Clereco

Cross-Layer Early Reliability Evaluation for the Computing cOntinuum

SEVENTH FRAMEWORK PROGRAMME

Project Number: FP7-611404

D2.4.2 - Report on system level reliability metrics (final)

Authors[1]

A. Grasset (THALES), S. Ozdemir (INTEL), R. Canal (UPC), G. Rafiq (ABB), T. Løkstad (ABB), K.-J. Alme (ABB), S. Tselonis (UoA), N. Foutris (UoA), M. Kaliorakis (UoA), D. Gizopoulos (UoA), S. Di Carlo (POLITO)

Version 1,0 – 24/03/2016

Lead contractor: THALES
Contact person: Arnaud Grasset Thales Research & Technology 1, avenue Augustin Fresnel 91767 Palaiseau Cedex, France E-mail: arnaud.grasset@thalesgroup.com
Involved Partners[2]**:** THALES, UPC, ABB, POLITO, CRNS, YOGITECH, UoA
Work package: WP2
Affected tasks: T2.3

Nature of deliverable[3]	[✓ R]	P	D	O
Dissemination level[4]	[✓ PU]	PP	RE	CO

[1] Authors listed here only identify persons that contributed to the writing of the document.

[2] List of partners that contributed to the activities described in this deliverable.

[3] **R**: Report, **P**: Prototype, **D**: Demonstrator, **O**: Other

[4] **PU**: public, **PP:** Restricted to other programme participants (including the commission services), **RE** Restricted to a group specified by the consortium (including the Commission services), **CO** Confidential, only for members of the consortium (Including the Commission services)

COPYRIGHT

CONFIDENTIALITY NOTE

INDEX

Scope of the document

This document is an outcome of the task T2.3 **"Definition of system level reliability metrics"** described in the description of work (DoW) of CLERECO project under Work Package 2 (WP2).

Figure 1 depicts graphically the goal of this deliverable, its main results, the inputs it uses and which work packages will use its outputs.

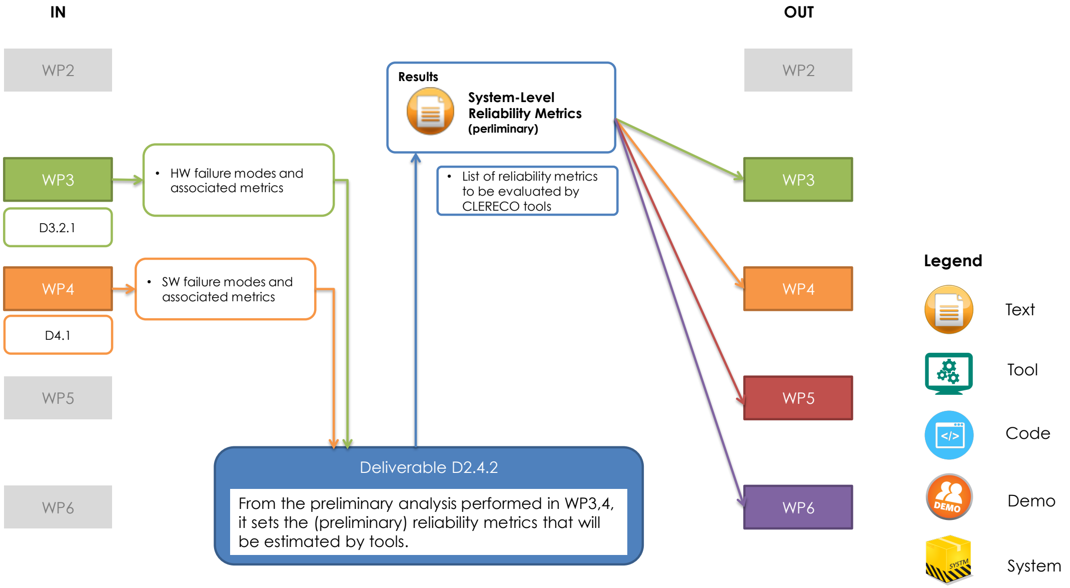

IN
OUT
WP2
WP3
D3.2.1
WP4
D4.1
WP5
WP6
• HW failure modes and associated metrics
• SW failure modes and associated metrics
Results
System-Level Reliability Metrics (perliminary)
• List of reliability metrics to be evaluated by CLERECO tools
Deliverable D2.4.2
From the preliminary analysis performed in WP3,4, it sets the (preliminary) reliability metrics that will be estimated by tools.
WP2
WP3
WP4
WP5
WP6
Legend
Text
Tool
Code
Demo
System

Figure 1: Task and deliverable overview

This document sets the reliability metrics that will be likely estimated by the CLERECO framework. The outcome of this deliverable is thus a preliminary list of reliability metrics that have been selected as interesting for the CLERECO project. This list is a revision of the preliminary list presented in deliverable D2.4.1. The focus is on system-level reliability metrics. Some metrics that are more intermediate metrics than system-level have however been included. An intermediate metric is a metric computed by the flow and used to compute other metrics. As intermediate metrics can in some case be required by end-users, some of them are presented in the document and can be redundant with metrics presented in deliverables D3.2.1 (Report on components reliability aspects and models) and D4.1 (Software impact on system reliability: Metrics and models).

The document is organized in the following Sections:

- **Introduction.** It introduces the background in which reliability metrics of interest for CLERECO can be identified.
- **Terminology and concepts.** It presents the general concept of dependability and the taxonomy used for the classification of reliability metrics.
- **System level reliability metrics.** It lists and defines existing reliability and dependability metrics.
- **Impact of faults on non-functional characteristics.** It discusses the impact of faults on performance, energy and other non-functional parameters of the system.

- **Safety standards and norms.** It shows how the CLERECO methodology could comply with existing safety standards of different application domains.
- **Metrics selections.** It presents the selection of reliability metrics that has been done for the CLERECO methodology.
- **Conclusions**. A short summary of the activities described in this deliverable.

The taxonomy of the metrics was already very extensive and well defined in the first release of this deliverable. In this final version of the document the metrics selections of the deliverable D2.4.1 has been revised according to the progress of the project. No modifications have been done in others sections as no significant changes in the state-of-the-art have been identified since the preliminary version of the document.

1. Introduction

As devices dimensions shrink and new types of devices progressively appear (FinFET [1], FDSOI [2], RRAM [3], etc.), computing systems are going to be more and more affected by an increasing number of hardware faults [30]. It is foreseen that in the near future transient, intermittent and permanent faults, as well as higher static and dynamic variations could severely compromise the reliability of Integrated Circuits (IC) [5] .Thus, the evolution of semiconductor technologies is now being considered as major risk for the throughput and the delivered quality of service in high-performance computing systems, for the safety of critical systems [4], as well as the cause of decline in the performance of consumer electronic systems [36].

To support this trend, the CLERECO project recognizes the early reliability estimation as one of the main challenges. This evolution towards new reliability estimation methods also involves revising how reliability is observed and measures, i.e., what metrics are used, and requires a redefinition of some of them. The selection of a metric is indeed heavily dependent of its use and the application domain of the computing system. Different metrics can eventually be used to drive the design and to validate it. Using the right metrics to drive the design is a key point, because inappropriate metrics could lead to non-optimal design decisions. The metrics are required not only for the evaluation of reliability, but also for the understanding of the system behavior and of the propagation of faults, and how faults can affect other non-functional aspects of the system.

