Industrial Embedded Systems
- Design for Harsh Environment -

Dr. Alexander Walsch
alexander.walsch@ge.com

WS 2011/12

Technical University Munich (TUM)

Agenda

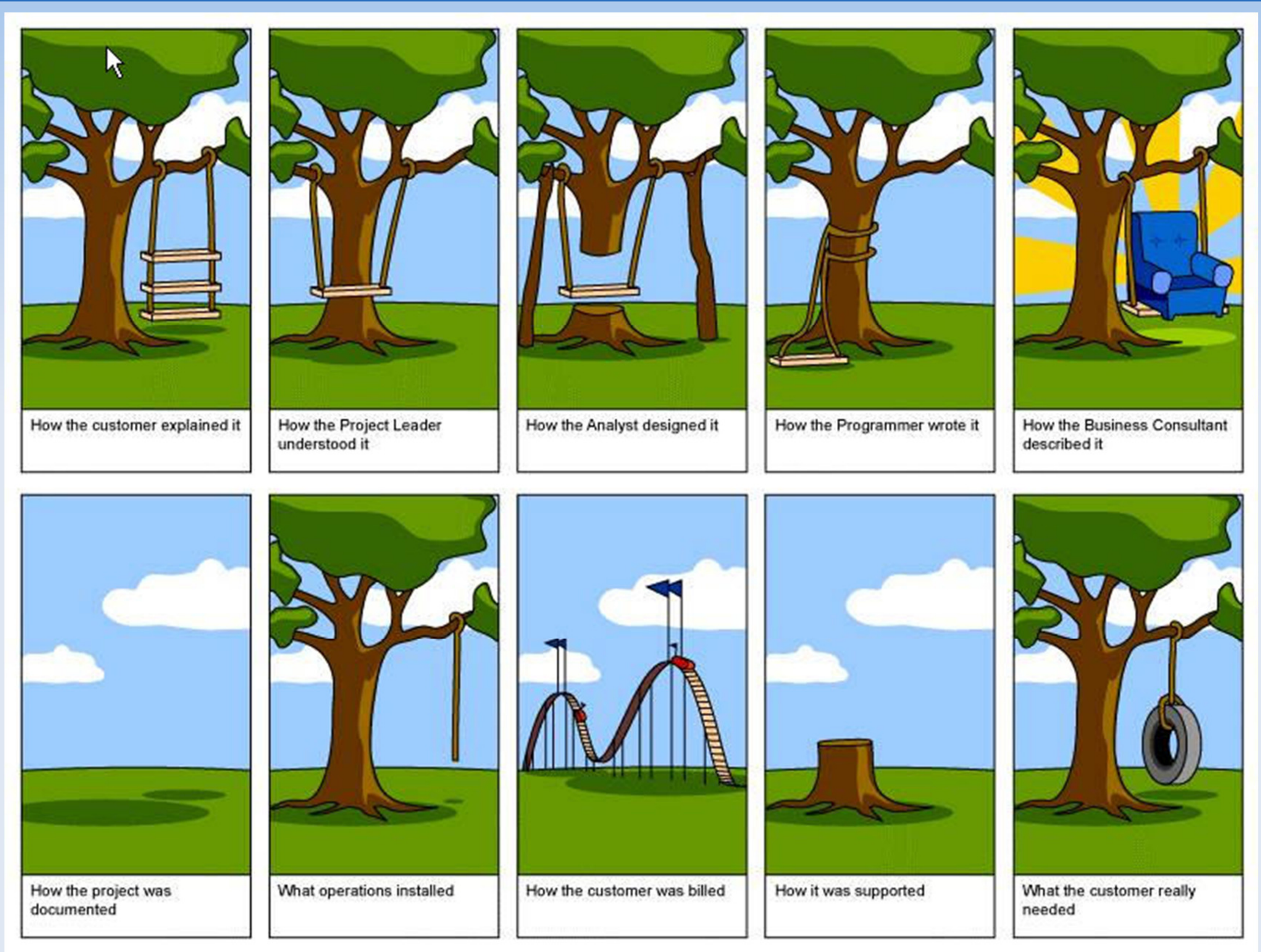
Today:

Requirements Analysis
Quality Metrics

Recap:

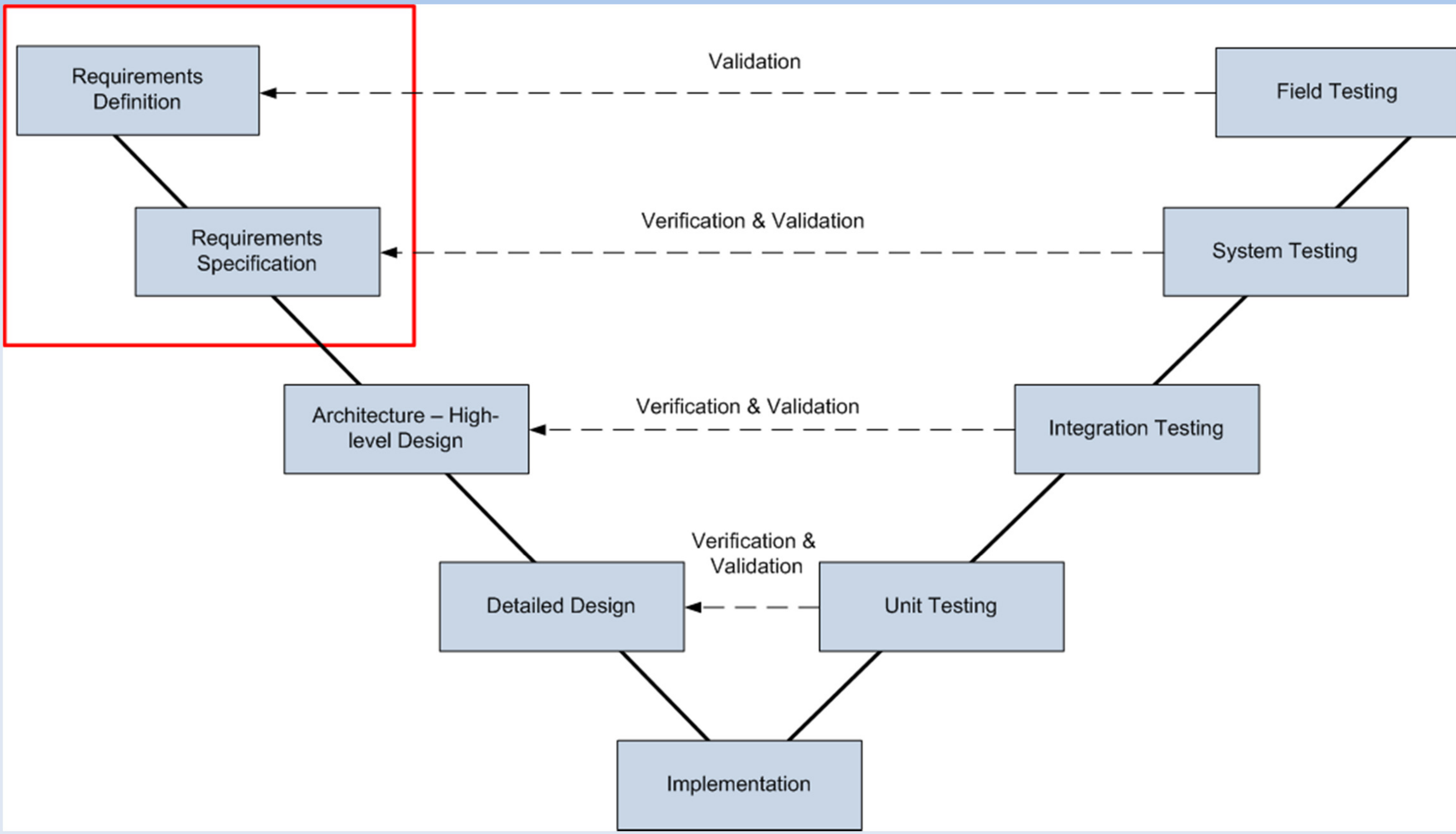
- Small footprint system (mechanical, electrical, and information view)
- Harsh environment (temperature, shock, vibration, ...)

