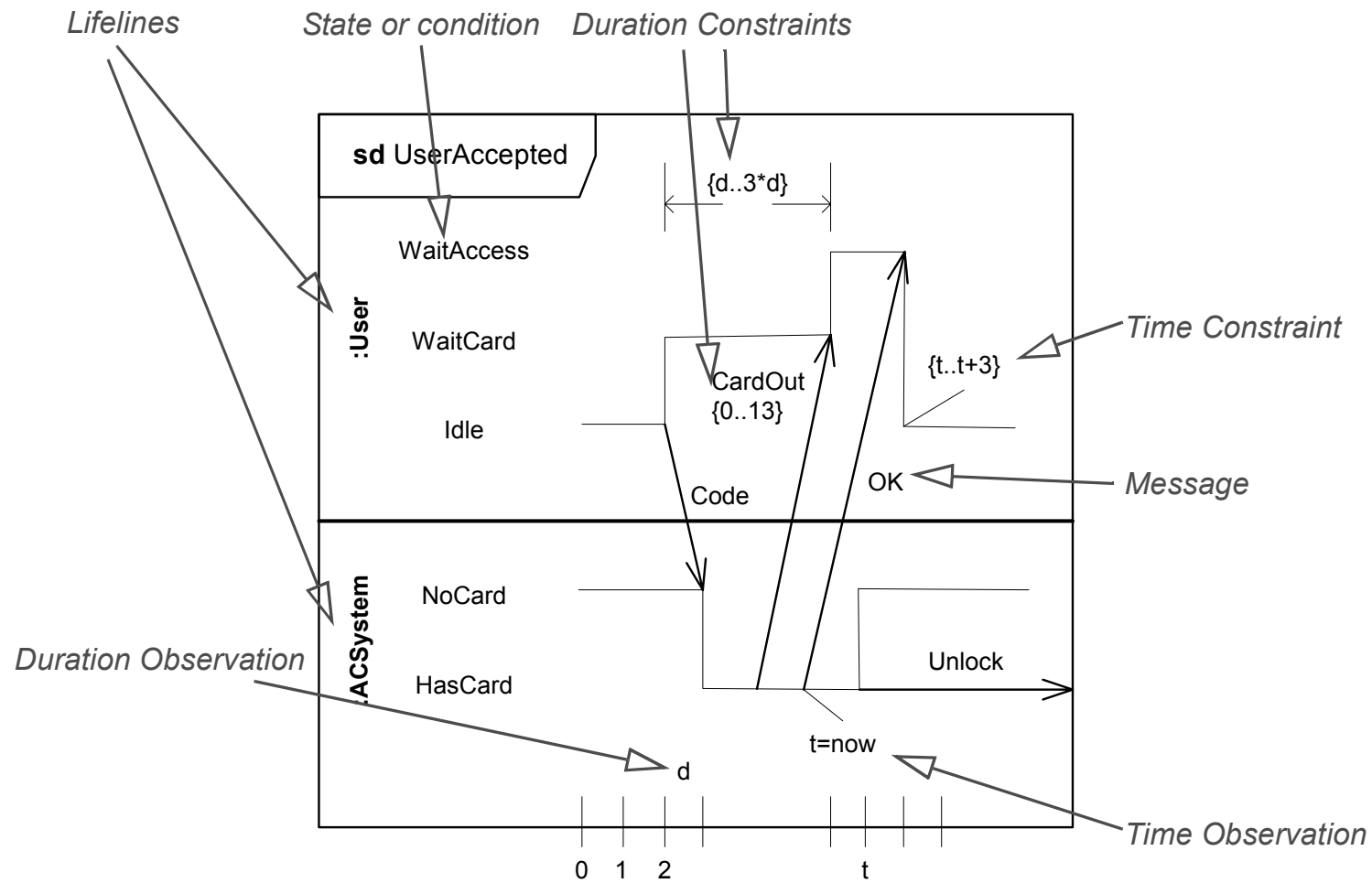


Communication diagram

- ▶ *Equivalent to sequence diagram but with a **spacial** representation*

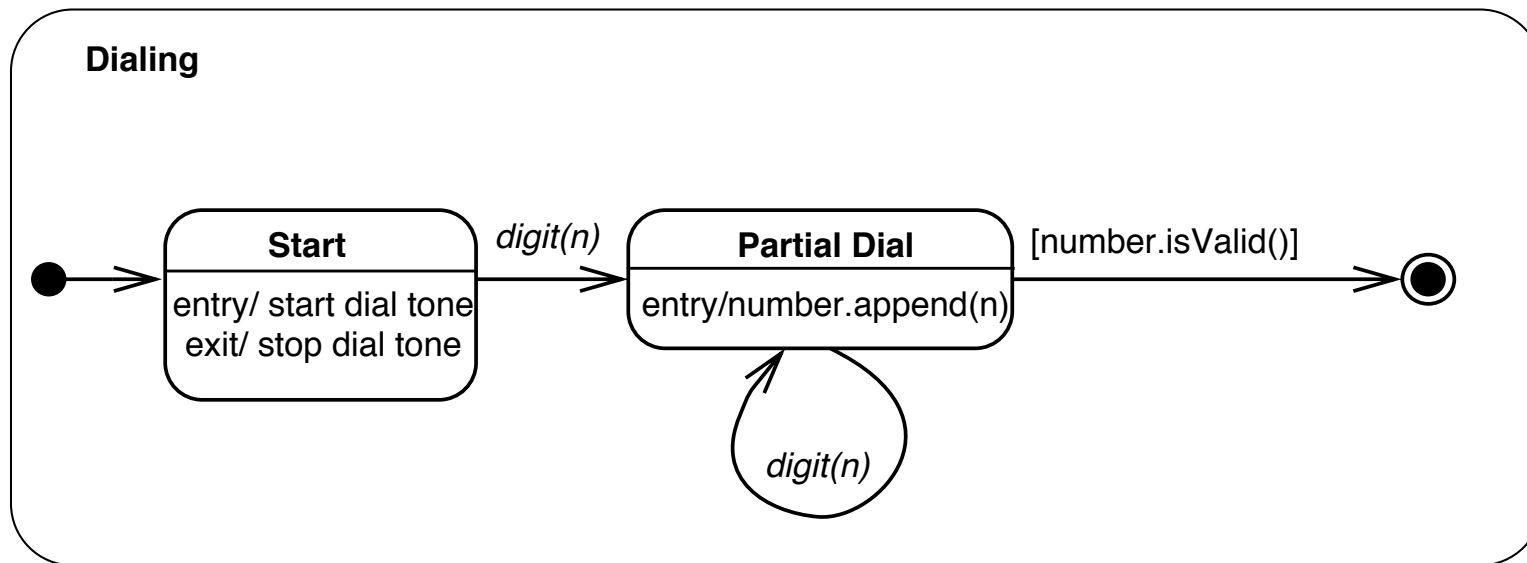


Timing Diagram



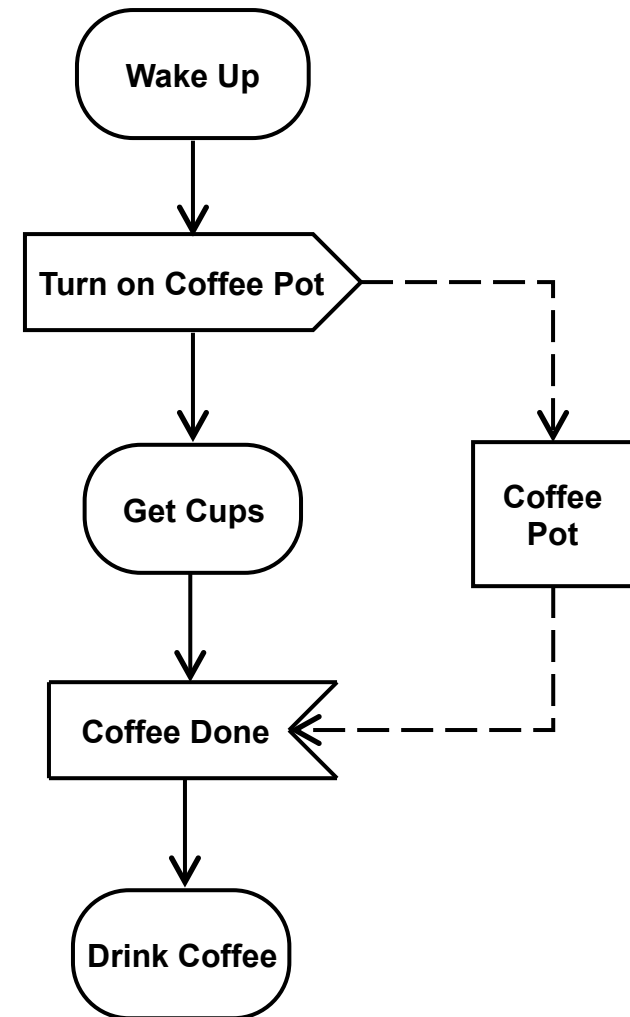
State-Transition diagram

- ▶ Represents *life cycle* of an object



Activity diagram

- ▶ *Represents an activity flow in an operation, a use case or a business process*





Classes and objects

The objects

- ▶ Real world objects born, live and dead
- ▶ Computer system objects are a simple representation of real world elements
- ▶ Objects represent concrete entities (a sensor, an actuator) or abstract (PID regulator, Neural...)



Graphical notation of object

One object

Another object

And another one



Objects are abstractions

- ▶ An abstraction is a summary
- ▶ Of essential characteristics
- ▶ Hide the details
- ▶ An abstraction depends on a viewpoint (e.g. mathematical, automatic, architectural)



Abstraction examples

- ▶ A television
- ▶ A complex number
- ▶ A financial operation
- ▶ A logical gate
- ▶ A battery
- ▶ An actuator
- ▶ A sensor

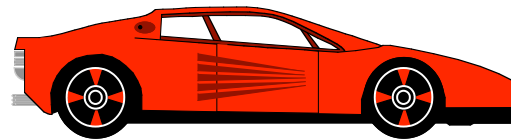
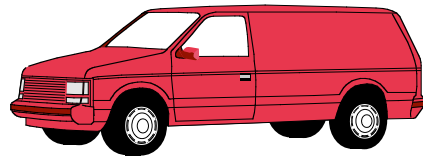
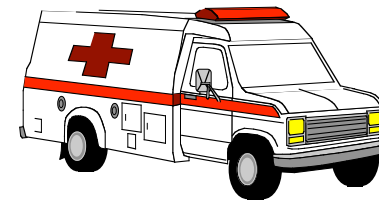
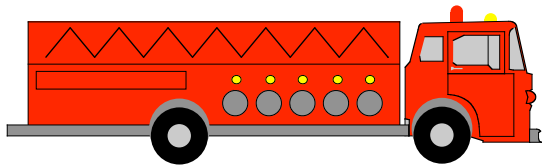


Object chaos

- ▶ Many many objects
- ▶ To understand : categorization, classification
- ▶ Humans are always classing : animals, plants, mushrooms, atoms...