Figure 2 represents the propagation of faults through the different layers of the system. We note that in the context of CLERECO only hardware faults are considered. The software faults resulting from mistakes in the specification or the implementation of software parts are out of the scope of the document and of the CLERECO project. System-level reliability metrics are necessary to evaluate the dependability of the system. However, finer grained metrics should provide information concerning the faults' propagation through the system and the resilience of the HW/SW layers, too.

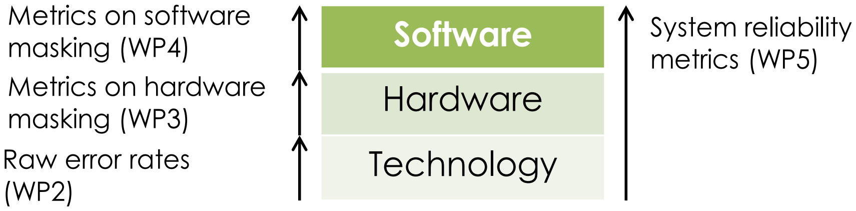

Metrics on software masking (WP4)
Metrics on hardware masking (WP3)
Raw error rates (WP2)
Software
Hardware
Technology
System reliability metrics (WP5)

Figure 2: CLERECO reliability stack

This document lists existing and already well-known reliability metrics of computing systems (like SDC/DUE FIT rates) that are generally found in the literature and/or used in the industry, too. These metrics have been classified according to typical taxonomy of dependable computing [6]. All core metrics that have been identified as essential for different application domains have been included. An expression of reliability metrics under a generic and non-application specific form has been favored. The list of metrics that presented in this document is as much comprehensive as possible. In order to have a clear overview of the different kind of metrics that can be used, a broad list of state-of-the-art metrics has first been elaborated. A reduced number of metrics to evaluate with the CLERECO framework has been selected from this list. New metrics have also been investigated and metrics that have to be redefined have also been identified. The selected metrics take into account the impact of faults that do not result in failures on other usual metrics (like power consumption, performance, etc.). The challenge is to determine how to quantify them through new metrics (or by redefining old metrics). Moreover, an analysis of the requirements of safety standards has been done, to prepare the integration of the CLERECO methodology and tools into industrial design flows. The objective of this analysis was only to check that metrics required by standards had been considered.

2. Terminology and concepts

This section briefly presents the general concepts of reliable and dependable computing. The concepts and the taxonomy [6] presented here will serve as the basis for the classification of reliability metrics in the rest of the document. As introduced before, the goal of the deliverable is not only to study the impact of faults on the reliability of the system, but also to study how the occurrence of faults is correlated to other parameters that are generally used as the figures of merit of a system.

2.1. Fundamental system properties

A computing system is generally evaluated with respect to a set of basic fundamental properties. These properties characterize the system and are the figures of merits of the system. Thus, all these different aspects should be considered, when we evaluate the global impact of hardware faults on a system:

- Functionality
- Performance
- Energy, power, power efficiency
- Size, Weight and Power (SWaP)
- Cost (design, manufacturing)
- Dependability
- Security

This deliverable is mainly focused on the impact of faults on dependability as the concept of dependability includes the reliability and covers all aspects related to the deviation of a system from its specifications due to manufacturing faults as well as physical faults. However, the impact of faults on performance, power and cost is studied as well.

2.2. Concepts of dependability

This section provides a short glossary of reliability-related terminology to guarantee a common understanding of the terms used in this document. The taxonomy of dependable computing defined in [6] is used as a reference. Dependability can be defined as a global concept covering different aspects of a system and representing the extent to which a system is expected to operate in compliance to its specifications. The attributes of dependability are defined as follows in [6].

- **Availability:** *readiness for correct service*.
- **Reliability:** *continuity of correct service.*
- **Safety:** *absence of catastrophic consequences on the user(s) and the environment.*
- **Integrity:** *absence of improper system alterations.*
- **Maintainability:** *ability to undergo modifications and repairs.*

Figure 3 represents these different concepts, as well as the dependability threats and the means to improve dependability.

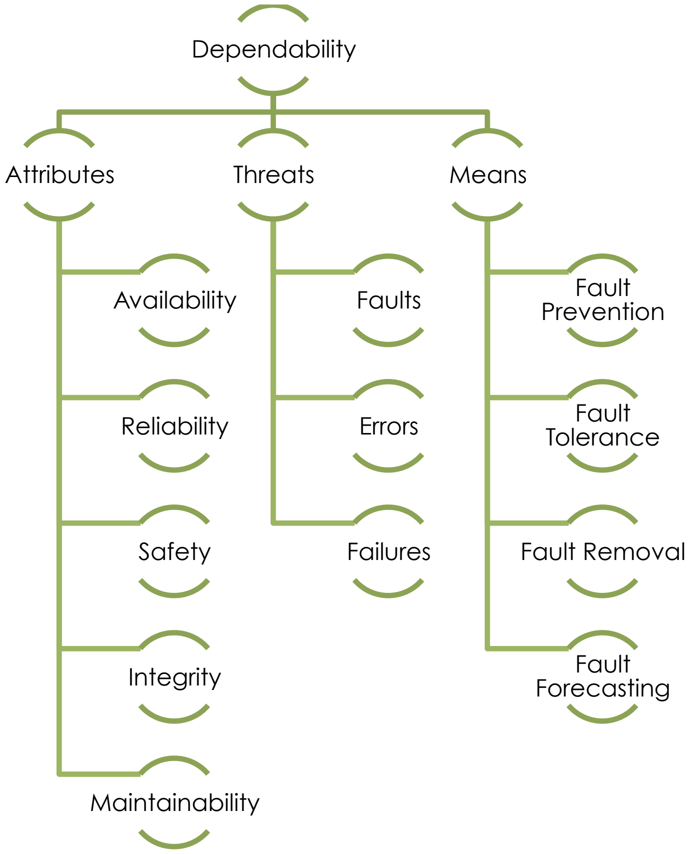


Figure 3: Dependability concepts [6]

The terms used to describe a dependability threat are defined as follows.

- **Failure:** *an event that occurs when the delivered service deviates from correct service. A service fails either because it does not comply with functional specification, or because this specification did not adequately describe the system function. Correct service is delivered when the service implements the system function* [6].
- **Error:** *part of the total state of the system that may lead to its subsequent service failure. It is important to note that many errors do not reach the system's external state and thus do not cause a failure [6].*
- **Fault:** *Adjudged or hypothesized cause of an error. A fault is active when it causes an error; otherwise it is dormant* [6]**.**

3. System level reliability metrics

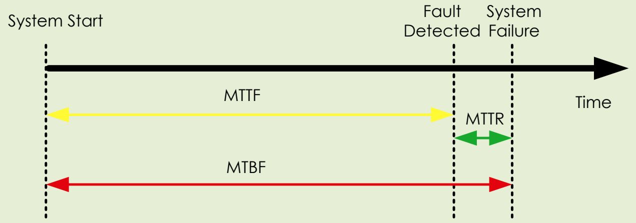
This section lists and classifies the reliability and dependability metrics that are traditionally used for the computing systems. Most of these metrics are applicable at different levels of abstraction and for different components of the system. When it has been considered as particularly relevant, it has been explicitly stated that these metrics could be applied for different components. But as indicated in the introduction, the goal of the deliverable is to list all core metrics, and the list of secondary metrics cannot be considered as exhaustive. The definition of the reliability metrics is thus closely related to the work done in work packages WP3 and WP4. An effort has been made to maintain consistency between the definitions in this deliverable and in the deliverables of work packages WP3, WP4 and WP5.