Motivation



Source: http://www.bowdoin.edu/~disrael/what-the-customer-really-needed/what-the-customer-really-needed.jpg

V-Model


Requirements Definition
Validation
Field Testing
Requirements Specification
Verification & Validation
System Testing
Architecture – High-level Design
Verification & Validation
Integration Testing
Verification & Validation
Detailed Design
Unit Testing
Implementation

The Big Picture

The requirements analysis phase of system development is about getting all information together and about showing scope, usage, and constraints of the proposed system. Wrong (e.g. missing, contradicting) information will make us fail at a very cost intensive level.

V&V

<u>**Verification:**</u>
Test against a specification (owned by technology). This is usually an internal activity without customer involvement. Its intent is to show that <u>the specification as compiled by technology has been implemented correctly</u> – Have we done it right?

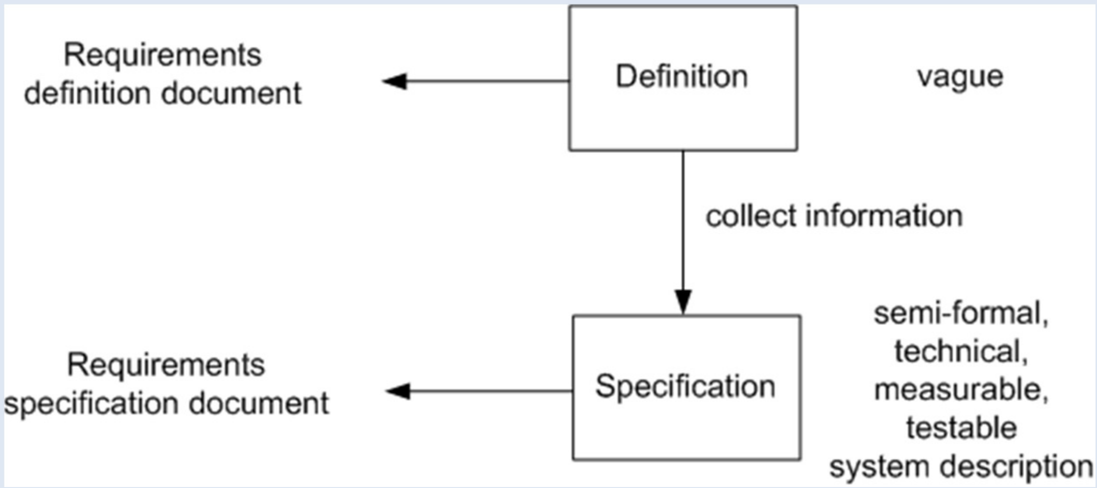
<u>**Validation:**</u>
Test against a requirement definition (owned by the customer). This is usually an external activity with customer involvement (pilot). Its intent is to show that <u>the system meets the customer's expectations</u> – Have we done the right thing?

Requirements

Requirement:

Features of a system or system function used to fulfill the system purpose.

Requirements Analysis:



Functional Requirements

Functional Requirement:

Core system function used to fulfill the system purpose –

What must the system do?

Includes

- Inputs and associated outputs (valid inputs, invalid inputs, warnings, errors)
- Formats for I/O
- User Interfaces and different roles (technician, customer, …)
- States of the system (operational, error)
- Failure modes

Functional Requirements Capture

Look at system as black-box

- Look at what it interacts with
  Other systems, devices, users (identified as user-roles)
  UML: use case diagram
- Look at how it interacts
  Data flow, control
  UML: sequence diagram
- Traditional, basic form: textual, according to some template or standard form (text document, unique ID)
- Model-based form: use case and sequence diagrams (UML) + textual description

Examples

“The system shall connect to a pressure sensor with 4 – 20 mA interface.”