Object chaos cont'd

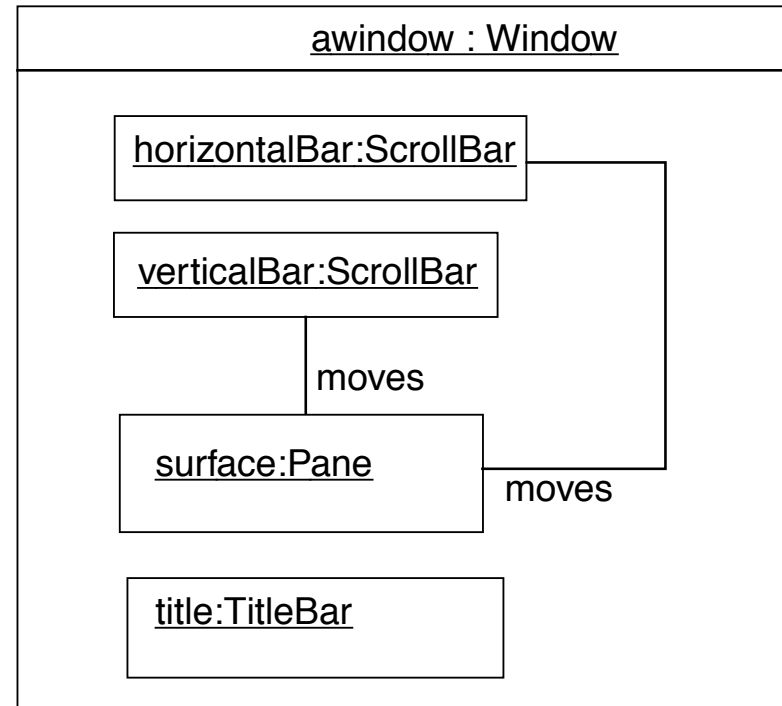
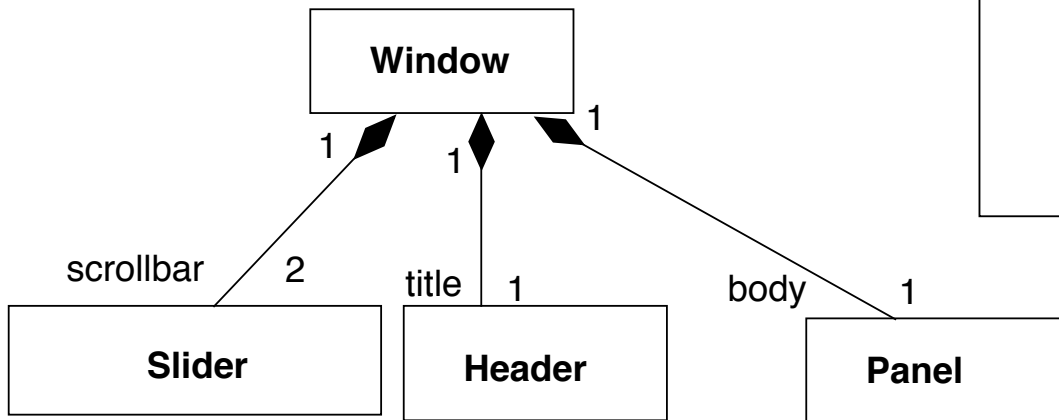


Classes

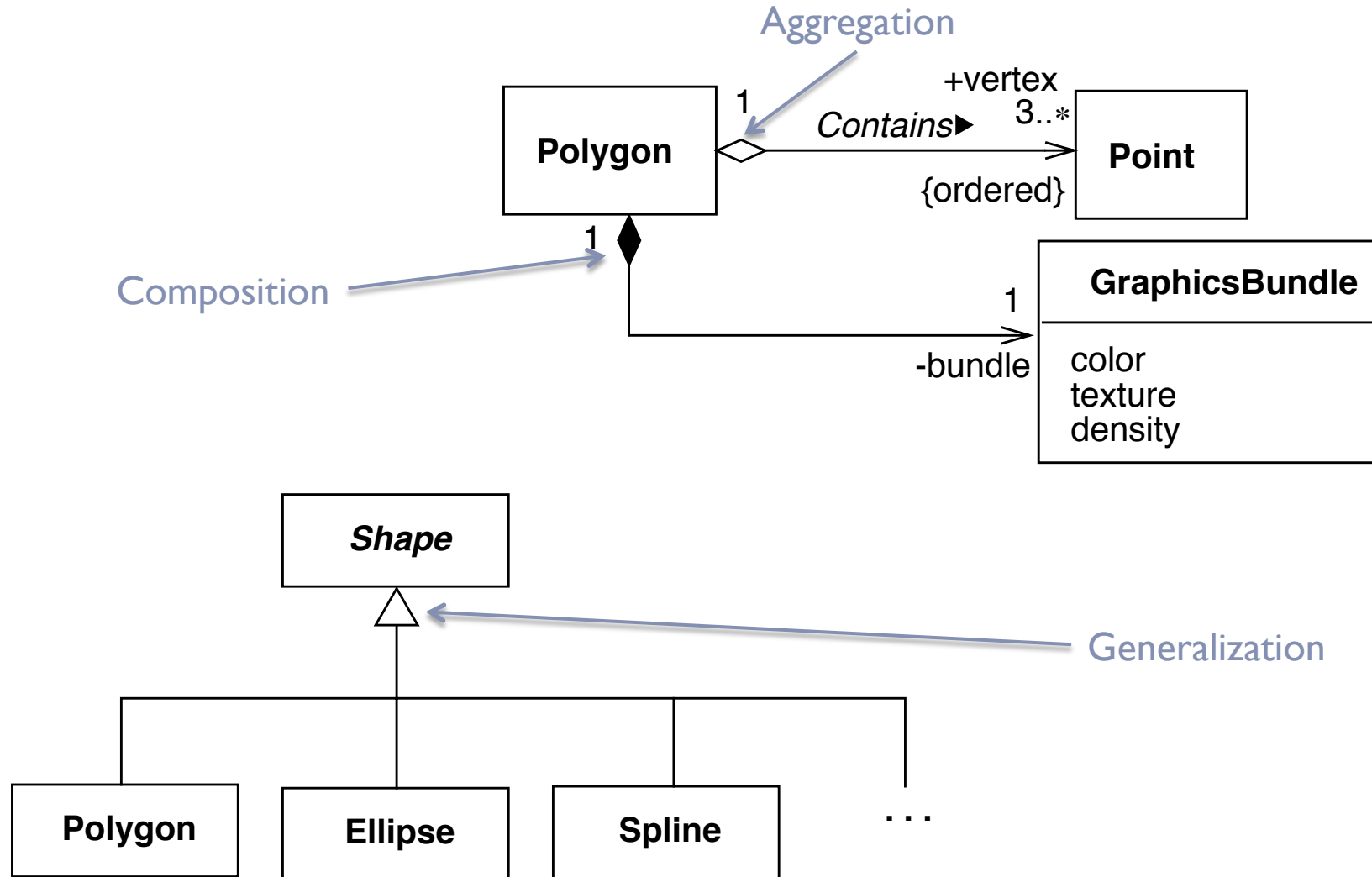
- ▶ A class is an abstraction of several objects
- ▶ Can be interpreted as a factorization



Classes and objects



Classes Relationship

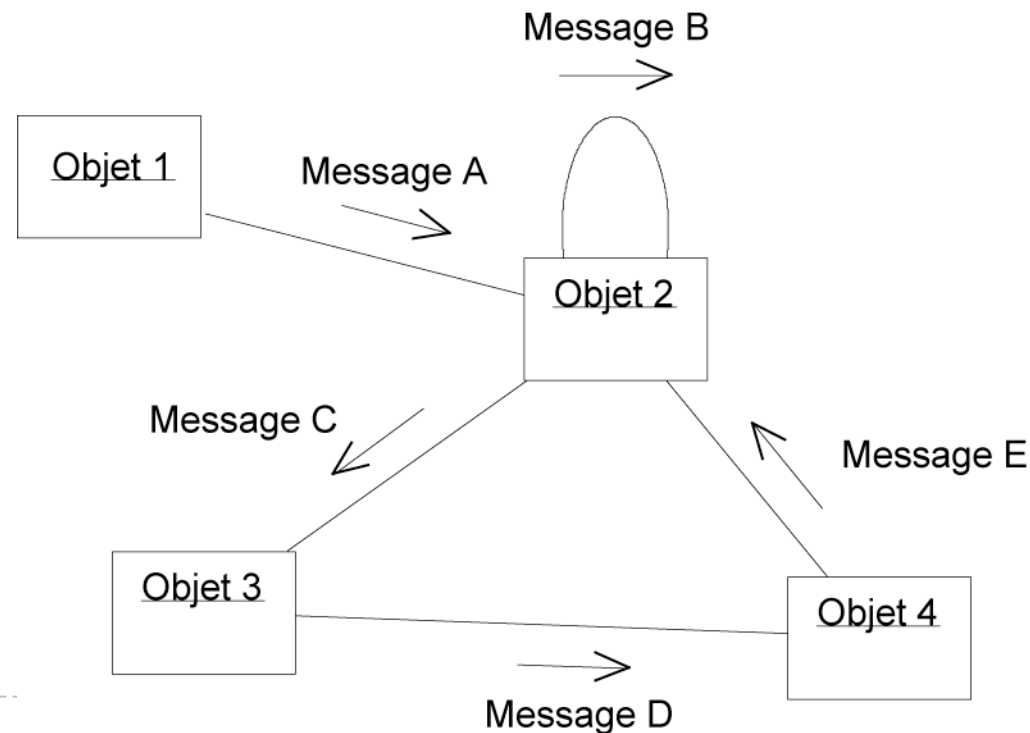




Object dynamics

Communication between objects

- ▶ System = society of collaborative objects
- ▶ Object work together to perform the service
- ▶ The behavior of a system depends on how the objects collaborate



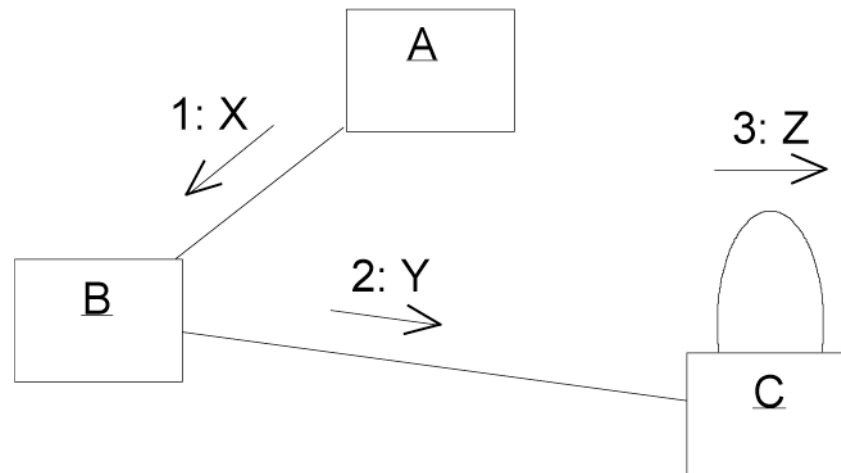
A message

- ▶ Is the communication unit between objects
- ▶ Very general concepts with various application
- ▶ Can represents both control and data flow
- ▶ And also events, or activities



Communication diagram

- ▶ **A** send a message *X* to object **B**, the object **B** sent *Y* to **C**, then etc...