3.1. Dependability attributes

Table 1 lists the basic metrics. Most of these metrics are applicable at the system level as well as at the component level.

Table 1: Summary of basic reliability metrics

Name	**Description/Comment**
Failure rate (λ)	Number of failures per unit of time. If the failure rate λ is constant, reliability can be modeled using an exponential distribution: $R(t) = e^{-\lambda t}$
Failures In Time (FIT)	Measure of failure rate in 10^9 device hours. $FIT = \lambda_{hours} * 10^9$ The FIT rate formula of an entire system is the following: $FIT_{system} = \sum_{i=0}^{n} FIT_i$ The formula assumes that the errors in each sub-component are independent. As a result, if a system is comprised of two sub-components, each one having an error rate of 20 FIT, then the total failure rate of this system is 40 FIT.
Mean Time To Failure (MTTF)	Arithmetic mean (average) time to failure of a system. Usually expressed in hours. For instance, if a system's MTTF is 2 years, then on average a failure occurs every 2 years. It is a basic measure of reliability for non-repairable items. For non-repairable items with constant failure rate: $MTTF = \frac{1}{\lambda}$ A system's MTTF is derived by summing the individual MTTF's of the various system sub-components. For instance, assuming that a system consists of two sub-components, with MTTF$_{component1}$ and MTTF$_{component2}$ respectively, then the MTTF of the system is: $MTTF_{system} = \frac{1}{\frac{1}{MTTF_{Component1}} + \frac{1}{MTTF_{Component2}}}$ and in general: $MTTF_{system} = \frac{1}{\sum_{i=0}^{n} \frac{1}{MTTF_i}}$ Finally, MTTF and FIT metrics are correlated as illustrated in the following equation (a mnemonic rule of the correlation between FIT and MTTF is that an MTTF of 1000 years translates into a FIT rate of 114 FIT). $MTTF\ (in\ years) = \frac{10^9}{FIR\ rate \times 24\ hours \times 365\ days}$
Mean Time To Repair **(MTTR)**	Expresses the mean time to repair an error once it is detected. It therefore measures service interruption. This time is determined by the repair and recovery mechanisms that a system is equipped with. The smaller the MTTR the higher the reliability of the system. It is a basic measure of the maintainability of repairable items.
Mean Time Between Failure (MTBF)	Arithmetic mean (average) time between failures of a system. Usually expressed in hours. Basic measure of reliability for repairable items. MTBF is calculated from the following formula that is also visualized in the figure: $MTBF = MTTF + MTTR$

Name	Description/Comment
	 System Start Fault Detected System Failure MTTF Time MTTR MTBF
Mean Work to Failure (MWTF)	Captures the average amount of work between two errors and is useful to compare the reliability of different workloads [40]
Mean Instructions to Failure (MITF)	Expresses the average number of committed instructions in a microprocessor between two errors [41]
Uptime/downtime	Period of time that a system succeeds/fails to provide or perform its primary function.
Availability	Ability of a system to be in state to perform a required function at a given instant of time or at any instant of time within a given time interval, assuming that the external resources, if required, are provided [32]. $Avaialbility = \frac{MTTF}{MTTF + MTTR}$ Alternatively a measure of unavailability can be used.
Useful life	Period of time when the failure rate turns from a random event to a predictable event based on normal wear-out [31].

3.2. Dependability threats

The metrics that characterize the dependability threats are necessary to understand and identify the weak points of the system. As explained in Section 2 the dependability threats are generally described by the concepts of faults, errors and failures (see Figure 4). Dependability threat metrics measure the probability of occurrence of these events, and the relations among these events.

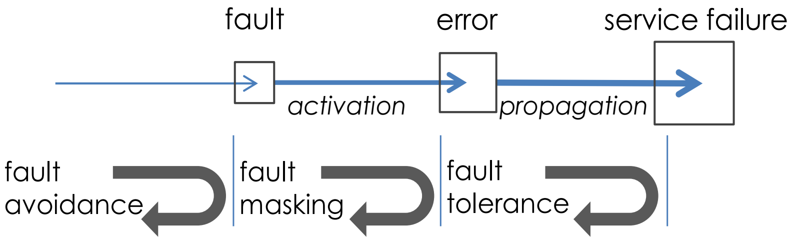


Figure 4: The chain of dependability threats

3.2.1. Fault

Following the taxonomy of [6], the faults can be classified according to different properties: the phase of creation or occurrence, the system boundaries, the phenomenological cause, the dimension, the objective, the intent, the capability and the persistence. In the context of the CLERECO project, only hardware faults are considered. Thus, the types of faults that are covered by the methodology are the following:

- Manufacturing defects: open or short circuits, parametric failures

- Physical deterioration: wear-out effects like NBTI, electromigration, TDDB, HCI …
- Physical interference: soft-errors and electromagnetic interferences (EMI)

Future manufacturing technologies may expose different types of faults and corresponding fault models. CLERECO will regularly monitor the evolution to update the above list. Moreover, the concept of intermittent faults [33] has been added to the used taxonomy in order to take into account the different nature and impact of these faults.

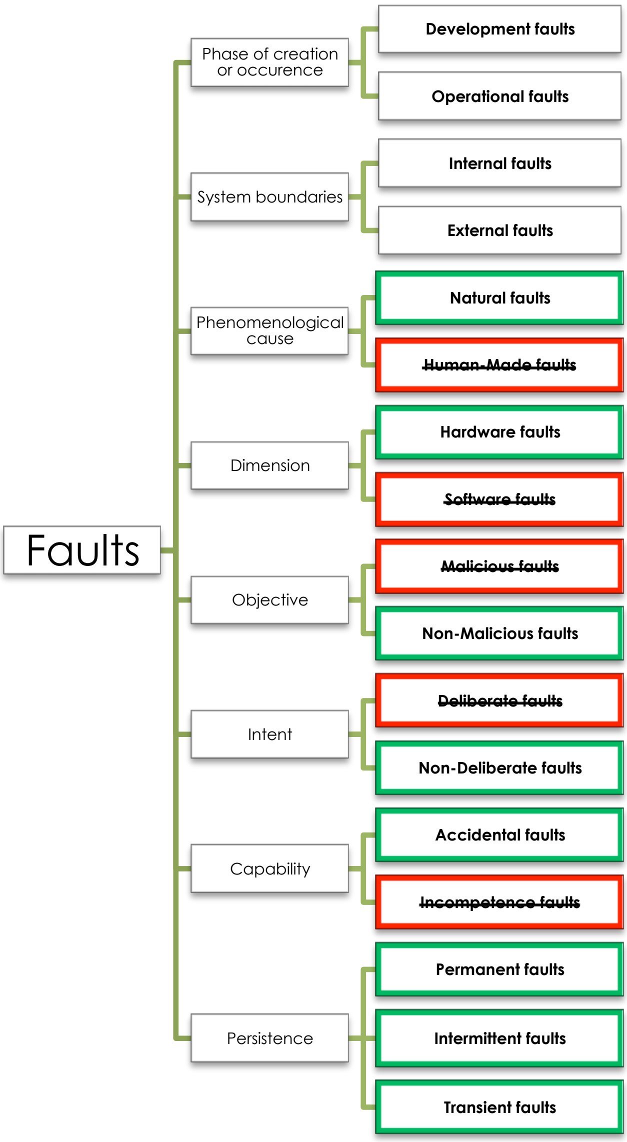


Figure 5: Classification of faults (based on [6])

Table 2 lists the metrics that are typically used for the measurement of fault occurrences in a computing system.