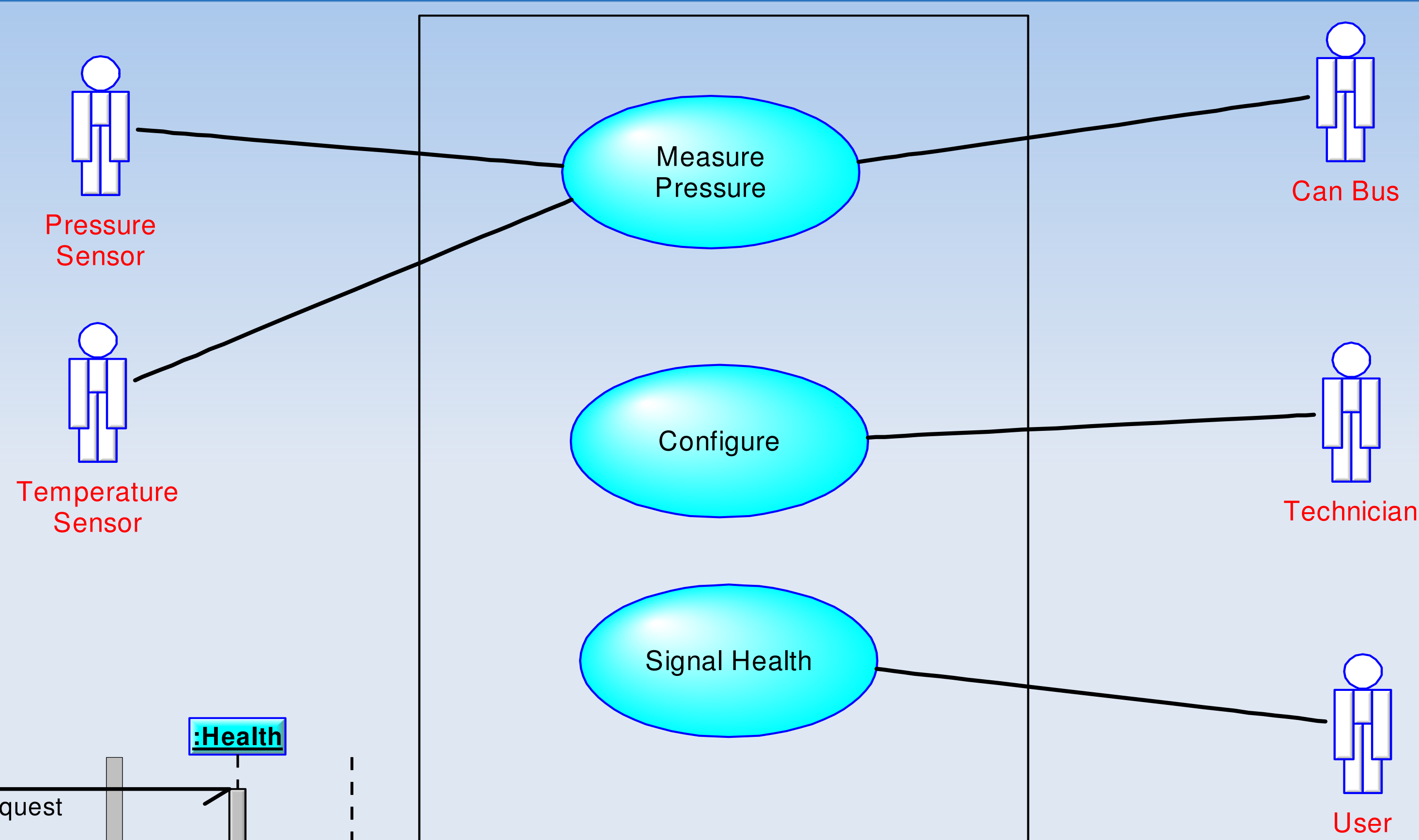
“The system shall not supply power to the pressure sensor.”

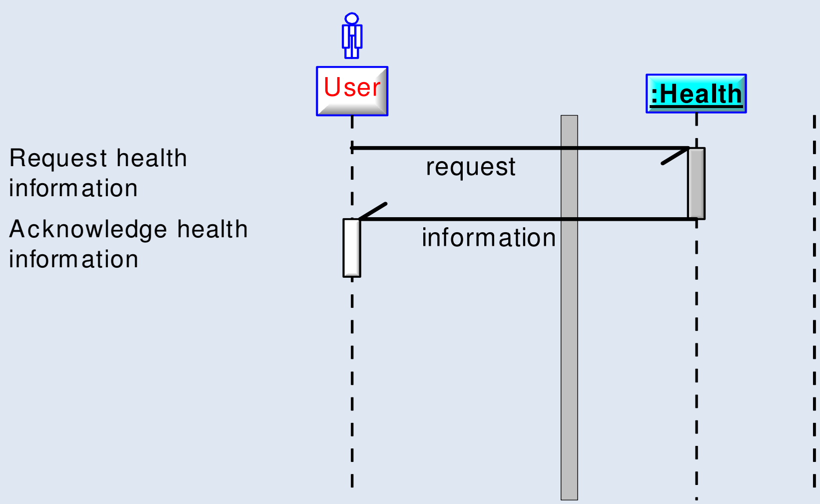
“The system shall indicate a violation of input range by an “out of range” error message according to [*std. xyz.*] if the current input is less than 5 mA or more than 19 mA.”

“All pressure readings shall be communicated via the CAN bus.”

“All pressure readings shall be communicated according to [*std. xyz*]”