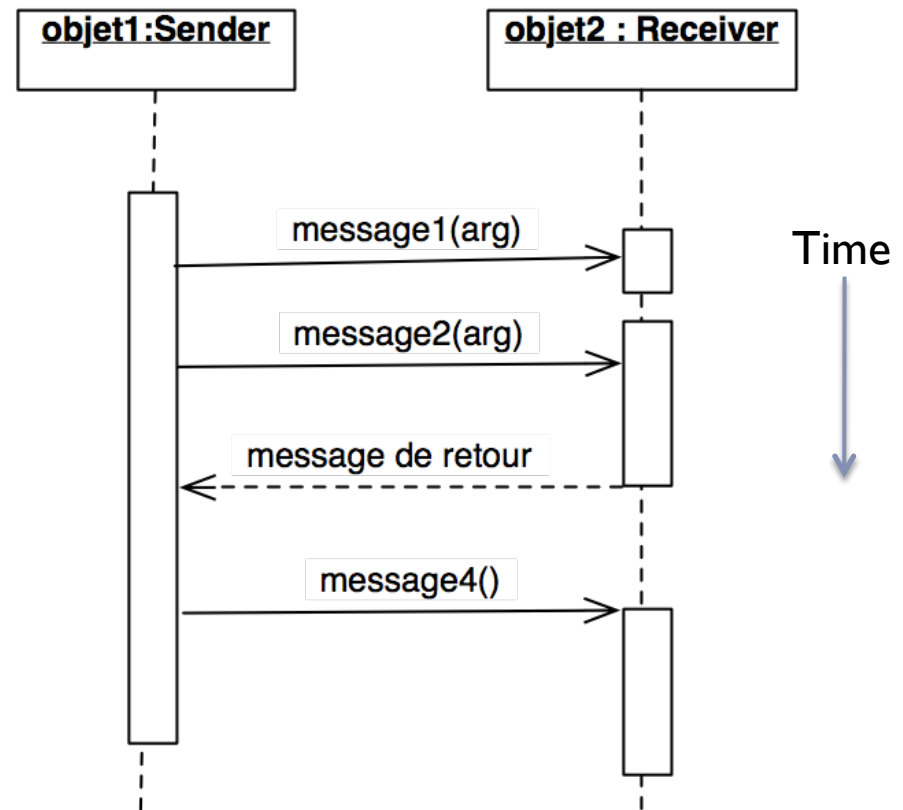


Sequence diagram

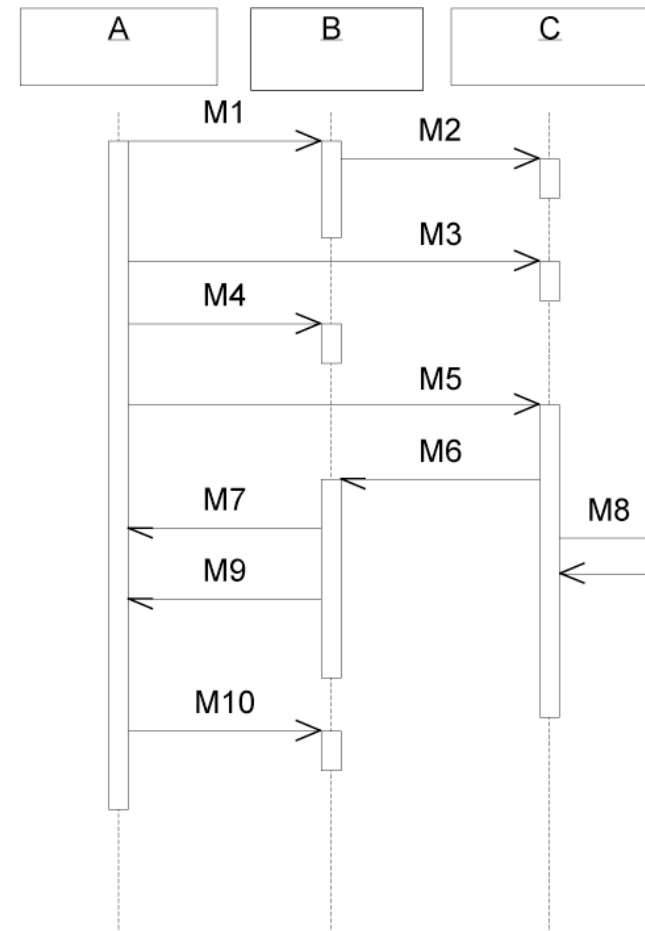
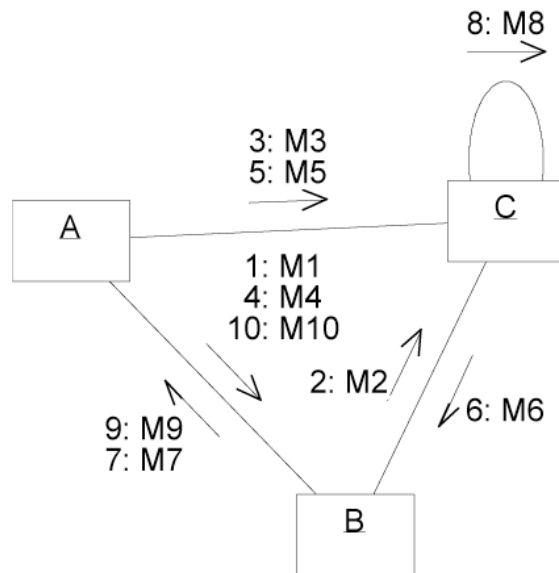
- ▶ Lifelines are objects
- ▶ The tag is objectName:ClassName

→ is for a message

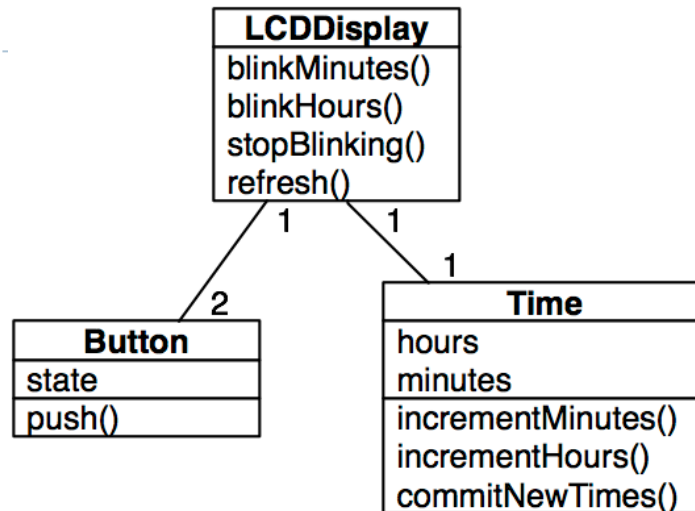
- -> is a return



Collaboration and sequence diagram



Exercise : Simple Watch

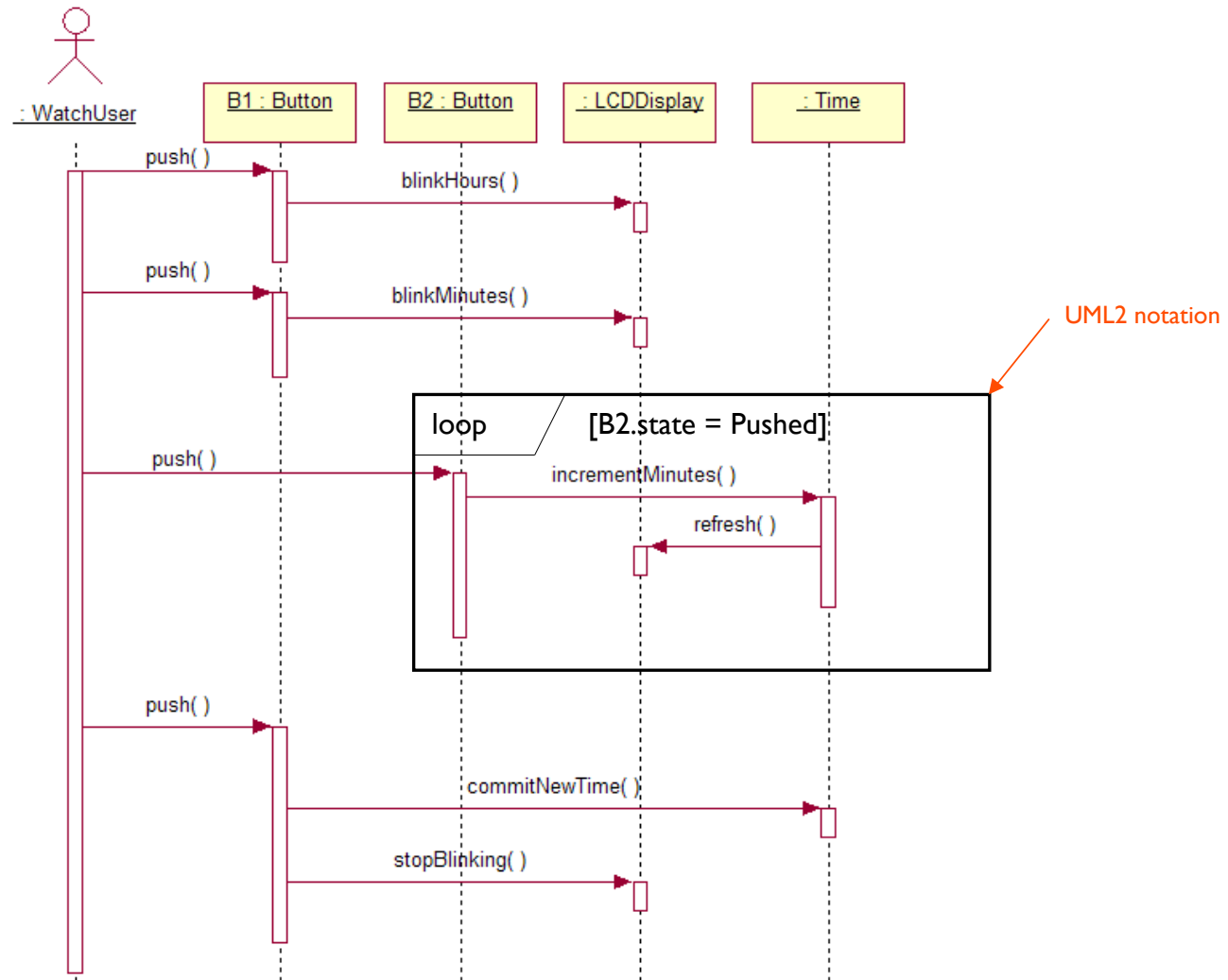


From class diagram :

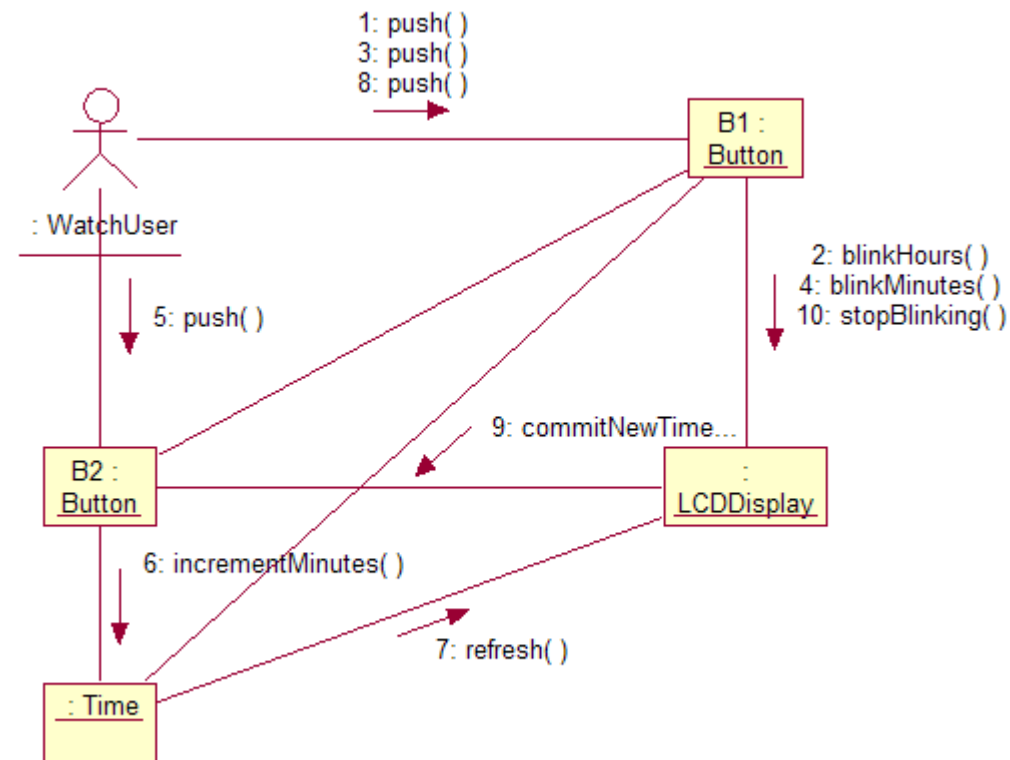
- I. Perform a sequence diagram of the following scenario : a user wants to set the minutes

Pushing twice the button 1, he can set the minutes (hours blinks and then minutes). Then with the button 2 (with releasing it), minutes are incremented. Once minutes are set, the user push the button 1 and the minutes stop blinking.

Simple watch: Sequence diagram



Simple Watch: Communication diagram



State machine diagram

- ▶ **A state machine diagram is used to represents**
 - ▶ Lifecycle of an object (instance of a class),
 - ▶ Events that produce transitions from a state to another
 - ▶ Actions due to change state



State

- ▶ Initial state



- ▶ Intermediate state

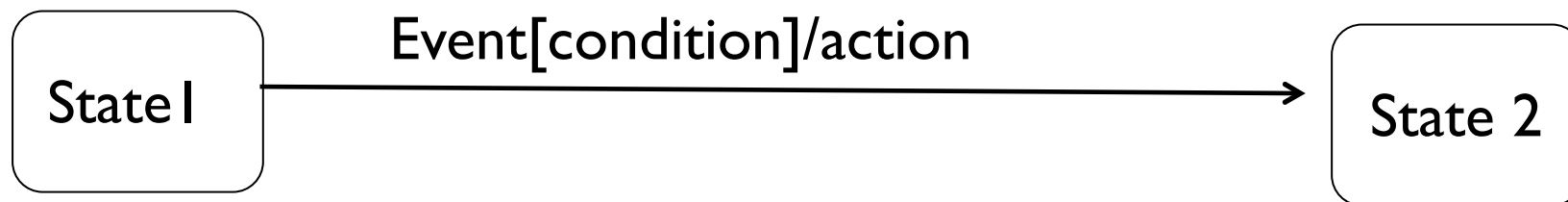


- ▶ Final state



Transitions

- ▶ Unidirectional connections for a directed graph
- ▶ Triggered by an **event** if the **condition** is true and produce an **action**.