Table 2: Metrics on faults

Name	Description/Definition/Comment
Single Event Upset (SEU) rate	Measures the occurrences of SEU per unit of time. Expressed in FIT/Mb or SEU/Mbit/h. Depends on technology and environment.
Multiple Bit Upset (MBU) rate	Measures the occurrences of MBU per unit of time. A MBU is defined as "*any event or series of events that cause more than one bit to be upset during a single measurement*" [39].
Single Event Functional Interrupt (SEFI) rate	Measures the occurrences of SEFI per unit of time. A SEFI is "*a soft error that causes the component to reset, lock-up, or otherwise malfunction in a detectable way, but does not require power cycling of the device (off and back on) to restore operability*" [38].
Single Event Transient (SET) rate	Measures the occurrences of SET per unit of time. A SET is defined as a "*momentary voltage excursion (voltage spike) at a node in an integrated circuit caused by a single energetic particle strike*" [38].
Single Event Latch-Up (SEL) rate	Measures the occurrences of SEL per unit of time. A SEL is a "*abnormal high-current state in a device caused by the passage of a single energetic particle through sensitive regions of the device structure and resulting in the loss of device functionality*" [38].
Intermittent failure rate	Number of intermittent faults per unit of time.
Permanent failure rate	Number of permanent faults per unit of time. The metric is generally broken down by failure mechanisms (λ_{NBTI},λ_{TDDB},λ_{HCI}, …). Assuming constant failure rates of the individual failure mechanisms, the permanent failure rate can be computed by a sum of failure rates.

3.2.2. Errors

When faults are activated, they result in errors. However, a large part of faults are not activated but dropped. The possible outcomes of a single-bit fault in different states have been classified in [7] and are represented in Figure 6.

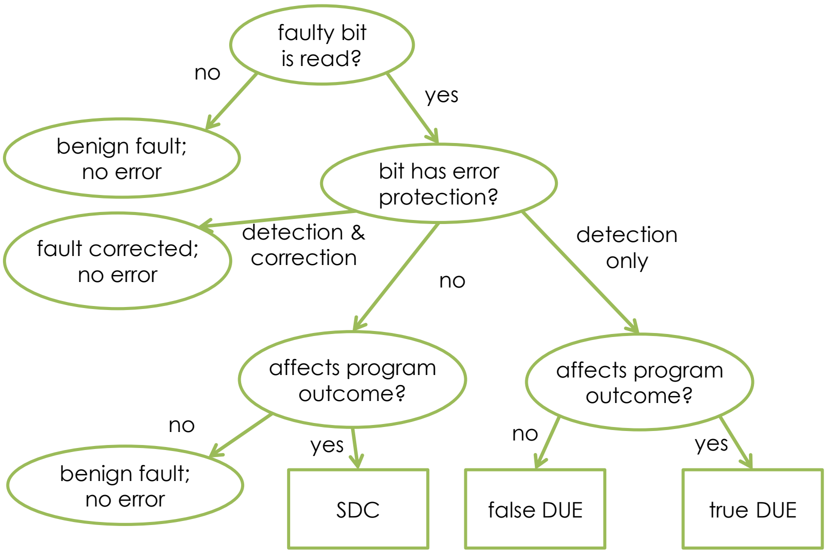


Figure 6: Classification of the possible outcome of a faulty bit [7]

Table 3 presents the metrics that are most commonly used for the characterization of errors. It includes metrics that are used to characterize the probability of occurrence of errors and also to characterize the probability of activation of a fault (which results in an error).

Table 3: Metrics on errors and vulnerability to errors

Name	Description/definition/Comment
Silent Data Corruption (SDC) rate	Probability of occurrence of Silent Data Corruption. Form of error where a fault induces the system to generate erroneous outputs [34].
Detected Unrecoverable Error (DUE) rate	Probability of faults that are detected but cannot be recovered.
Architectural Vulnerability Factor (AVF)	Measures the vulnerability of a hardware structure to faults. Defined as the probability that a fault in that particular structure will result in an error [9].
Program Vulnerability Factor (PVF)	Captures the architecture-level fault masking inherent in a program, allowing software designers to make quantitative statements about a program's tolerance to soft-errors [35].
Hardware Vulnerability Factor (HVF)	Quantifies the vulnerability of hardware structures to errors [35].
Hard-Fault Architectural Vulnerability Factor (H-AVF)	Measures the probability to commit an erroneous state due to hard faults in a microprocessor structure [24].

3.2.3. Failures

A failure of the system happens when the delivered service deviates from correct service. The way the system deviates from a correct service is the failure mode of the system. Figure 7 represents the set of failure modes of a computing system as defined in [6].

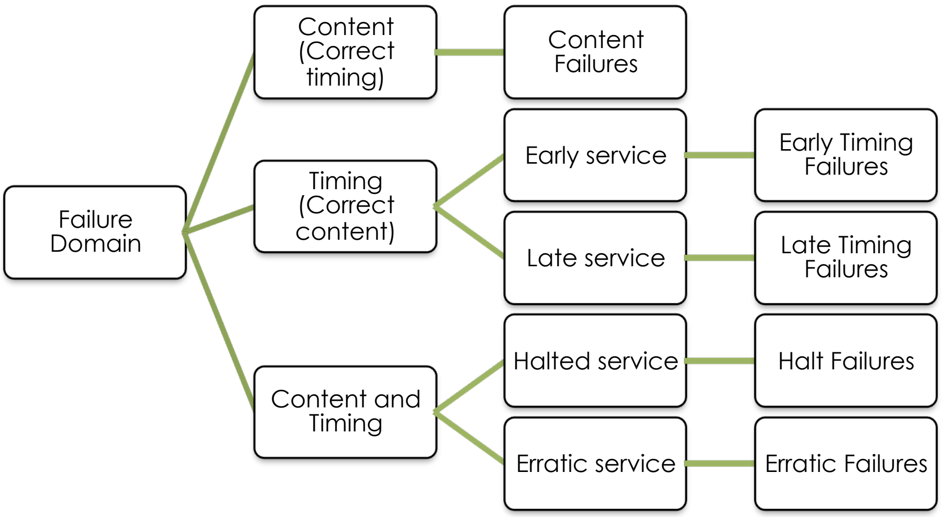


Figure 7: Failure modes of a computing system [6]

Beside its mode, a failure is characterized by:

- Its detectability - If the failure of the system is detected and signaled.
- Its consistency - If the failure is perceived identically by all system users.
- Its severity and its consequences - If the failure has a minor impact or catastrophic consequences.