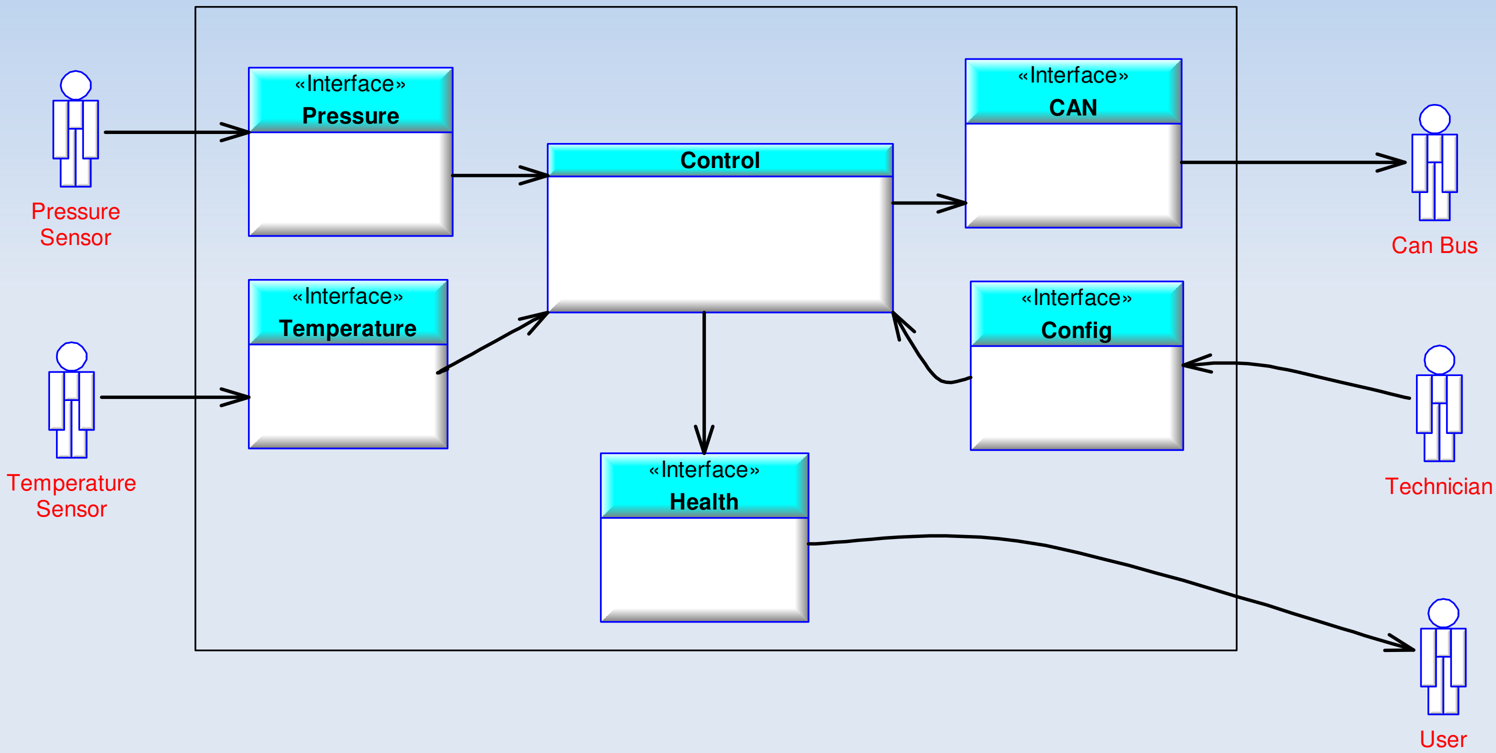
MBD Example

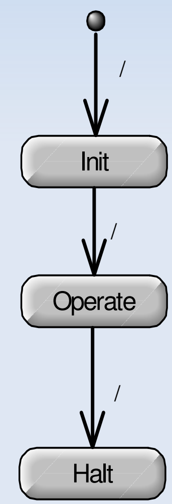




- Functional View
- Actors = external users or devices
- Use cases = functions

MBD Example Ctd.





Non-Functional Requirements

Non-Functional Requirement:

Constraints on implementation – How should the system be?

Includes

- Global constraints that influence system as a whole (shock, vibration, temperature,…)
- Performance (response time, repeatability, utilization, accuracy)
- The “-ilities” (reliability, availability, safety, security, maintainability, testability, …)
- Other quality (ease of configuration and installation, …)

Non-Functional Requirements Capture

Look at system as black-box

- Look at real-time aspects
  e.g. response time
- Data quality
  accuracy, precision, sampling rate
- Refine functional requirements – make more specific and testable
- Look at comparable systems (prior art, competitors)
- Look at new laws or regulations (e.g. disasters – Fukushima, Deepwater Horizon)

Examples

“Pressure samples shall be taken every 1s.”
“The response time for pressure measurement shall be less than 10ms.”
“Reliability: 1000 FIT”
“The measurement shall have an accuracy of 2%.”
“The measurement shall be repeatable with a precision not less than 0.5%.”
“The system shall meet the safety criteria according to [std.].”

Requirements Analysis

How do we get all these requirements?

- Involves technical staff working with customers or users to find out about the application domain (field technicians), the services that the system should provide and the system's operational constraints.
- May involve end-users, our customers, managers, engineers involved in prior development and/or maintenance, domain experts, certification bodies, etc. These are called stakeholders.

Challenges in Requirements Analysis

- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terminology – maybe not precise.
- Different stakeholders may have conflicting requirements.
- Political factors may influence the system requirements (e.g. disasters).
- The requirements change during the analysis process.
- Some requirements might be common sense and not explicitly mentioned.

Feasibility Study

<u>Feasibility Study:</u>

A feasibility study decides whether or not the proposed system is worthwhile. Usually a study on the most risky elements of a new development.

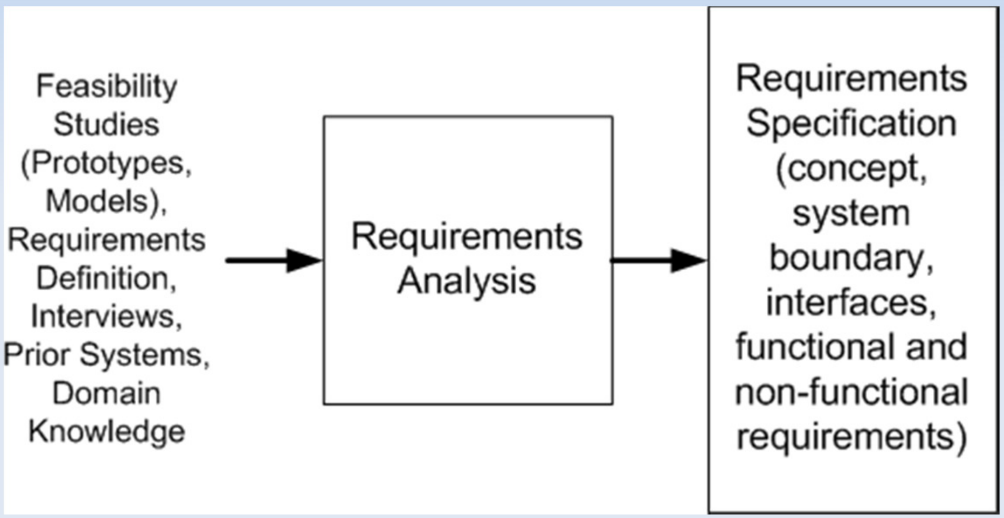
A short focused study that checks

- If the proposed system can be engineered using current technology and within budget (technical and economic feasibility);
- If the proposed system can be integrated with other systems that are used (interoperability).
- If the proposed system can meet the requirements (especially non-functional like reliability, e.g.)

A final Look at Requirements

- Validity. Does the system provide the functions which the customer expects?
- Consistency. Are there any requirements conflicts?
- Completeness. Are all functions required by the customer included? Are more functions included?
- Realism. Can the requirements be implemented given available budget and technology -> feasibility?
- Verifiability. Can the requirements be tested?

Requirements Analysis Summary



Typical document layout:

Requirement Specification

1. Objective
2. System Description (boundary, interfaces, major components)
3. Functional Requirements
4. Non-functional Requirements
5. Performance
6. Mechanical Constraints
7. Environmental Constraints
8. RAMS

All requirements get numbers which allow forward and backwards tracing.

Traceability

Traceability:

Traceability is concerned with the relationships between requirements, their sources and their design implications. Traceability can be a requirement itself.

- Source traceability
  - Links from requirements to stakeholders who proposed these requirements;
- Requirements traceability
  - Links between dependent requirements;
- Design traceability
  - Links from the requirements to the design;

RAMS

In practice there are four non-functional requirements that get a special focus – reliability, availability, maintainability, and safety (RAMS). It is important to understand their definition and metrics to furnish requirements specifications. They do influence the architecture.

Failure

Failure:
Failure is defined as deviation from the specification. The designed function can not be executed anymore as specified.

Failure Mode:
A component (or system) can fail in various ways. In our analysis we pick the failure mode that leads to the failure we investigate.