Event, condition and actions

- ▶ Event

- ▶ 4 main types :

- Signals : asynchronous occurrence of external event (e.g. button pushed)
 - Call: another object request a service
 - Change of an attribute (ex: battery level = 10%)
 - Time : delay (after 15s) or absolute time event (time=12.42 pm)

- ▶ Condition : boolean expressions

- ▶ Actions :

- ▶ Services/operation of the class
 - ▶ Instantaneous, cannot be interrupted

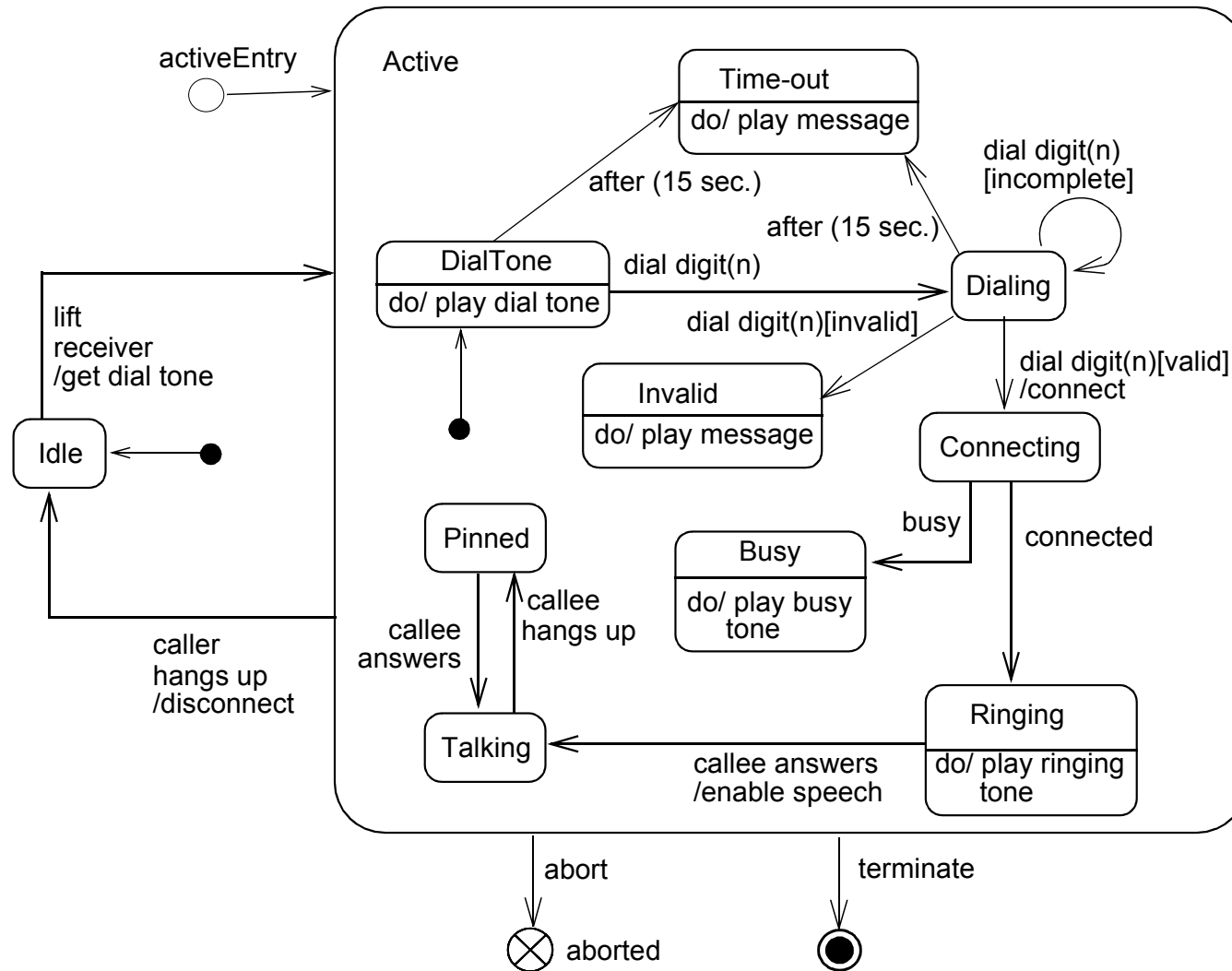


Exercice : Telephone

- ▶ Model the state machine of the telephone



State diagram example

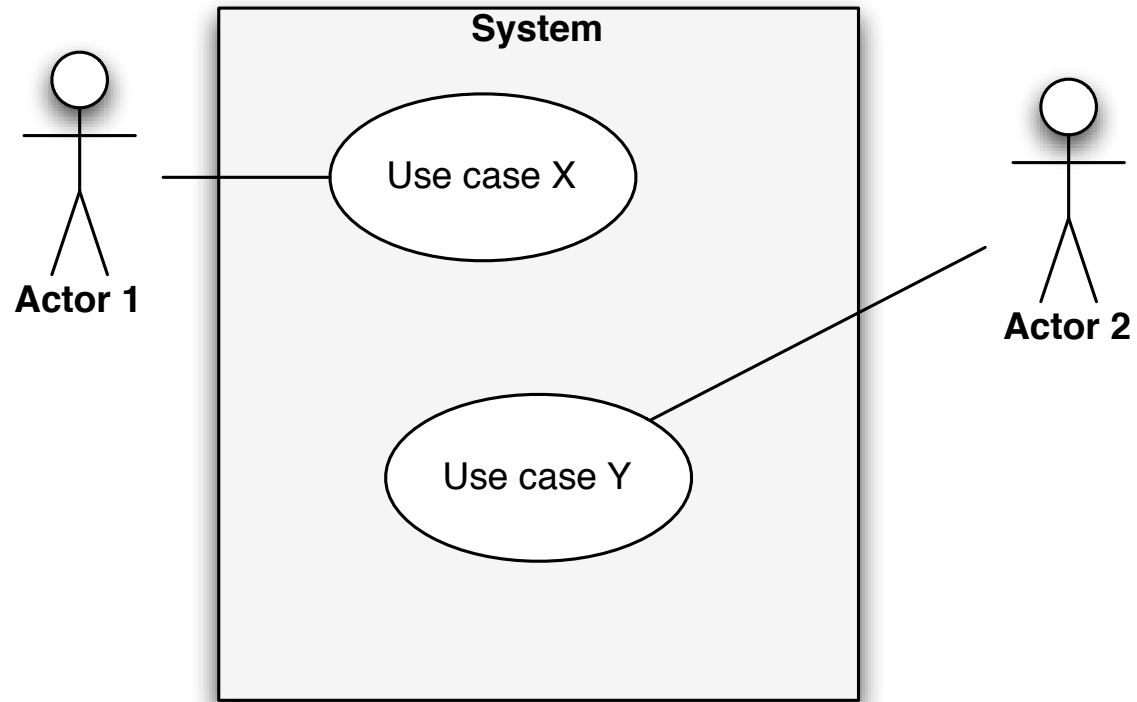




Use cases

Use cases

- ▶ Represent functional requirement

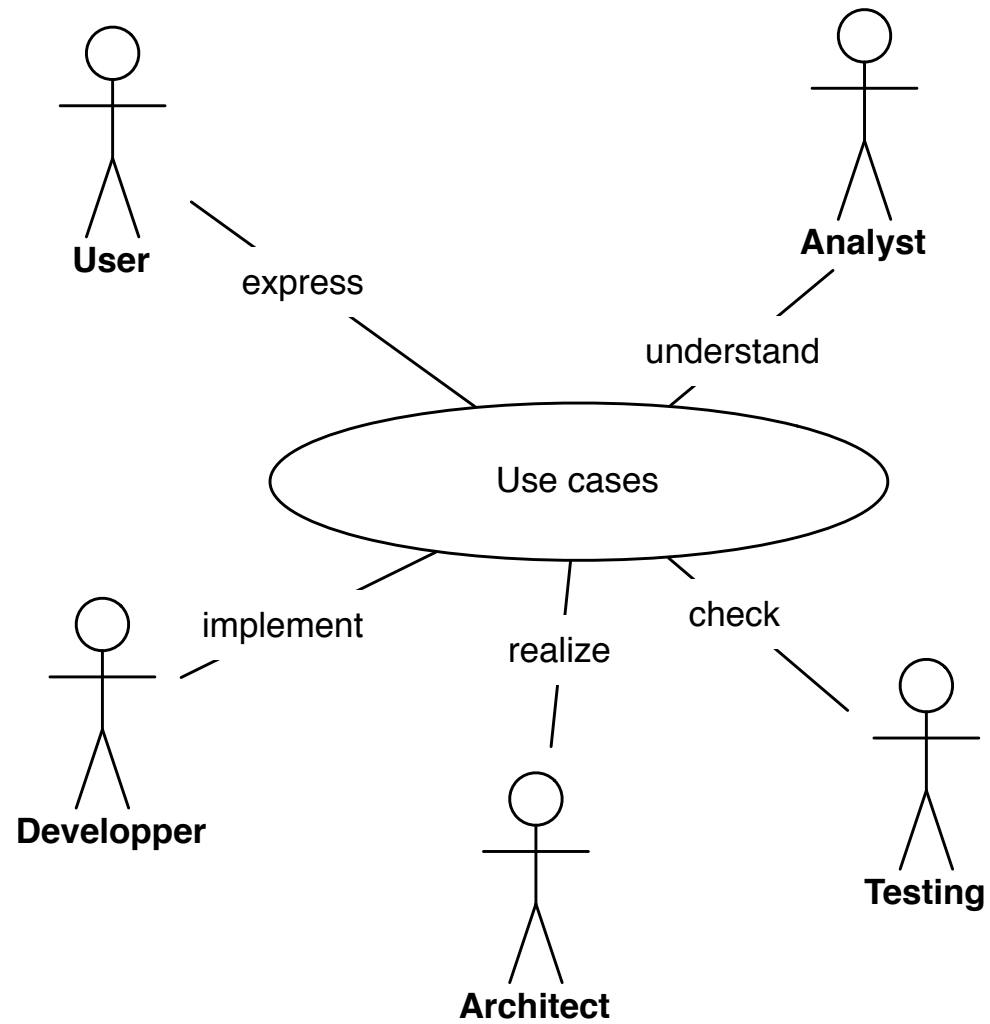


Why use case diagrams ?

- ▶ A graphical modelling of requirements
- ▶ Used by final users to express/discuss about their requirements
- ▶ Are usefull to communicate at the first steps of the developement
- ▶ Are a basis for functional testing



Project main thread



Actors

- ▶ Represent roles that humans, hardware devices, or external systems play while interacting with a given system
- ▶ They are not part of the system and are situated outside of the system boundary
- ▶ Actors may be both at input and output ends of a use case



Identify actors

- ▶ Define system boundary to identify actors correctly
- ▶ Identify users and systems that depend on the system's primary and secondary functionalities
- ▶ Identify hardware and software platforms with which the system interacts
- ▶ Select entities that play distinctly different roles in the system
- ▶ Identify as actors external entities with common goals and direct interaction with the system
- ▶ Denote actors as nouns



Identifying Use Cases

- ▶ **Business / Domain Use Cases:**
 - ▶ Interactions between users and the business (or domain)
- ▶ **System Use Cases:**
 - ▶ Interactions between users and the system
 - ▶ One business use cases contains a set of system use cases
- ▶ **To name the use cases, give it a verb name to show the action that must be performed**
 - ▶ Describe a transaction completely
 - ▶ No description of user interface whatsoever



Capture Use Cases

- ▶ Capture use cases during requirements elaboration
- ▶ Use cases are not mapped one-to-one to requirements
 - ▶ Each requirement must be covered by at least one use case
 - ▶ However, use cases may contain many requirements
- ▶ Use scenarios to model assumptions and define system scope
- ▶ List exceptions separately



Scenarios

- ▶ **Specify behaviour of use case by description, not modeling**
 - ▶ Examples include informal structured text, formal structured text with conditions, and pseudocode
- ▶ **Typically specify:**
 - ▶ How and when the use case starts and ends
 - ▶ Interaction with the actors and the exchange of objects
 - ▶ Flow of events: main / typical (success), alternative (success), and exceptional (failure) flows



Identifying Scenarios

- ▶ Extract the functionality that is available to each actor
- ▶ Establish specific instances and not general descriptions
- ▶ Denote situations in the current and future systems

Identify:

- ▶ Tasks to be performed by the user and the system
- ▶ Flow of information to the user and to the system
- ▶ Events that are conveyed to the user and to the system
- ▶ For the events flow, name steps in active voice



Example of textual description

<project>	
Use-Case:	<use-case name>
Brief Description	<brief description of use-case>
Actor Brief Descriptions	<Actor I Name>
Preconditions	<pre-condition I >
Basic Flow of Events	The use case begins when <actor>, <does something>... <basic flow step I > ... <basic flow step n> The use case ends.
Alternative Flows	<alternate flow I > If in step <x> of the basic flow the <actor or system does something>, then <describe flow> The use case resumes at step <y>
Subflows	<subflow I, step I > ... <subflow I, step n>
Post-conditions	<post-condition I >
Special Requirements	<special requirement I >