The notion of failure severity is heavily dependent on the application domain. There are no generic definitions of the severity of the system. Table 4 lists the failure metrics, which correspond to the different types of failures that are identified in deliverable D4.1 (Software impact on System Reliability: Metrics and Models).

Table 4: Metrics on failures

Name	Description/ Definition/ Comment
Early timing failure rate	Probability of an early timing failure.
Late timing failure rate	Probability of a late timing failure.
Deadline miss ratio	Percentage of task deadlines missed in soft real-time applications.
Fatal Hardware Traps rate	Probability of Fatal Hardware Traps.
Hangs	Probability of failure due to a hang in the application or OS.
Abnormal Application Exit rate	Probability of Abnormal Application Exit.
High OS activity	Probability of failure due to High OS activity.

Some metrics are purely related to the application or the application domain. For instance, the number of packet loss or the corrupted ones could be used to evaluate the dependability of a telecommunication application. Establishing an exhaustive list of all application-specific metrics is not possible and is anyway out of the scope of this document.

3.3. Dependability enhancing mechanisms

Figure 8 represents the general approaches that are used to improve dependability. A set of metrics is used to characterize the used means to achieve the required level of dependability.

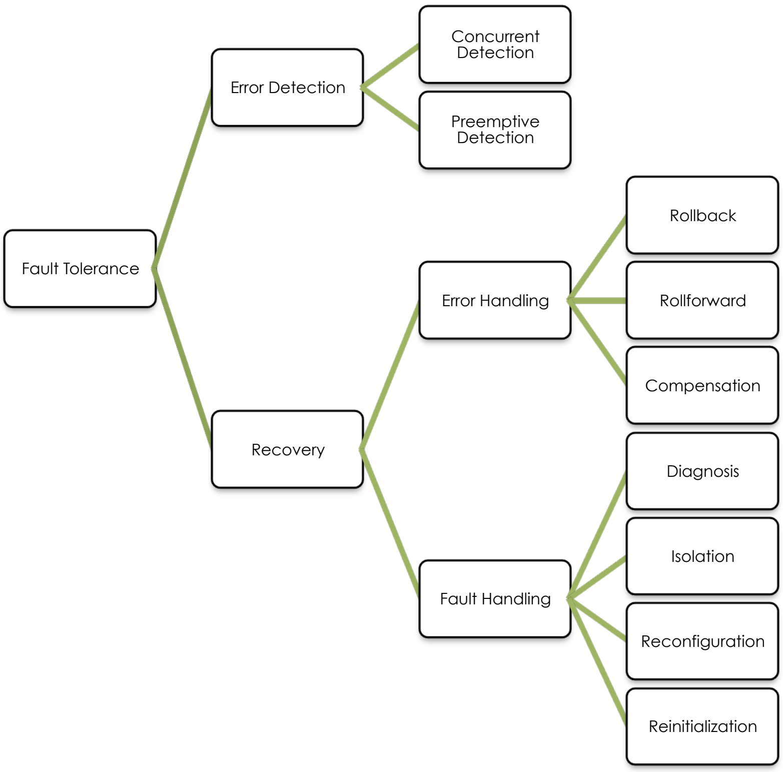


Figure 8: Classification of dependability means

Table 5 presents the metrics related to dependability mechanisms.

Table 5: Metrics on dependability mechanisms

Name	Description/Comment
Test coverage	Percentage of modeled faults that are detected by a set of test vectors [15]: $$T = \frac{Number\ of\ modeled\ faults\ detected}{Total\ number\ of\ modeled\ faults}$$
Counting Error Recovery/Repair	e.g. ECC counter for bit corrections
Failure mode coverage	Measures that the failure modes of all components are in the FMEA model [ref to add].
Mean Time To Detection	Average time between apparition of a fault and its detection.
Mean Time To Recovery	Average time for the recovery of the system.
Safe Failure Fraction (SFF)	Measures the cover and the effectiveness of the diagnostics in the system. Defined as the ratio of detected failures in a system as compared to the total failure rate.

4. Impact of faults on non-functional characteristics

This section explores existing and possible new metrics to evaluate the impact of faults on the different figures of merit of a system other than the functional correctness. Even if a fault does not lead to the failure of the system, it can significantly influence its behavior, and as a result, the user's experience or the cost related to the use of the system. A fault can indeed impact the behavior of the system without impacting its outcomes. For instance, a fault in a performance related part of the system (predictors) could degrade (or in exceptional cases improve) its performance.

A fault can impact the non-functional parameters and the figures of merits of a system in the following ways:

- Malfunction of the system: the fault leads to a malfunction of a part or the system and a non-optimal operation even though the final provided service is correct.
- Error & recovery overhead: faults are correctly handled by protection mechanism that ensure a correct service but at the cost of a reduction of performance or a degradation of power consumption (depends on fault rates and overhead of fault mitigation).

4.1. Performance

Embedded and high-performance microprocessor architectures integrate an arsenal of performance-related mechanisms towards a single aim: deliver the highest possible performance. However, chip manufacturing technologies have already entered an era when permanent, intermittent and transient faults have unacceptable high rates due to manufacturing defects, process variation, environmental impact, wear-out and aging effects. Apart from functional correctness, hardware faults can impact performance. The permanent errors in architectural (e.g., cache memories) and non-architectural arrays (e.g., branch predictor, branch target buffer, prefetcher) of a microprocessor can have a severe impact on performance [18]. In particular [8] introduces the performance vulnerability factor (PVF) that is an analytical metric to estimate the performance degradation due to hardware errors in the non-architectural structures of a microprocessor. The following table analyzes the parameters of PVF metric.

Table 6: Definition of metrics and parameters for the estimation of PVF.

Name	Description/definition	Comment
ETV	$ETV = \#failures \times penalty$	Execution Time Vulnerability (ETV) is the normalized execution cycle increase of a program due to faults in arrays.
Penalty	$penalty = \dfrac{overhead}{C_{base}}$	Penalty is the normalized overhead of a single failure to the duration of a fault free run.
CC	$CC = \dfrac{1}{1 + ETV}$	Computation Capacity (CC) is the fraction of the fault free performance that is still available given the current hardware conditions.
PVF	$PVF = 1 - CC$	Performance Vulnerability Factor (PVF) gives an estimate of the performance degradation that hard faults in prediction structures and caches (those that don't lead to functional errors) can add in an application.

Table 7 shows the system-level impact of performance degradation in HPC and Embedded Computing systems.

Table 7: Performance Degradation examples.

Name	Description/definition	Comment
Average performance degradation	Average Performance slow-down due to hardware errors.	It can be observed due to the effects of faults and the overhead induced by the use of error detection and recovery mechanisms [18] [19] [21] [22] [20].
Worst-Case Execution Time degradation	Maximum performance slow-down due to hardware errors.	Real-time systems, which are required to deliver results by a deadline, require knowledge of the system timing, frequently in the form of worst case execution times (WCET). A degradation of WCET can be observed due to the effects of faults and the overhead induced by the use of error detection and recovery mechanisms [21] [22].