Failure Rate:
Each failure mode comes with a certain failure rate (failures/time). We pick the failure rate of the failure mode that leads to the failure under investigation.

Example

Function:

Pressure is measured and the reading transmitted using a 4 – 20 mA interface.

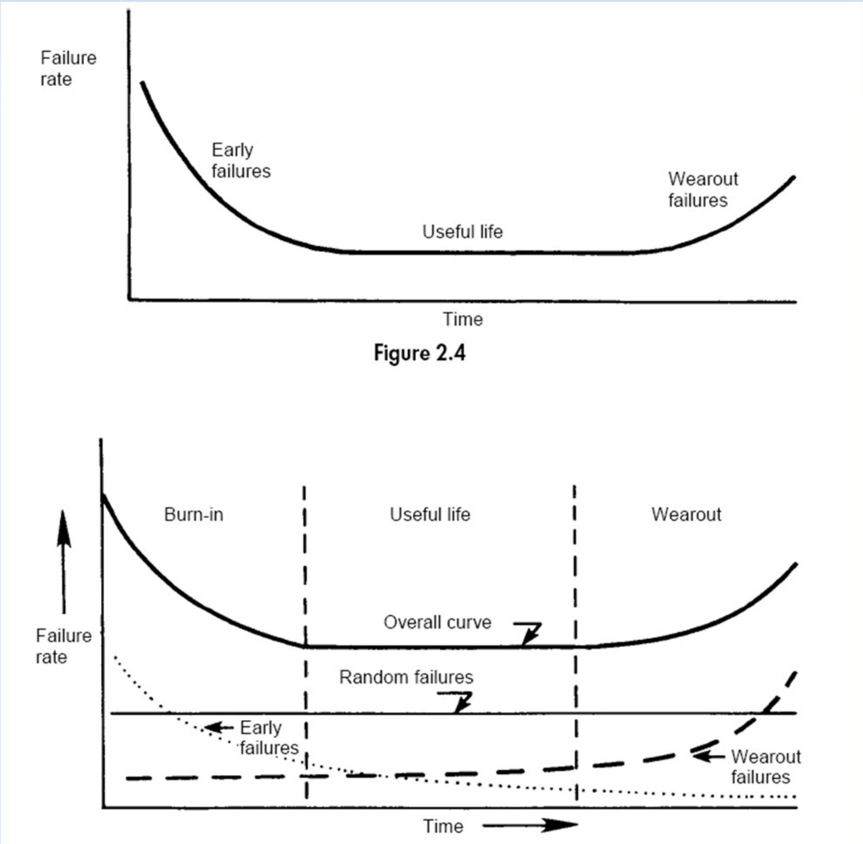
The following failure modes and rates are known. What failure modes do influence our design?

Failure Mode	Failure rate ($10^{-6}h^{-1}$)
4 – 20 mA current signal stuck fail	5
4 – 20 mA current signal low fail	3
Sensor head fail	4
0 – 10 V voltage signal stuck fail	6
Other	3

Failure Rate

Failure Rate:

A time dependent measure of #failures/time. Commonly only random failures are considered. The symbol for failure rate is $\lambda(t)$.



Source:
Smith: Reliability, Maintainability and Risk

Reliability

<u>Reliability:</u>
Reliability of a system is defined to be the probability that a given system will perform a required function under specified conditions for a specified period of time.

"probability of non-failure (survival) in a given period"

Reliability of a system is modeled as:

$\mathrm{R(t)} = e^{-\lambda t}$ if the failure rate λ is constant.

λ is often expressed as failures per 10^6 hours or FIT (failures per 10^9 hours).
If "λt" small then $R(t) = 1 - \lambda t$

MTBF

MTBF:

Mean Time Between Failures (MTBF): is the average time a system will run between failures. The MTBF is usually expressed in hours.

$\Theta = \int_0^\infty R(t)dt = \int_0^\infty e^{-\lambda t}\,\mathrm{dt} = \lambda^{-1}$, λ = const.

The observed MTBF:

$\widehat{\Theta}$ = T/k; T = total time, k = failed items (total N)

MDT and MTTR

MDT:

Mean Down Time (MDT) is the average time a system is in a failed state and can not execute its function. MTBF can be understood as the mean up time.

MTTR:

Mean Time to Repair (MTTR) is overlapping with MDT. Used for maintenance calculations. It can be visualized as the average time it takes (a technician) to repair the system such that it is up again.

Availability

Availability:
Availability is the probability that a system is functioning at any time during its scheduled working period.

$$A = \frac{up\ time}{total\ time} = \frac{up\ time}{up\ time + down\ time} = \frac{MTBF}{MTBF + MDT}$$

similar:

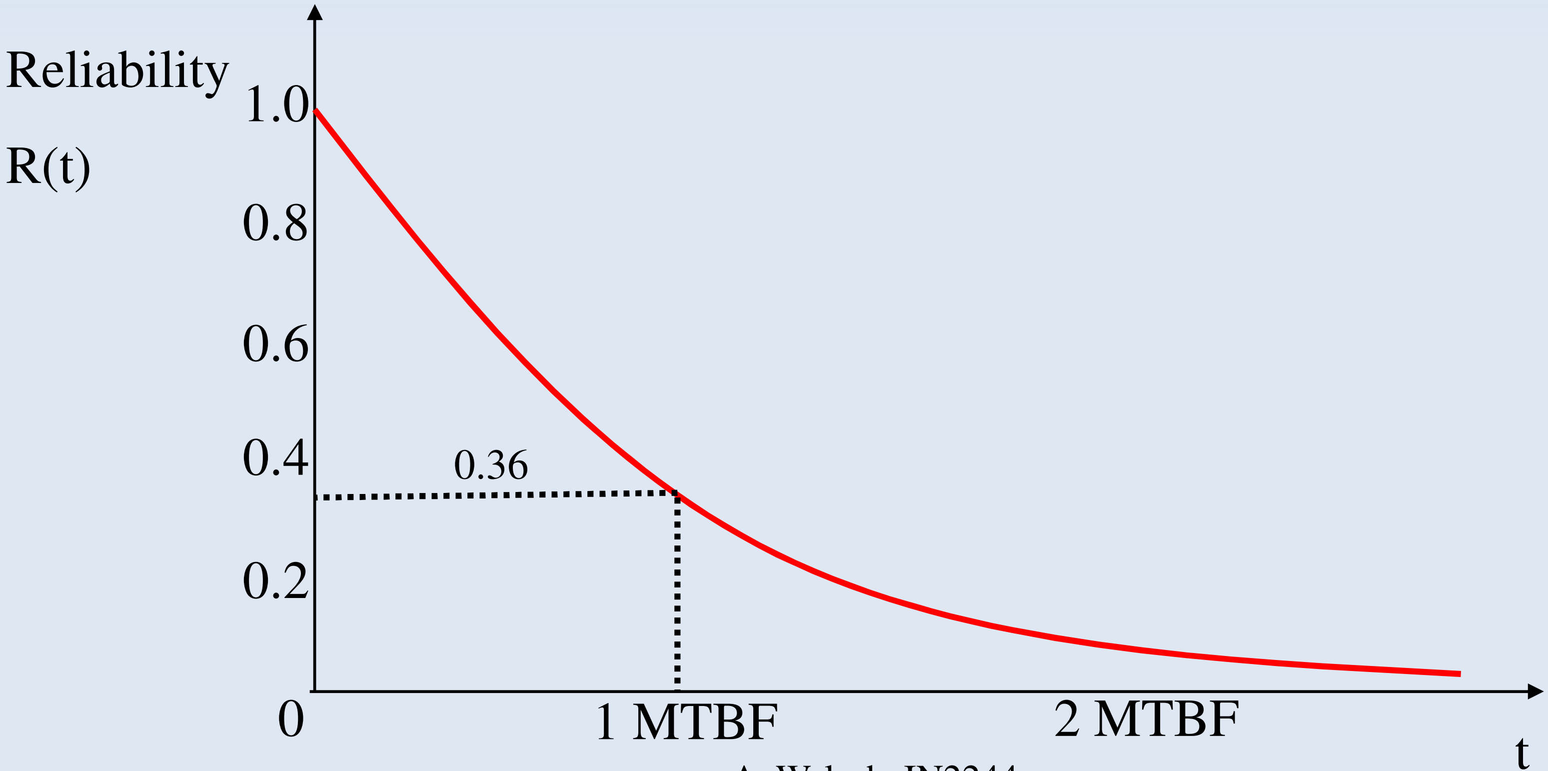
calculation of unavailability (PFD)