A process for scenario based risk
analysis

Issues

- ▶ A method usable at the very first steps of the development process
- ▶ Studying the dynamics of the system
- ▶ Not requiring important skills in modelling
- ▶ Easily understandable by non experts
- ▶ Integrating human factors

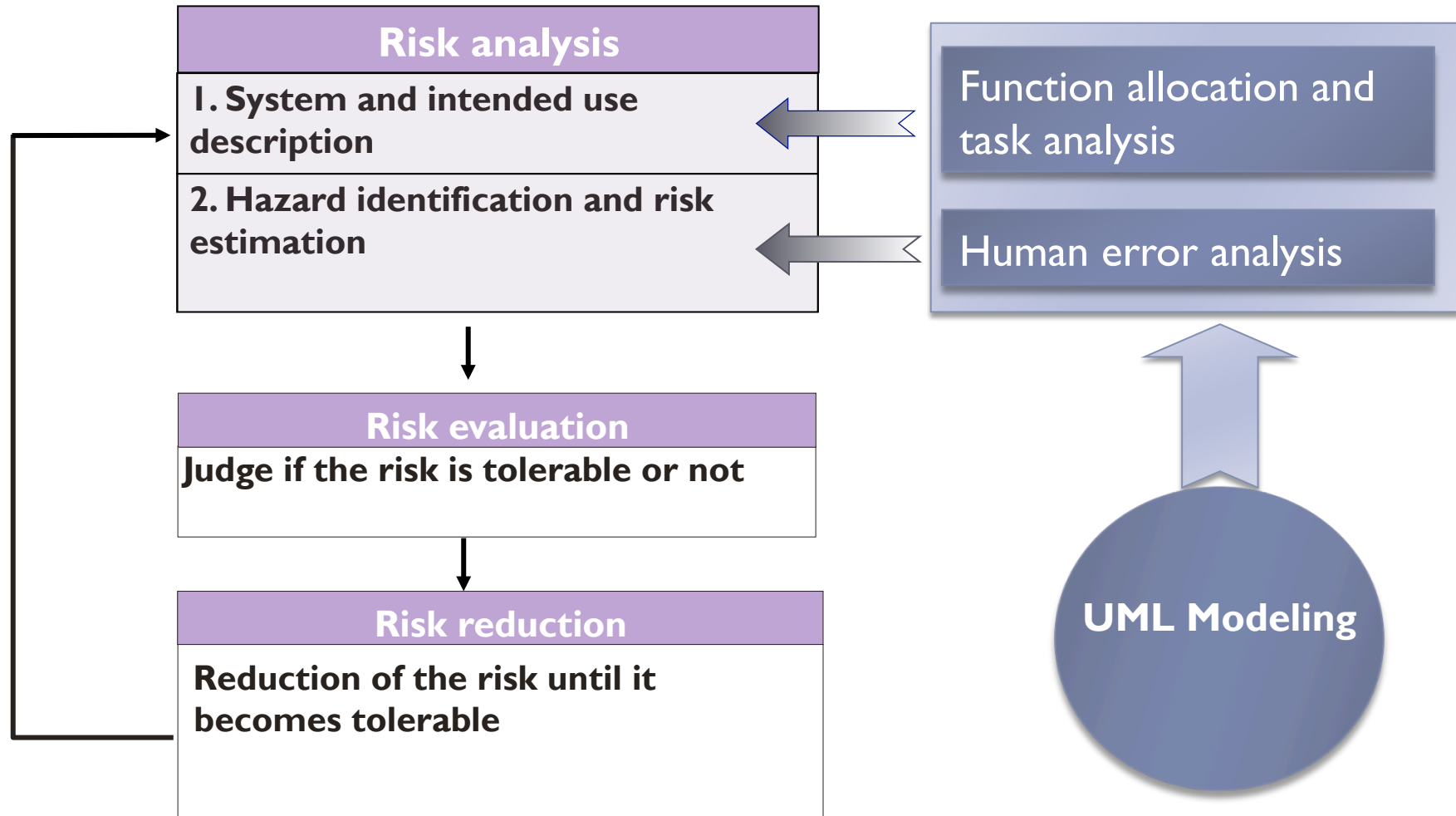


Integration of several Methods

- ▶ Based on the risk management approach
- ▶ Integrating UML
- ▶ Integrating task analysis and function allocation results
- ▶ Integrating human error analysis
- ▶ HAZOP for deviation analysis
- ▶ FTA for risk estimation



Risk management

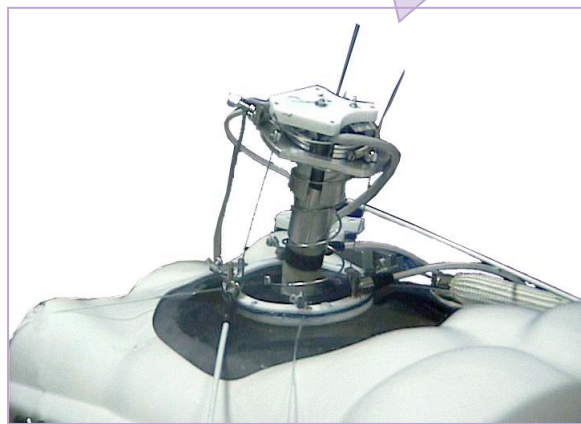
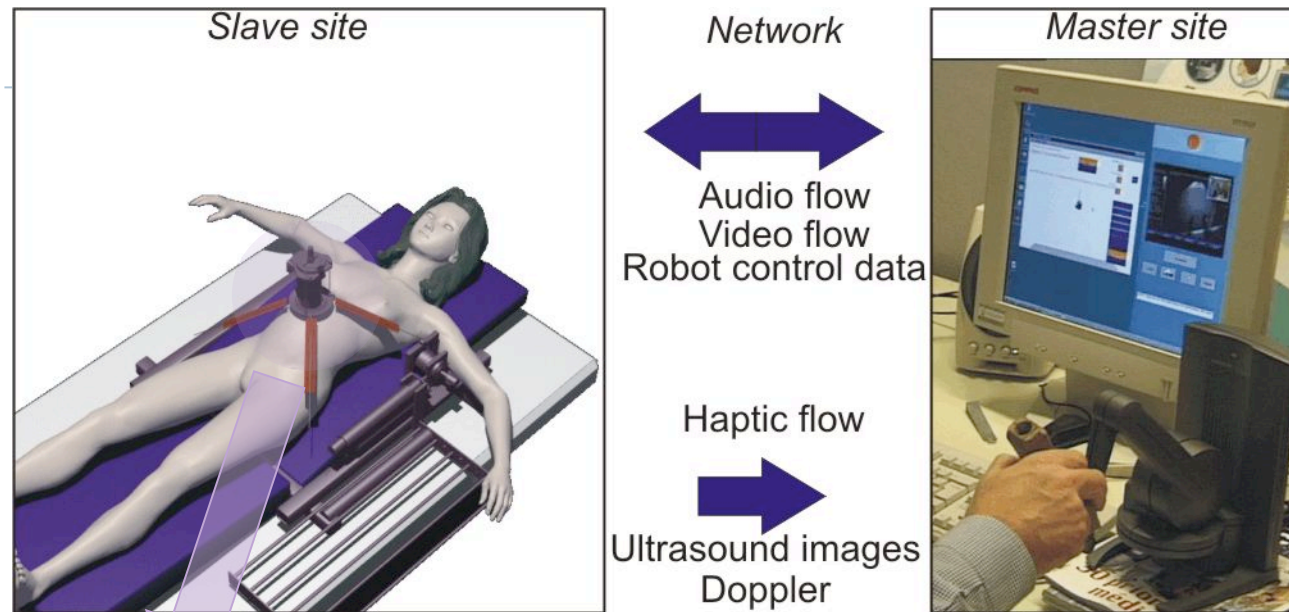


Process overview

1. **System intended use and description**
 - a) General scenario description UML Use Case
 - b) Robot integration UML Use Cases
 - c) System definition UML System use cases
 - d) Task description : UML Sequence diagrams
2. **Preliminary Hazard identification**
3. **Hazard identification with HAZOP**
 - a) HAZOP on use cases conditions
 - b) HAZOP on sequence diagrams
 - c) Communication of results and determination of top events
4. **Risk estimation**
 - a) Fault tree analysis without any risk reduction strategy
 - b) First risk estimation and determination of integrity levels (recommendations)
- ~~— Risk evaluation / Risk Reduction (not presented here)~~
5. **Residual risk estimation**
 - a) Fault Tree Analysis with risk reduction strategies (minimal cut sets analysis and use of PARETO for order 2 min. cut sets.)
 - b) Final recommendations for safeguards and integrity levels.

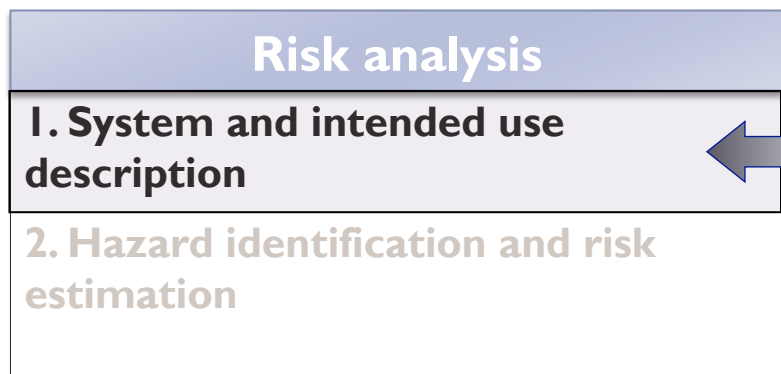


Example of application : Tele ultrasound system overview



- ▶ **Master site**
 - ▶ Expert move by hand virtual probe and diagnose
- ▶ **Slave Site**
 - ▶ parallel robot, artificial muscles





Function allocation and task analysis



System and intended use description

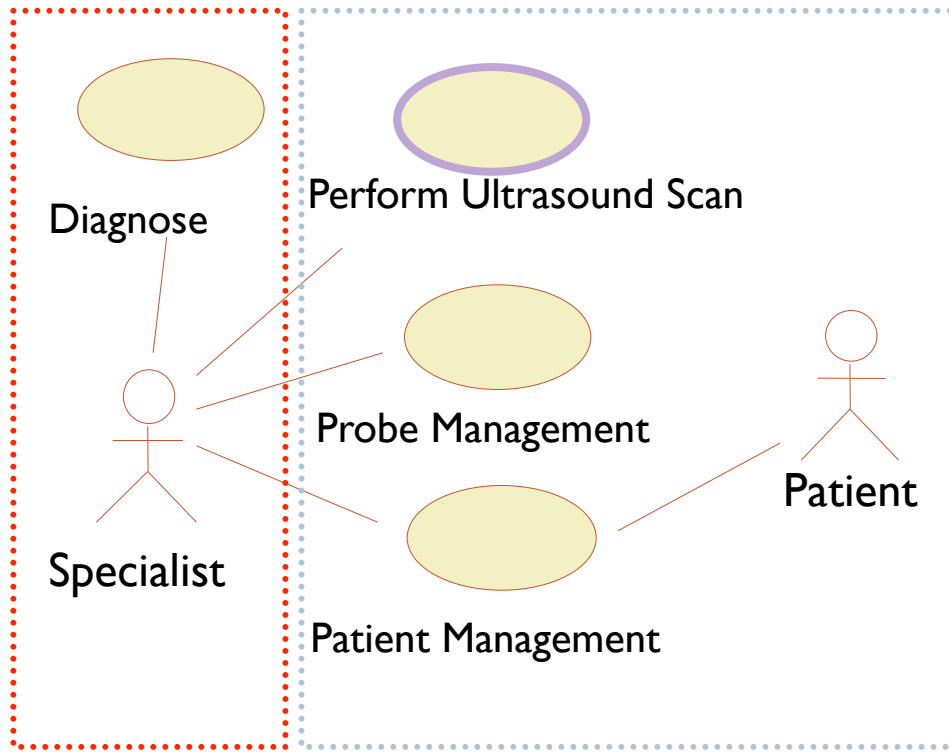
- ▶ **Function allocation and task analysis**
 - ▶ Determine distribution of work
 - ▶ Identify details of specified tasks (required knowledge, skills, attitudes)