4.2. Power

The emergence of battery-operated devices and the need of high-performance systems for power usage effectiveness, make low power designs increasingly popular. Thus, the researchers put intensive efforts on power estimation and modeling in order to meet the low power requirement [25]. However, hardware faults can impact power. As a result, power budget requirements that are set on design phase may be proven inadequate due to hardware errors. For instance, faults on the tag sub-field of cache memories may result to unexpected cache misses, which in turn will erroneously fetch data from a lower level of memory hierarchy and therefore impact power. To the best of our knowledge none metric has been defined in the literature that accurately estimates the excessive power consumed due to errors early enough in chip's development cycle.

CLERECO project aims to develop such metrics to steer the reliability evaluation of microprocessor designs. If such metrics are successfully defined and correlated to the system behavior due to faults, we will provide details in the final version of this deliverable.

4.3. Others metrics: safety, security, cost

Table 8 lists the impact that faults can have on other non-functional parameters.

Table 8: Impact on non-functional parameters

Name	Description/Comment
β **Factor**	Defined as "*the fraction of the total failure probability attributable to dependent failures*" [29]. $\beta = \frac{\lambda_c}{\lambda}$ where λ_c is the failure rate due to Common Cause Failures
Design diversity metric	Proposed by [27] to evaluate the diversity of two implementations used in a redundant system. Defined as "*the probability that, in response to a given input sequence, the two implementations either produce error-free outputs or produce different error patterns on their outputs*".
Total Cost of Ownership (TCO)	Direct and indirect cost of a system. Dependent on system reliability [28].

5. Safety standards and norms

This section presents the existing safety standards (and norms) from different application domains that have been studied and analyzed in the CLERECO project. The safety standards generally define the safety requirements at the system level, and provide guidelines and methods to check that the requirements are met. It is important to highlight that safety standard are not the main goal of the CLERCO project. Nevertheless, it is important to identify the requirements of standards regarding the reliability metrics to evaluate in order to take proper choices that meet industrial practices.

5.1. Industrial domain

The aim of this section is to list important safety metrics/measures defined by some of the key safety standards relevant for the industrial sectors. Additional focus is also given to the non-sector specific safety standard IEC 61508, due to its importance in developing control products targeting industrial safety applications. The standard IEC 61508 also plays an important role in the sector specific safety standards.

IEC 61508 consists of 7 parts. The parts 1 to 4 are normative and 5 to 7 are informative. The functional steps in the application of IEC 61508 are indicated in part 6 of the standard.

The standard IEC 61508 defines requirements for systems consisting of hardware and software, grouped into the following failure categories:

- *Random hardware failures* can be split into permanent or transient error. Permanent errors exist until their repair, but transient errors occur randomly, without damaging the hardware. They can usually be mitigated by simple measures to control failures (i.e., detection and correction mechanisms), e.g., by sending packages twice or recalculate twice. However, detecting them is not simple, because the periodic diagnostics will normally not detect them.
- *Systematic failures* there exist in hardware and software and are hardly eliminated. If only they are found then they can be eliminated. Measures on how to avoid systematic failures are specified in the standard. Typically avoidance of systematic failures is handled by good design procedures and measuring the detected design flaws, while control of systematic failures can be achieved by diversity.
- *Common cause failures (CCF)* are the result of one or more events, causing concurrent failures in two or more separate channels in a multi-channel system, leading to system failure. This is typically caused by environmental issues (e.g., temperature, EMC, etc.) simultaneously in redundant hardware (safety function carried out more than once). As diversity introduces differences in hardware, design, or technology, this kind of failures will be reduced.

In general, the safety level for IEC 61508 certified devices is specified using the Safety Integrity Level (SIL). Four levels are defined in the standard (SIL 1 to SIL 4), taking into account both systematic- and hardware integrity aspects:

- Systematic Integrity: Avoidance and control of systematic failures, by following procedures and requirements from the standard. In general, the probability of occurrence of systematic faults cannot be quantified.
- Hardware Integrity: Determined based upon calculated PFH (Average frequency of dangerous failures) and PFD (Probability of Dangerous Failure on Demand), for high demand and low demand systems, respectively. These high-level safety metrics are calculated based upon the hardware (HW) failure rates, HW architecture/fault tolerance (HFT), and the platform diagnostic capabilities. In addition, the maximum

SIL that can be claimed is limited by architectural constraints, considering the subsystem safe failure fraction (SFF) and hardware fault tolerance (HFT).

From a HW integrity point of view in order to fulfill a given SIL, the calculated PFH and PFD_{avg} need to be within the defined limits in Table 9. These values are required, but not sufficient, in order to claim a certain SIL. Requirements for systematic capabilities and architectural constraints must also be fulfilled for the targeted SIL level.

Table 9: Safety Integrity Level (SIL), as specified in [11]

	High demand of operation*	**Low demand of operation****
Safety Integrity Level (SIL)	Average frequency of a dangerous failure of the safety function, PFH [h^{-1}]	Average probability of a dangerous failure on demand of the safety function, PFD_{avg}
4	$\geq 10^{-9}$ to $< 10^{-8}$	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-8}$ to $< 10^{-7}$	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-7}$ to $< 10^{-6}$	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-6}$ to $< 10^{-5}$	$\geq 10^{-2}$ to $< 10^{-1}$

***High demand or continuous mode: frequency of demands are greater than once per year or the safety function retains the safe state as part of normal operation [11].**

****Low demand: safety function is only performed on demand, and the frequency of demands is no greater than once per year [11].**

Different architectures can be applied in order to simplify the process of achieving a target SIL. One example with a 1oo2D architectures is shown in Figure 9. In normal operation both channels need to put a demand in order to trigger the safety function, but in cases of detected failures the voting is adapted so that the faulty channel has no impact on the output. For cases where there are discrepancies between channels (or detected failures in both channels), the diagnostic circuitry will force the output to safe state.

The 1oo2D architecture with redundant HW (two channels) could be implemented with additionally external HW watchdog (channel independent diagnostics) to successfully mitigate common cause failure (CCF).

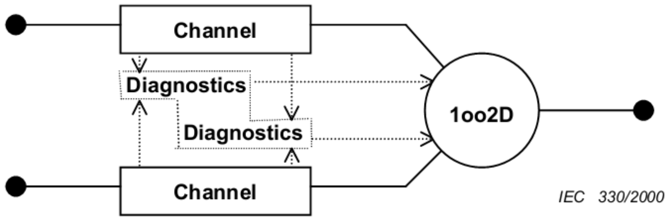


Figure 9: 1oo2D architectures

The HW of a safety-instrumented system often consists of sensors, logic, and actuators as shown Figure 10. The PFH is split into the different parts leaving typically 15% for the programmable logic (where 1% is normally allocated for communication), 35 % for the sensor, and 50 % for the actuator.