Relation between Reliability and MTBF

$$R(t) = e^{-\lambda t} = e^{-t/\Theta}$$

$t = \Theta$ -> $R = e^{-1} \approx 0.37$

$t = 2\Theta$ -> $R = e^{-2} \approx 0.14$



Example

A system (S) has 10 components (C) with a failure rate of 5 per 10^6 hours each. Calculate λ and MTBF.

$\lambda_C = 5 * 10^{-6}$ failures/hour
$\lambda_S = 10 * 5 * 10^{-6}$ failures/hour $= 5 * 10^{-5}$ failures/hour

$$\Theta = \frac{1}{\lambda_s} = 20000\text{h}$$

Example

$\lambda = 10^{-6}$ failures/hour ; MDT = 10h

Unavailability = ?

$$\bar{A} = \frac{down\ time}{total\ time} = \frac{MDT}{MTBF+MDT} \approx \lambda * MDT$$

-> $\bar{A} = 10^{-5}$

Reliability in Product Descriptions

Source:
Rosemount

Literature

IEC 61709:
Parts count method.


IEC 61709

Edition 2.0 2011-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Electric components – Reliability – Reference conditions for failure rates and stress models for conversion

Composants électriques – Fiabilité – Conditions de référence pour les taux de défaillance et modèles de contraintes pour la conversion

XD

IEC 62380:
Parts stress method.

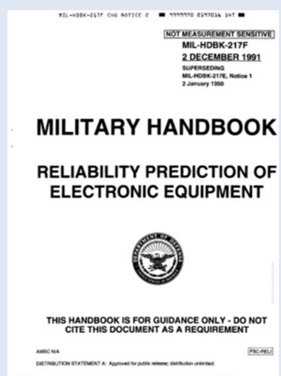

TECHNICAL REPORT

IEC TR 62380

Reliability data handbook – Universal model for reliability prediction of electronics components, PCBs and equipment

XC

MIL-HDB-217:
Parts stress method.


NOT MEASUREMENT SENSITIVE

MIL-HDBK-217F

2 DECEMBER 1991

SUPERSEDING

MIL-HDBK-217E, Notice 1

2 January 1990

MILITARY HANDBOOK

RELIABILITY PREDICTION OF ELECTRONIC EQUIPMENT

THIS HANDBOOK IS FOR GUIDANCE ONLY - DO NOT CITE THIS DOCUMENT AS A REQUIREMENT

The Bernoulli Experiment

Definition:

We have a total number of n identical systems. For each system only two states are defined: "functioning" or "has failed". Both states have a certain probability assigned. The Bernoulli experiment gives us the probability of finding k (out of n) systems in a functioning state.

We state:

P(functioning) = 1 – P (failed);
P(functioning) = p; P (failed) = q

The Bernoulli Experiment Ctd.

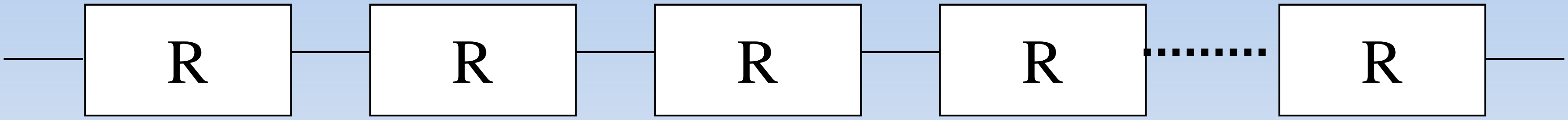
The probability of k functioning systems out of n total is

P(n,p,k) = $\binom{n}{k}\ p^k\ q^{n-k}$

Now we need the probability that a system is functioning -> reliability ("probability of survival")

P(n,p,k) = $\binom{n}{k}\ R^k\ (1-R)^{n-k}$ is the probability of having k functioning systems in an assembly of n total.

Series Reliability Calculation



All n systems above need to work such that the series assembly is functioning.

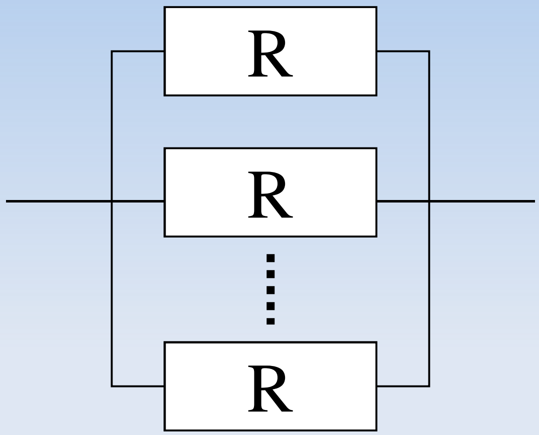
The probability of having n functioning systems out of n total is

$$P(n,n,k) = \binom{n}{n} R^n (1-R)^{n-n} = R^n$$

Parallel Reliability Calculation - Full Active Redundancy -

At least 1 system needs to be functioning in full active redundancy configuration.