➔ **Four steps with UML based development**

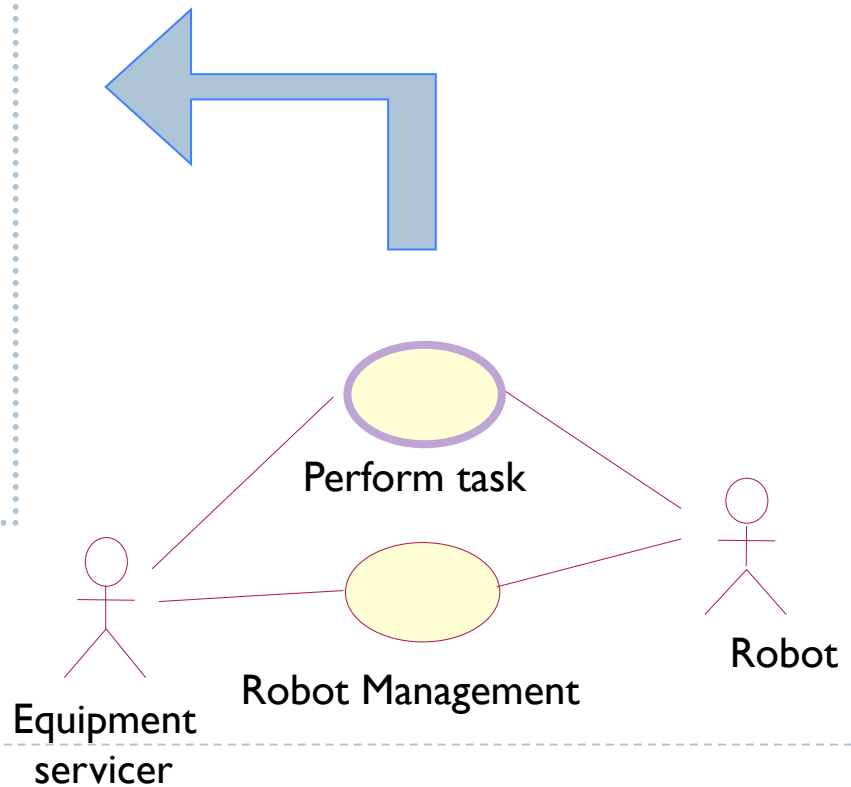
- A. General scenario
- B. Robot integration
- C. System definition
- D. Task description



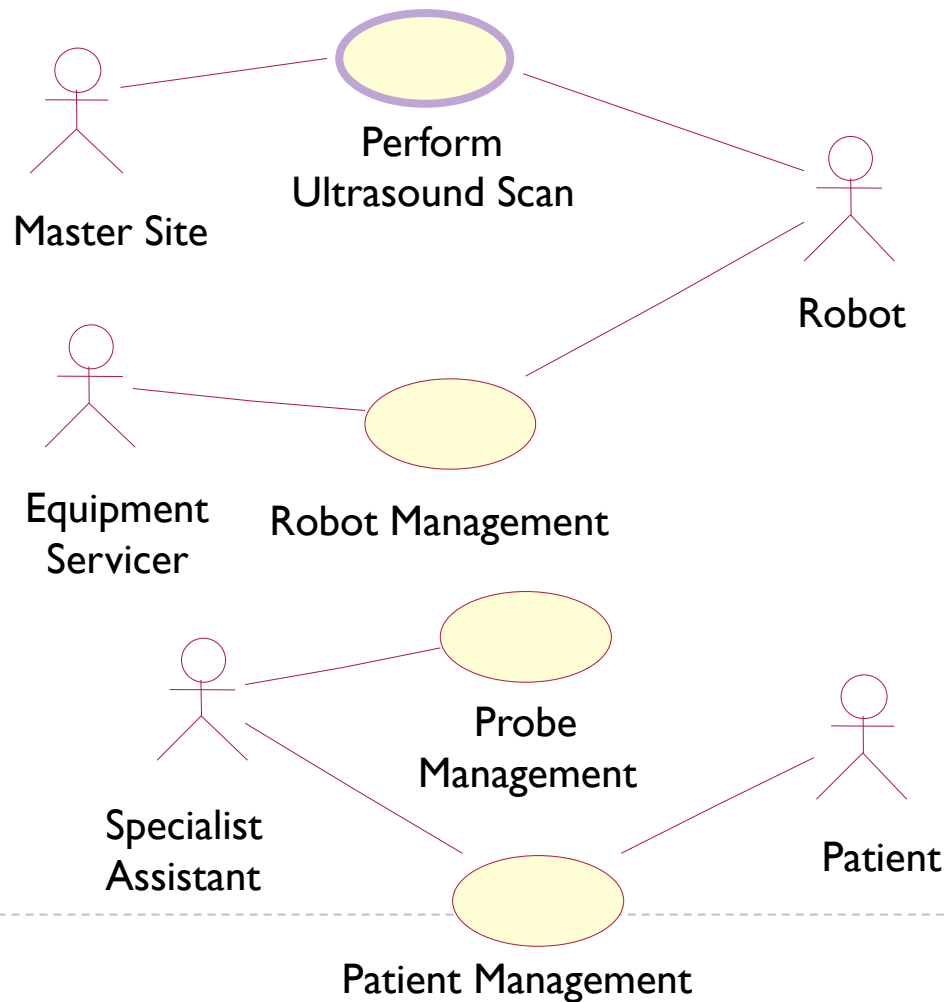
A. General scenario with business modeling



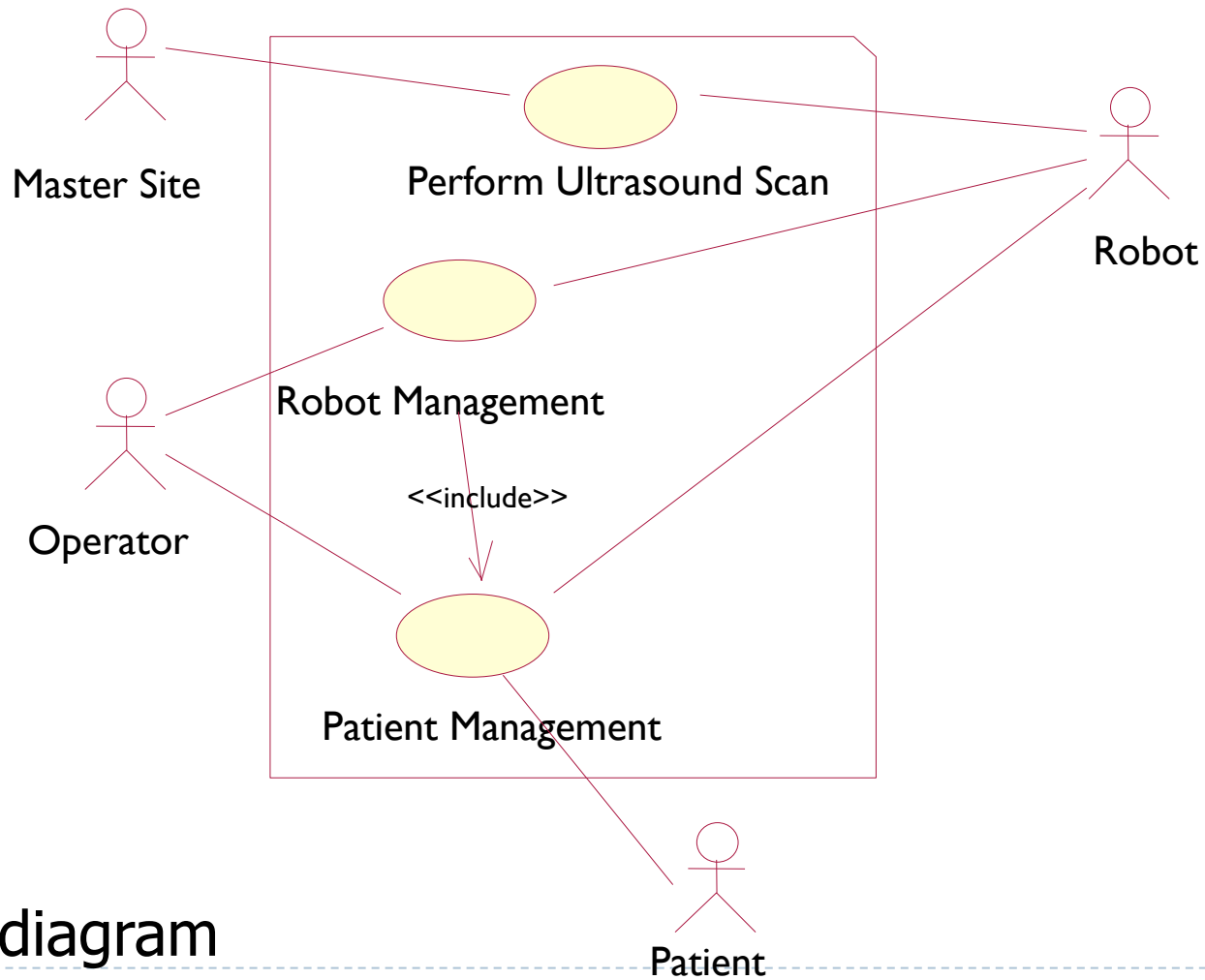
Use Case diagram



B. Robot integration



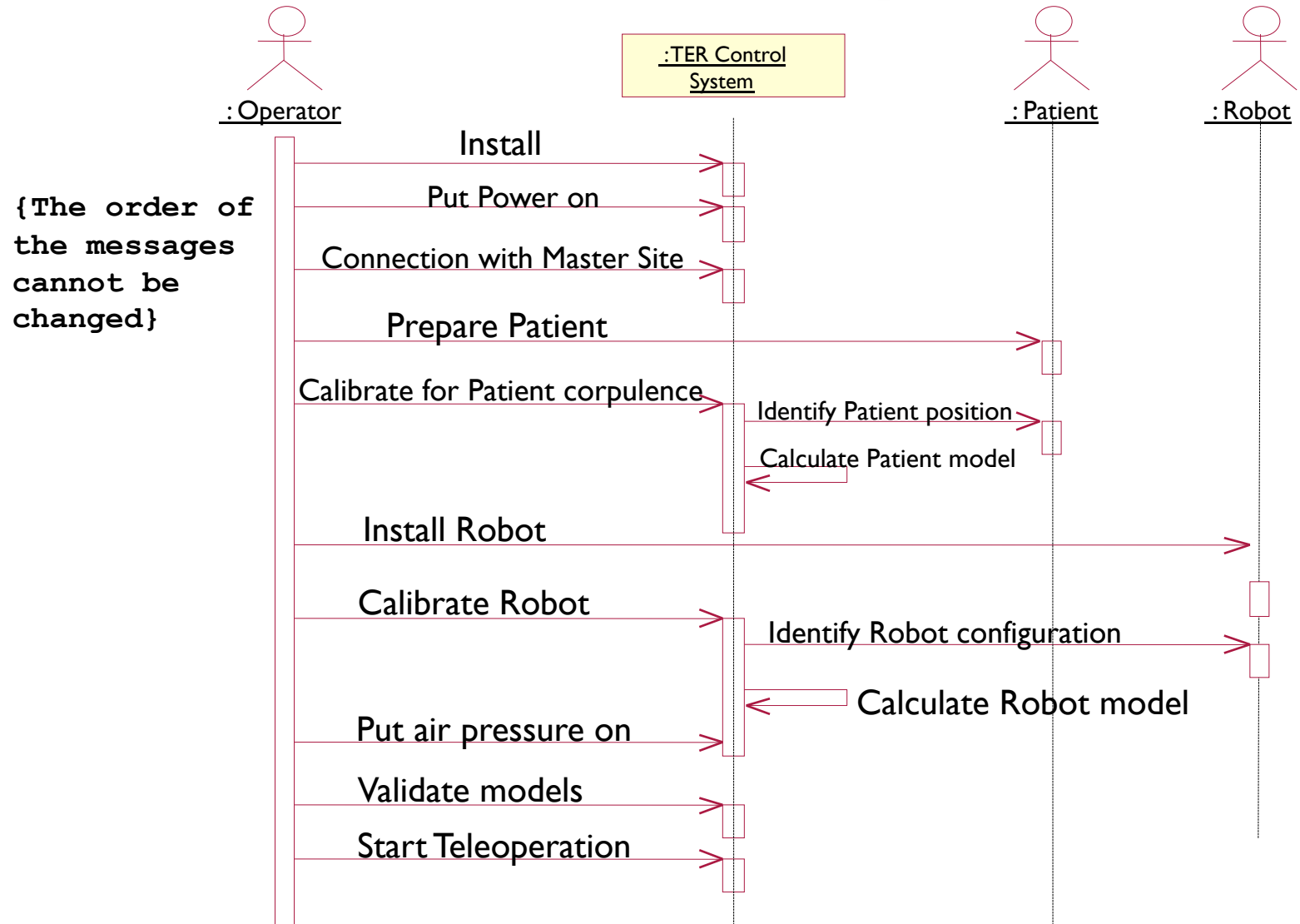
C. System definition



Use Case diagram



D.Task description



Sequence diagram




Preliminary Hazard Analysis

- ▶ At the very first step of the project
- ▶ Brainstorming / short meeting (max 2h)
- ▶ Same organisation as HAZOP (see chapter 4) but with a more simple worksheet
 - 1) Identify system hazards and sources
 - 2) Translate system hazards into high-level system safety design constraints.
 - 3) Assess hazards if required to do so.
 - 4) Establish the hazard log.