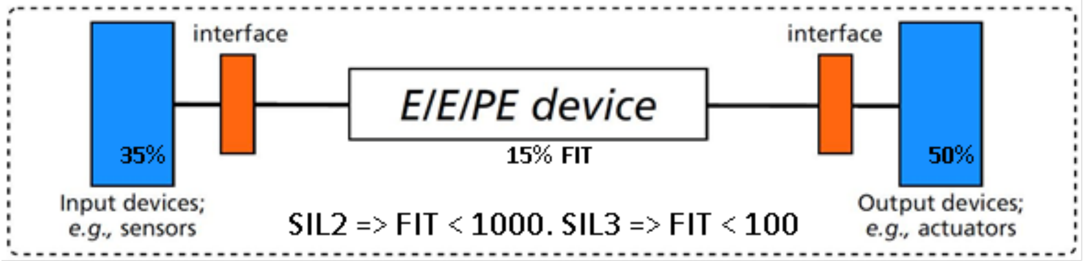


Figure 10: Input subsystem, Logic solver subsystem, and Output subsystem. The total PFH needs to be within the limits specified in Table 9

For subsystems, the top-level safety metrics listed in Table 10 should be considered.

Table 10: Top-level safety metrics for determination of SIL due to HW integrity aspects.

Parameter	**Unit**	**Description/comment**
PFH*	h^{-1}	Average frequency of dangerous failure
PFD_{avg}**	-	Average probability of dangerous failure on demand, mean unavailability.
SFF	-	Safe Failure Fraction**, SFF = $(\Sigma\lambda_s + \Sigma\lambda_{dd}) / \Sigma\lambda_{tot}$
DC	-	Diagnostic coverage*, DC = $\Sigma\lambda_{dd} / \Sigma\lambda_d$
HFT	-	Hardware Fault Tolerance (Input parameter)
***For high demand or continuous mode of operation.** ****For low demand mode of operation.**		
Description: **PFH_d or PFD are typically calculated based upon parameters given in Table 11 (as well as the employed HW architecture/HFT). Based upon PFH_d, SFF and HFT, the SIL level from an HW integrity point of view can be determined.**		

Different methods can be used to calculate the top-level safety metrics. Typical input parameters for the calculation are summarized in Table 11.

Table 11: Parameters typically used to calculate top-level safety metrics

Parameter	Unit	Description/comment
T1	h	Proof test interval
T2	h	Diagnostic time interval
λ_s	h^{-1}	Failure rate corresponding to safe failures
λ_d	h^{-1}	Failure rate corresponding to dangerous failures
λ_{dd}	h^{-1}	Failure rate corresponding to dangerous detected failures
λ_{du}	h^{-1}	Failure rate corresponding to dangerous undetected failures
λ_{sd}	h^{-1}	Failure rate corresponding to safe detected failures
λ_{su}	h^{-1}	Failure rate corresponding to safe undetected failures
λ_{tot}	h^{-1}	$\lambda_d + \lambda_s$
β	-	Fraction of undetected failures that have a common cause (Common cause factor for subsystems with HFT > 0). See [11].
β_d	-	Fraction of detected failures that have a common cause. See [11].
MTTR	h	Mean time to restoration, [11]
MRT	h	Mean repair time, [11]

***For systems with HFT = 0, only dangerous failures that are detected and can be reacted upon within the process safety time, can be included in the DC. See [11].**

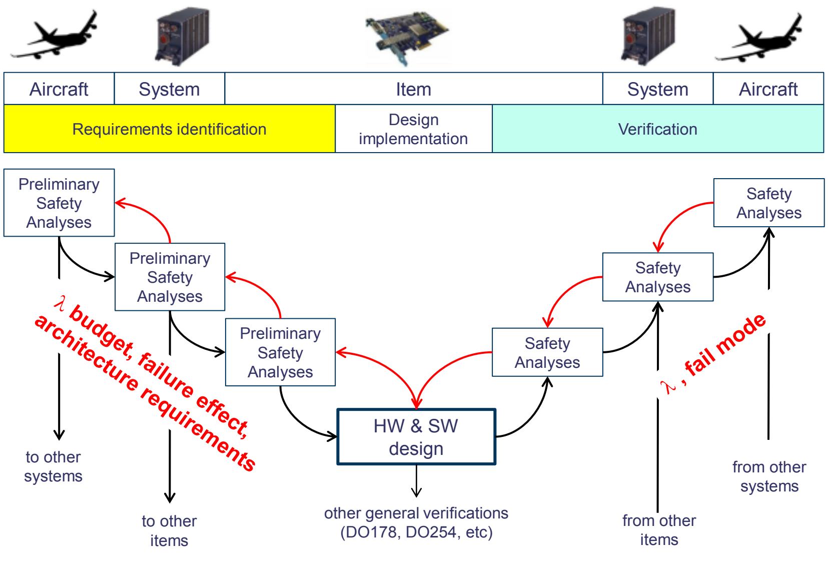
****SFF: No part failures are not used for SFF calculations. See [11].**

In order to determine these parameters, a detailed failure mode effects and diagnostics analysis (FMEDA) for the HW (and the associated channels) is required. Sector specific safety norms as IEC 61511, IEC 62061, and ISO 13849, see [12], [13] and [14], include the standard IEC 61508 as a normative reference for cases where complex devices are utilized in the safety functions. Offering products certified according to IEC 61508 greatly simplifies the application of control products in safety systems for the different industrial sectors.

For additional details see standard IEC 61508, [11].

5.2. Safety-Critical and Mission-Critical domain

In the avionic domain, the whole design process is driven by the safety constraints and the safety analysis. Safety-critical applications in the avionic domains have thus to follow different standards and a strict regulation. Figure 11 illustrates the design process used in the avionic domain and that follows a V-Model. Requirements regarding the reliability of an item derive from the preliminary safety analyses at the aircraft, system and item levels. In the context of avionics system design, it is not planned to use the CLERECO methodology for the safety assessment of the system (second part of the V-cycle), but only to assist the designers during initial phases of the design (first part of the V-cycle).


Aircraft
System
Item
System
Aircraft
Requirements identification
Design implementation
Verification
Preliminary Safety Analyses
Preliminary Safety Analyses
Preliminary Safety Analyses
λ budget, failure effect, architecture requirements
to other systems
to other items
HW & SW design
other general verifications (DO178, DO254, etc)
Safety Analyses
Safety Analyses
Safety Analyses
λ , fail mode
from other items
from other systems

Preliminary Safety Analyses : FHA, PFTA, CCA
Safety analyses : FTA, CCA updates, FMEA & FMES

Figure 11: Safety assessment process diagram

The preliminary safety analysis assigns to each function of the aircraft a level of criticality depending on what could be the impact of a failure of this function. Development methods and reliability requirements directly derive from this analysis (see Table 12).

Table 12: Failure condition severity in avionics

Development Assurance Level	Failure Condition Classification	Failure Condition Effect	Descriptive Probability	Quantitative Probability (Per flight hour)
Level A	Catastrophic	- all failure conditions which prevent continued safe flight and landing	Extremely Improbable	$<10^{-9}$
Level B	Hazardous / Severe Major	- large reduction in safety margins or functional capabilities - higher workload or physical distress such that the crew could not be relied upon to perform tasks accurately or completely - adverse effects upon occupants	Extremely Remote	$<10^{-7}$
Level C	Major	- significant reduction in safety margins or functional capabilities - significant increase in crew workload or in conditions impairing crew efficiency - some discomfort to occupants	Remote	$<10^{-5}$
Level D	Minor	- slight reduction in safety margins - slight increase in crew workload - some inconvenience to occupants	Probable	$<10^{-3}$
Level E	No Effect	- no effect	-	-

The safety assessment process as described by the SAE ARP4754/ED-79 and the SAE ARP4761 standards is based on:

- An aircraft and system Function Hazard Assessment (FHA), which identifies and classifies the failure conditions associated with the function.
- A Preliminary System Safety Assessment (PSSA) that identifies all derived safety requirements.