Therefore, the assembly is working if n or (n-1) or ... or 1 component work.



<u>n=2</u>: 2 or 1 component must be functioning.

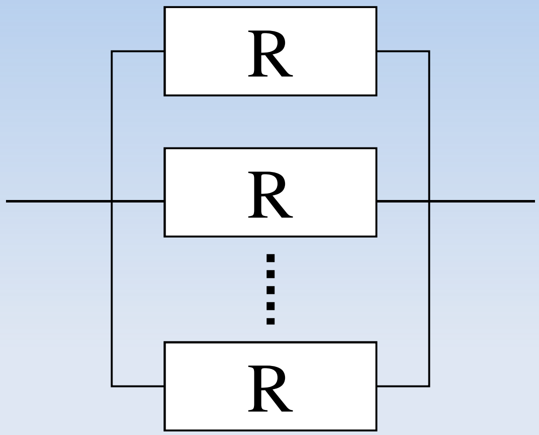
$$R_S = \binom{2}{2} R^2 (1-R)^0 + \binom{2}{1} R^1 (1-R)^1 = 2R - R^2$$

<u>n=n</u>:

$$R_S = \binom{n}{n} R^n (1-R)^0 + \ldots + \binom{n}{1} R(1-R)^n = 1-(1-R)^n$$

Parallel Reliability Calculation
- Partial Active Redundancy -

At least m systems need to be functioning in partial active redundancy configuration.



Therefore, the assembly is working if n or (n-1) or ... or m systems work.

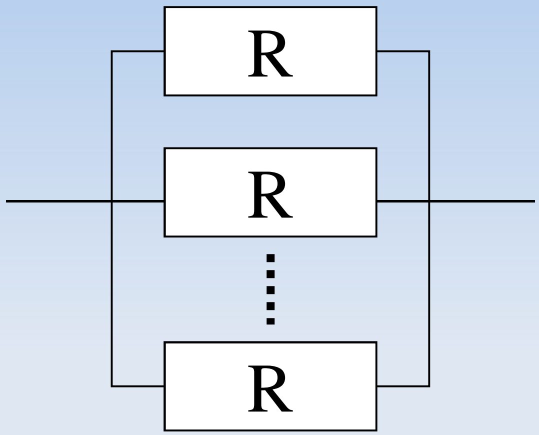
n=3: m = 2 (2oo3)

$$R_S = \binom{3}{3} R^3 (1-R)^0 + \binom{3}{2} R^2 (1-R)^1 = 3R^2 - 2R^3$$

Parallel Reliability Calculation - Partial Active Redundancy – Ctd.

At least m systems need to be functioning in partial active redundancy configuration.

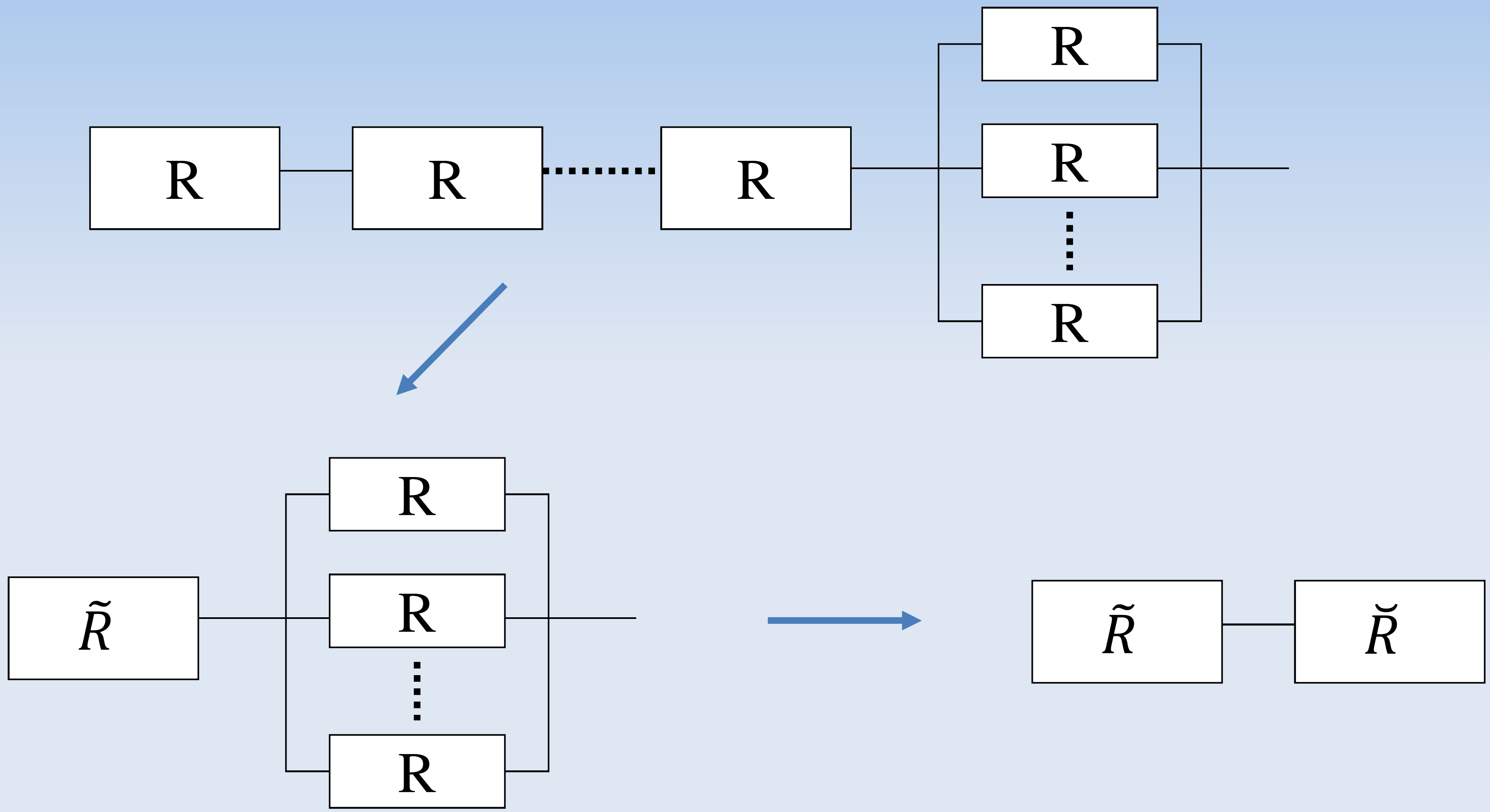
Therefore, the system is working if n or (n-1) or ... or m components work.