Example: System Hazards for Automated Train Doors

HAZARD	DESIGN CRITERION
Train starts with door open.	Train must not be capable of moving with any door open.
Door opens while train is in motion.	Doors must remain closed while train is in motion.
Door opens while improperly aligned with station platform.	Door must be capable of opening only after train is stopped and properly aligned with platform unless emergency exists (see below).
Door closes while someone is in doorway.	Door areas must be clear before door closing begins.
Door that closes on an obstruction does not reopen or reopened door does not reclose.	An obstructed door must reopen to permit removal of obstruction and then automatically reclose.
Doors cannot be opened for emergency evacuation.	Means must be provided to open doors anywhere when the train is stopped for emergency evacuation.



PHA worksheet example

Num.	Hazard	Source	Remarks	Recommendation	Who is in charge of application
		Hardware Software Human Environment Mechanical Electrical Etc.			

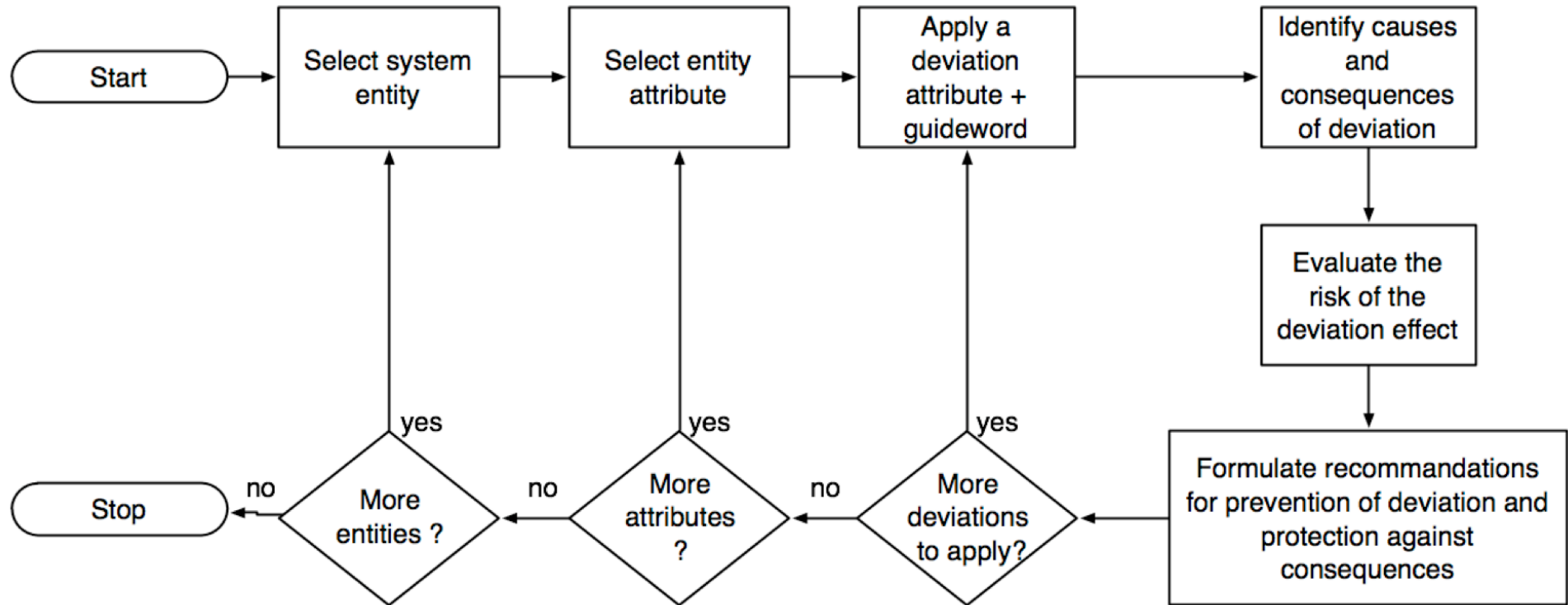


UML-HAZOP analysis

- 1) UML Use cases + sequence diagrams
- 2) Uses cases conditions
- 3) The HAZOP method is applied to:
 - 1) Each use case
 - 2) Each sequence diagram



HAZOP overview



WHAT is UML system entity ? And associated attributes ?

▶ **UML Entity :**

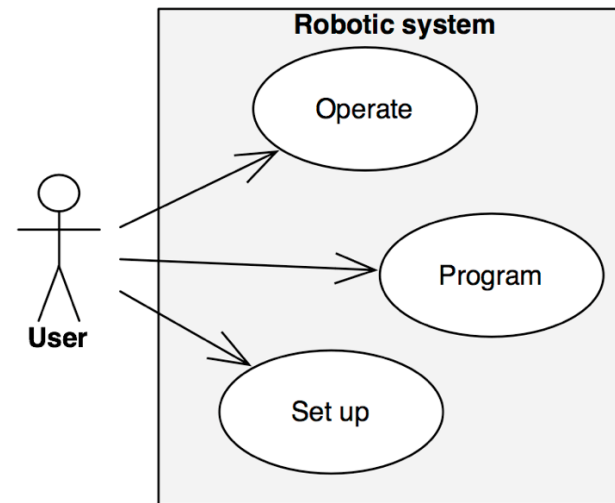
▶ Use cases

▶ Sequence diagrams



Use case attributes

Use case specification	
Use case name	The name of the use case provides a unique identifier
Abstract	Describes the interaction that occurs in the main scenario of the use case
Preconditions	Conditions that must be satisfied before the use case can be executed — they are part of the contract between the use case and the outside world
Postconditions	Conditions that must be satisfied after the use case has been completed successfully
Invariants	Conditions that must be fulfilled throughout the use case execution

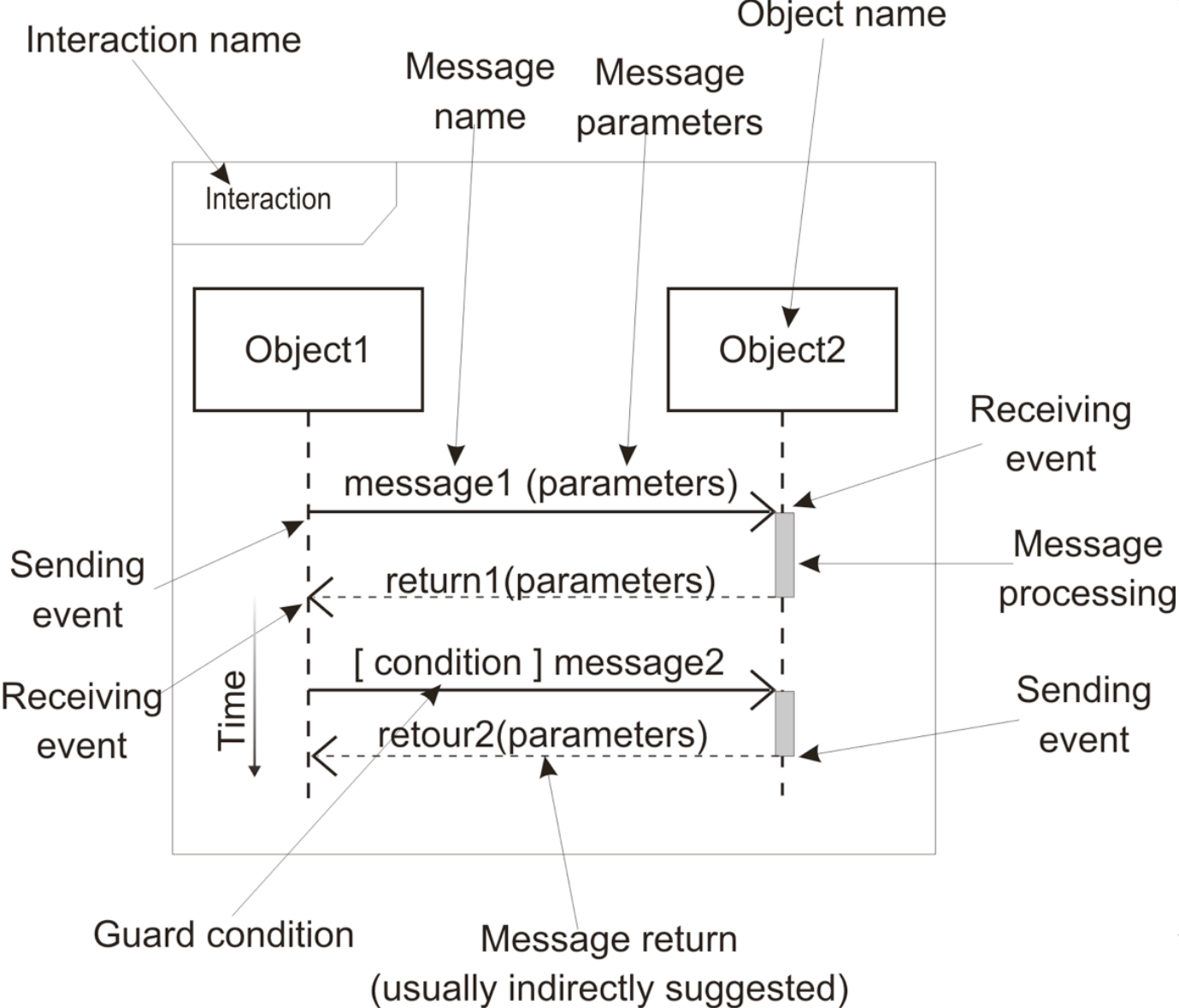


Use case guidewords

Entity = Use Case		
Attribute	Guideword	Interpretation
Preconditions / Postconditions / Invariants	No/none	The condition is not evaluated and can have any value
	Other than	The condition is evaluated true whereas it is false The condition is evaluated false whereas it is true
	As well as	The condition is correctly evaluated but other unexpected conditions are true
	Part of	The condition is partially evaluated Some conditions are missing
	Early	The condition is evaluated earlier than required (other condition(s) should be tested before) The condition is evaluated earlier than required for correct synchronization with the environment
	Late	The condition is evaluated later than required (condition(s) depending on this one should have already been tested) The condition is evaluated later than required for correct synchronization with the environment



Sequence diagram attributes



Sequence diagram guidewords

Entity = Sequence Diagram		
Attribute	Guideword	Interpretation
Predecessors / successors during interaction	No	Message is not sent
	Other than	Unexpected message is sent
	As well as	Message is sent as well as another message
	More than	Message sent more often than intended
	Less than	Message sent less often than intended
	Before	Message sent before intended
	After	Message sent after intended
	Part of	Only a part of a set of messages is sent
	Reverse	Reverse order of expected messages
Message timing	As well as	Message sent at correct time and also at incorrect time
	Early	Message sent earlier than intended time
	Later	Message sent later than intended time



Entity = Sequence Diagram		
Attribute	Guideword	Interpretation
...
Sender / receiver objects	No Other than As well as Reverse More Less	Message sent to but never received by intended object Message sent to wrong object Message sent to correct object and also an incorrect object Source and destination objects are reversed Message sent to more objects than intended Message sent to fewer objects than intended
Message condition	No/none Other than As well as Part of Late	The condition is not evaluated and can have any value (omission) The condition is evaluated true whereas it is false, or vice versa (commission) The condition is well evaluated but other unexpected conditions are true Only a part of condition is correctly evaluated The condition is evaluated later than required (other dependent condition(s) have been tested before) The condition is evaluated later than correct synchronization with the environment
Message parameters / return parameters	No/None More Less As Well As Part of Other than	Expected parameters are never set / returned Parameters values are higher than intended Parameters values are lower than intended Parameters are also transmitted with unexpected ones Only some parameters are transmitted Some parameters are missing Parameter type / number are different from those expected by the receiver

HAZOP worksheet

Attribute	Guideword	Deviation	Use Case Effect	Real World Effect	Severity	Possible Causes	Integrity Level Requirements	New Safety Requirements	Remarks	Hazard Number



Example : PHRIENDS case study

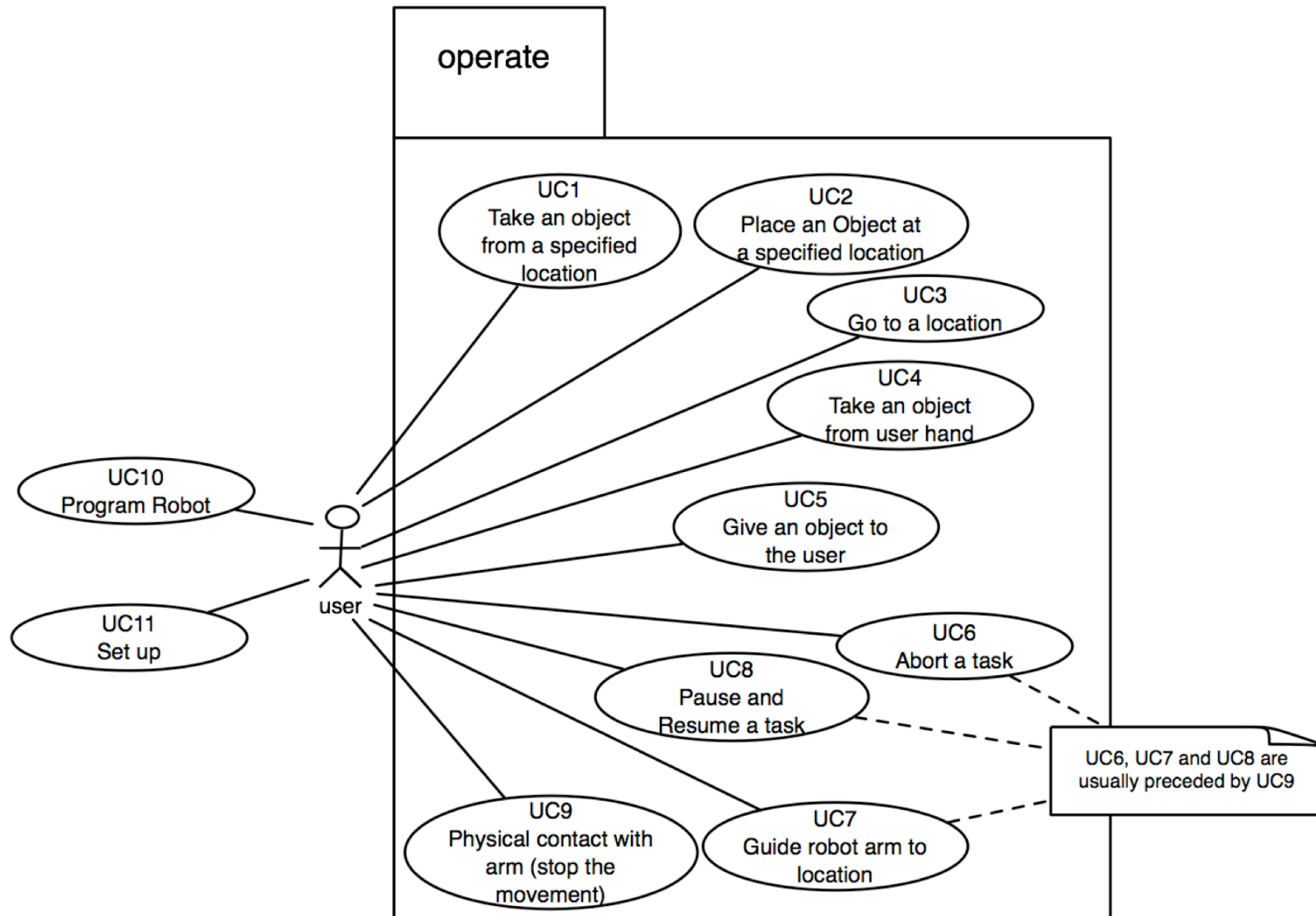
- ▶ Take an object from a specified location
- ▶ Place an object at a specified location
- ▶ Go to a location (holding or not holding an object)
- ▶ Take an object from the user's hand
- ▶ Give an object to the user

A second group of use cases applies to when the user can interrupt the previous actions to:


- ▶ Abort a task
- ▶ Guide the robot arm to a location
- ▶ Pause and resume a task
- ▶ Physical interaction



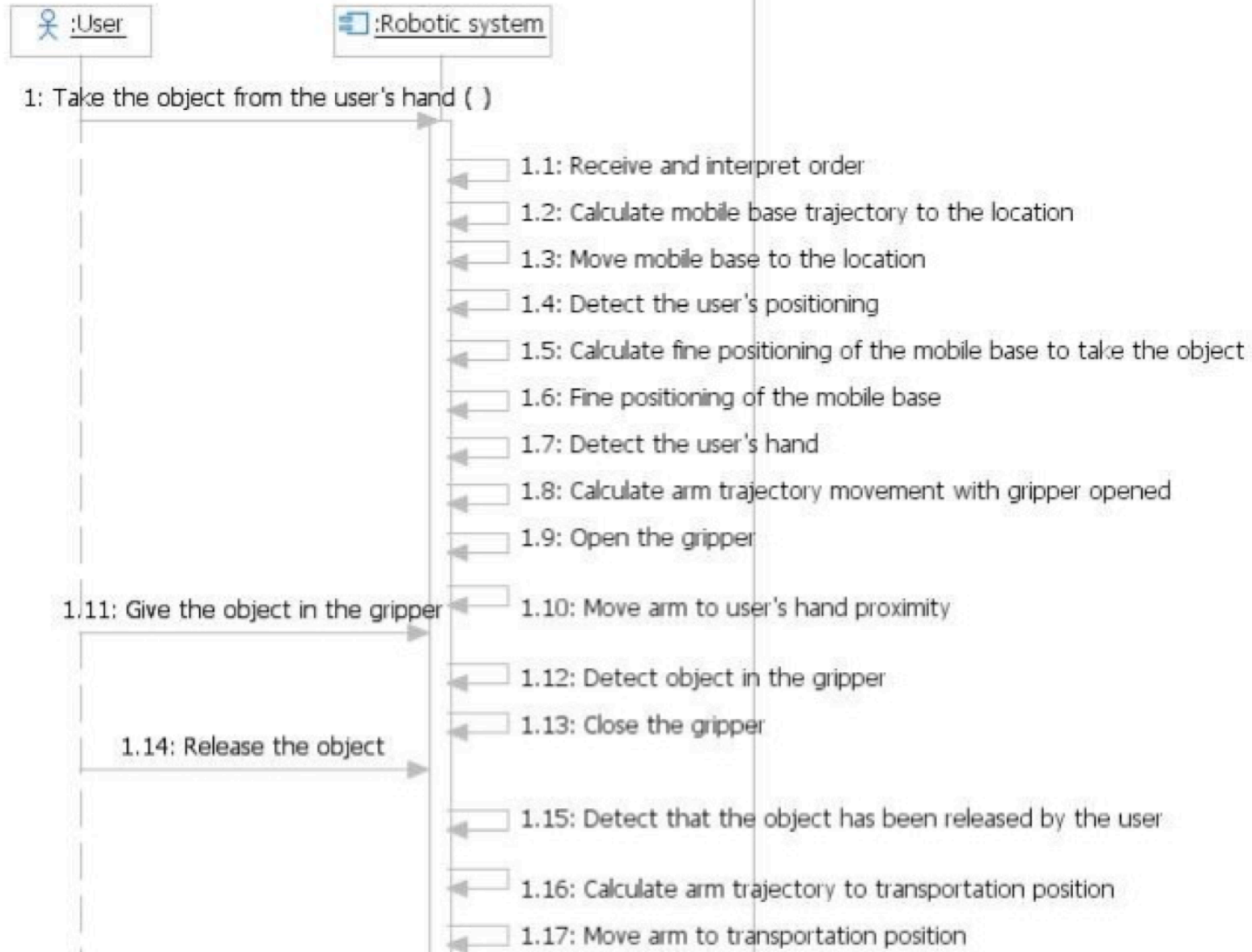
Use cases



Use case name	UC4. Take an object from the user's hand
Abstract	The user orders the robot to take an object from his hand
Precondition	_ No object in the gripper _ Location reachable _ Object can be taken
Postcondition	_ Robot base is stopped _ Object in the gripper _ Robot arm is in transportation position
Invariant	None



SD4 Take the object from the user's hand



Project : PHRIENDS
 HAZOP number : UC4
 Entity : Use Case 4 (UC4) "Take an object from the user's hand"

Date: June-01-2008
 Prepared by: Ofaina Taofifenua
 Revised by: Jérémie Guiochet
 Approved by :

Attribute	Guideword	Deviation	Use Case Effect	Real World Effect	Severity	Possible Causes	Integrity Level Requirements	New Safety Requirements	Remarks	Hazard Number
No object in the gripper (precondition)	Other than	No object detected in the gripper whereas there is one	The robot will move with an object in the gripper and when taking another, the former will fall (if user does not react)	Physical impact between object and environment (including user)	Severe	H/W failure (object detection sensor) S/W failure	H/W and S/W for object detection should be SIL2	Envisage redundant sensor system (vision, force, etc.) Provide means to force the robot to release the object (for a user to react to the incoherent situation)		3, 5, 19
	Other than	Object detected in the gripper whereas there is none	The robot will not execute the task	Break in the execution sequence Misunderstanding between human and robot	None	H/W failure (object detection sensor) S/W failure	None	None		
Robot base is stopped (postcondition)	As well as	Robot base is stopped as well as it is not in a secure location	The robot is stationary in an insecure place, e.g., behind a door or a place hindering the user	Physical impact between robot and environment (including user)	Moderate	H/W failure (motion function) S/W failure Insufficient specification	H/W for motion should be SIL1 S/W for base motion and navigation should be SIL1	Specify secure locations that the robot will go to if there are no more tasks to execute When robot is stopped collision detection should remain activated		20

Project : PHRIENDS
 HAZOP number : UC4/SD4
 Entity : Sequence Diagram 4 (sd4) "Take an object from the user's hand"

Date: June-01-2008
 Prepared by: Ofaina Taofifenua
 Revised by: Jérémie Guiochet
 Approved by:

Element	Guideword	Deviation	Use Case Effect	Real World Effect	Severity	Possible Causes	Integrity level Requirements	New Safety Requirements	Remarks	Hazard Number
H1 Take the object from the user's hand (at location)	5. More / Less / Other than	Parameter "Location" is incorrect	The robot does not detect any user at its location	Break in the execution sequence	None	Human error Insufficient specification	None	None		
R2 Receive and interpret order	1. More than / as well as	The robot receives several different orders	Wrong order taken into account	Wrong task, bad synchronization between robot and user, could result in collision	Moderate	H/W for order reception failure Human error	H/W for order reception should be SIL1	User education and training Define a protocol for communication between user and robot (e.g. acknowledgment messages, user can check interpretation of the order)	Means for communication between robot and user needs to be defined for the PHRIENDS use case (speech, graphical HMI, vision, etc.)	
	5. More / Less / Other than	The robot incorrectly interprets order: the robot computes a trajectory for the mobile base to a wrong location	The robot goes to the wrong location and is stationary in an insecure place, e.g., behind a door or a place hindering the user	Physical impact between robot and environment (including user)	Moderate	H/W failure (altered message) S/W failure (error in processing)	H/W for transmission of the user order to system should be SIL1 S/W for trajectory computation should be SIL1	Define a protocol for communication between user and robot (e.g. acknowledgment messages, user can check interpretation of the order)		15, 20



Some results of the application of the method

- ▶ **PHRIEND project safety analysis artefacts**
 - ▶ Hazard list
 - ▶ Recommendation list
 - ▶ Integrity level requirements list
 - ▶ Top events list => used for fault tree analysis



MIRAS example

Multimodal Interactive Robot for Assistance in Strolling

- ▶ The MIRAS project's aim is to develop an assistive robot for mobility capable of health state monitoring. It is designed to be used in elderly care centers by elderly people suffering from gait and orientation problems.
- ▶ The purpose is to offer more freedom and time to staff personnel, by releasing them from basic assistance tasks in mobility (such as rising from a chair and/or going to the bathroom etc.), so that they can focus on other more demanding tasks.
- ▶ It integrates the following functionalities, enabled by multimodal interaction:
 - ▶ Transparent control of the robot by the user when walking.
 - ▶ Dynamic stabilization of the user if a fall or inappropriate motion is detected, using force sensors combined with visual estimation of posture.
 - ▶ Adaptation of the robot's behavior to a detection of user overstrain (physiological state monitoring + changes in gait patterns).
 - ▶ "hello" function to call the robot (in dock position). The robot is able to autonomously move to the patient position. (a "bye bye" function should also exist...)