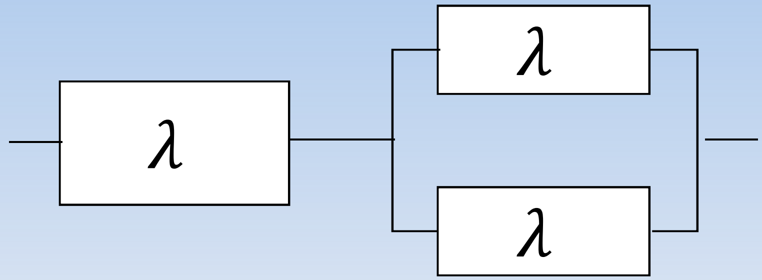
n=N, m = M: (MooN)

$$R_S = \binom{n}{n} R^n (1-R)^0 + \ldots + \binom{n}{m} R^m (1-R)^{n-m}$$

Hybrid Configurations



Example

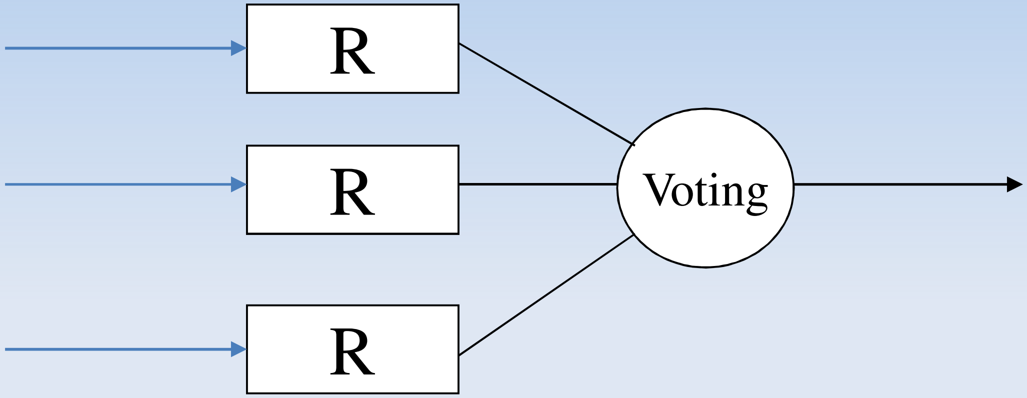


Calculate the MTBF of this system made of identical components. $\lambda = const.$

$$R_S = R_C * (2R_C - R_C{}^2) = 2e^{-\lambda t} - e^{-3\lambda t}$$

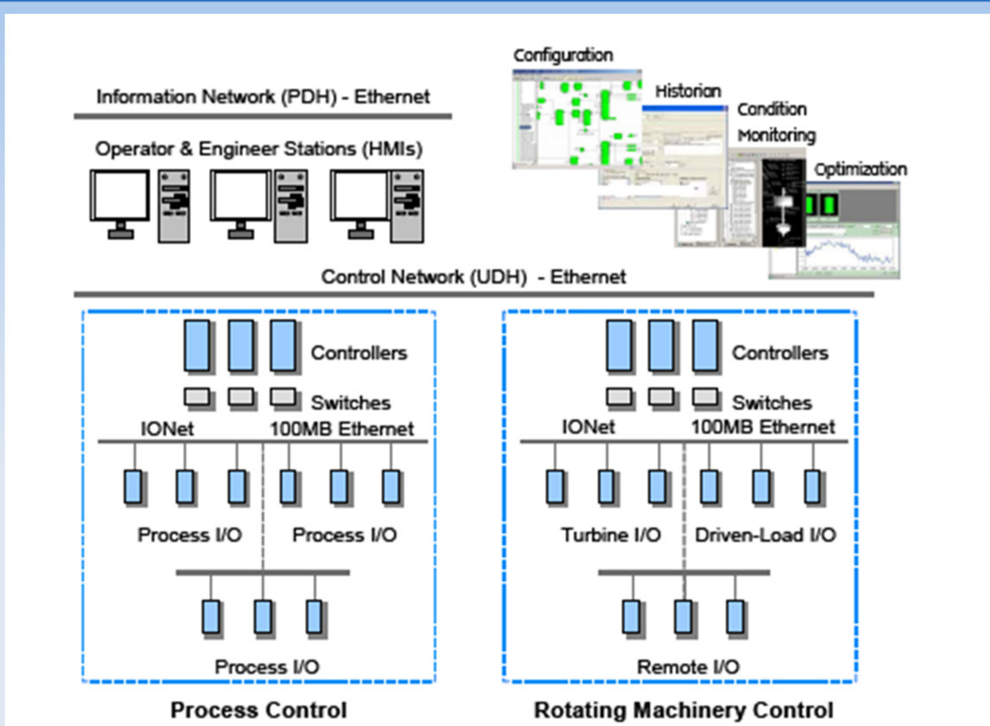
$$\Theta = \int_0^{\infty} (2e^{-\lambda t} - e^{-3\lambda t})dt = \ldots = \frac{2}{3\lambda}$$

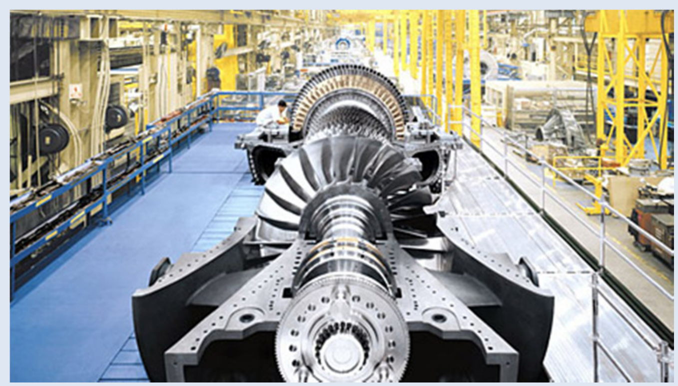
The Majority Voter – 2oo3 - TMR



Reliability: $3R^2 - 2R^3$, Voter very reliable – not considered

Example – Turbine Controller





Source:
GE Energy

Reliability and Availability Summary

- Important part of requirements analysis
- Allows to play with configurations of our systems – provide metrics (MTBF)
- Will also be used to play with different architecture choices later since we apply it at the architecture level.
- If our system is part of a plant (assembly line) availability calculations help to determine plant downtime (MDT).