- A System Safety Assessment (SSA) to show that safety requirements are met. At the difference of the PSSA, which is used to derive HW and SW requirements, the SSA is a verification process.
- A Common Cause Analysis (CCA) that evaluates the overall architecture sensitivity to common cause events. Its outputs are fed into the PSSA and the SSA.

As part of the PSSA, the Fault Tree Analysis is a top-down analysis technique used to determine what single failures or combinations of failures at the lower levels might cause each system failure. It proceeds down through successively more detailed levels of the design. The failure rates of the low level FTA events are provided by a Failure Mode and Effects Analysis (FMEA), which evaluate effects resulting from single failure. It typically includes a description of the Failure modes and associated hardware failure rates, the failure effects and the detectability and means of detection. Table 13 lists the failures rates that are typically included in the FMEA.

Table 13: Example of failure rates evaluated by the FMEA

Failure modes	Description
SEU Erroneous Program Execution	Failure rate for detected and not detected faults due to SEU.
HW Erroneous Program Execution	Failure rate for detected and not detected faults due to permanent faults.
SEU Erroneous Data Computation	Failure rate for detected and not detected faults due to SEU.
HW Erroneous Data Computation	Failure rate for detected and not detected faults due to permanent faults.
Module reset due to a SEU	Failure rate for module reset due to a SEU.
SEU Erroneous IO	Failure rate for IO due to SEU.
HW Erroneous IO	Failure rate for IO due to permanent faults.

6. Metrics selection

This section represents one of the core activities of this deliverable. It presents the selection of metrics that are considered as relevant for each application domain. In this final version of the deliverable (D2.4.2) the selection has been revised and focused on a reduced set of important metrics for the system's characterization.

6.1. Metrics for HPC

When thinking about HPC, the first thought is performance and, specially, performance measured in MFLOPS (Million Floating Point Operations Per Second) or its derivatives (TeraFLOPS, ExaFLOPS, etc.). Lately, it is of prime concern the energy consumption of such big infrastructures. Thus, Watts, as the basic measure of power consumption is widely used. If we combine both power and performance, we move into the area of energy-delay metrics (PDP, EDP, ED^{2}P). Figure 12 shows graphically the relationship between system power metrics and system performance to measure system efficiency.

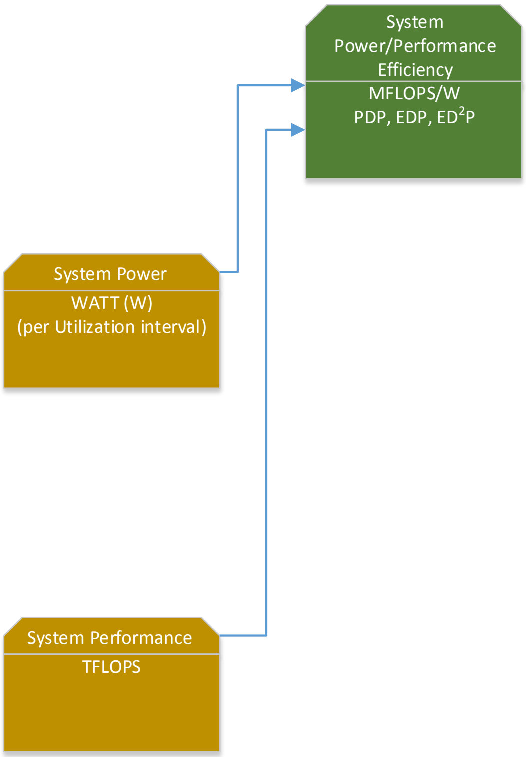


Figure 12: Fundamental HPC Power and Performance Metrics[5]

On top of that, HPC system costs are skyrocketing; thus, it is of key importance to reduce the costs of setting up an HPC system (buying the parts) and the costs of running it. We will consider these costs as a function of the number of processors (i.e., systems) built into the HPC system. The blue boxes in Figure 13 depict the number of processor, the processor cost and the total cost of purchasing. The number of processor will also affect the performance and power of the system. Consequently, the links between these two metrics appear on the figure.

[5] *While the layout of the figure it is not optimized, it is in the interest of the next figures that will add other metrics to keep a constant layout.*

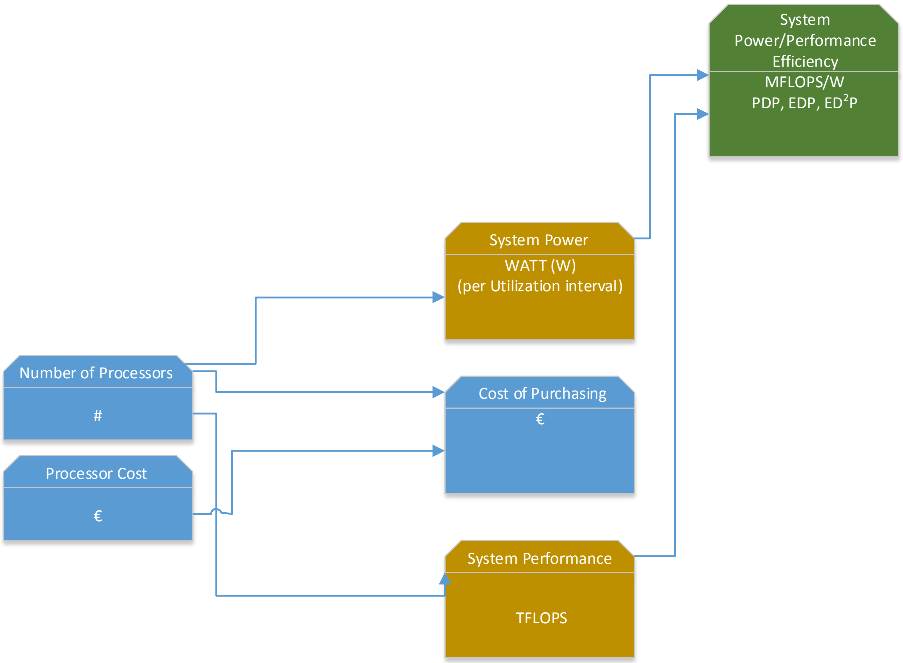


Figure 13: HPC system metrics (cost, power, performance and efficiency)

Apart from setting up the system, HPC systems have high running costs. These costs are summarized in the "Total Cost of Ownership" (TCO) metric that is expressed in euros/year. The TCO lumps the entire running costs and, in its turn, depend on the costs of the system (including repairs) plus the electricity bill (directly related to the system power). Through our bibliography study, we found out that Hardy et al. propose an analytical model to compute the TCO based on the parameters defined above (Hardy, Kleanthous, Sideris, Saidi, Ozer, & Sazeides, 2013). This will be the initial used model and, it will be extended if necessary.

Once we know the costs, a laboratory or data center with such a system will divide the costs on the users based on their occupation of the system. Thus, we have to add two new metrics: utilization and productivity (defined as amount of work done i.e. TFLOPS per euro). Figure 14 summarizes the relationships of these new metrics to the ones in the previous figure.

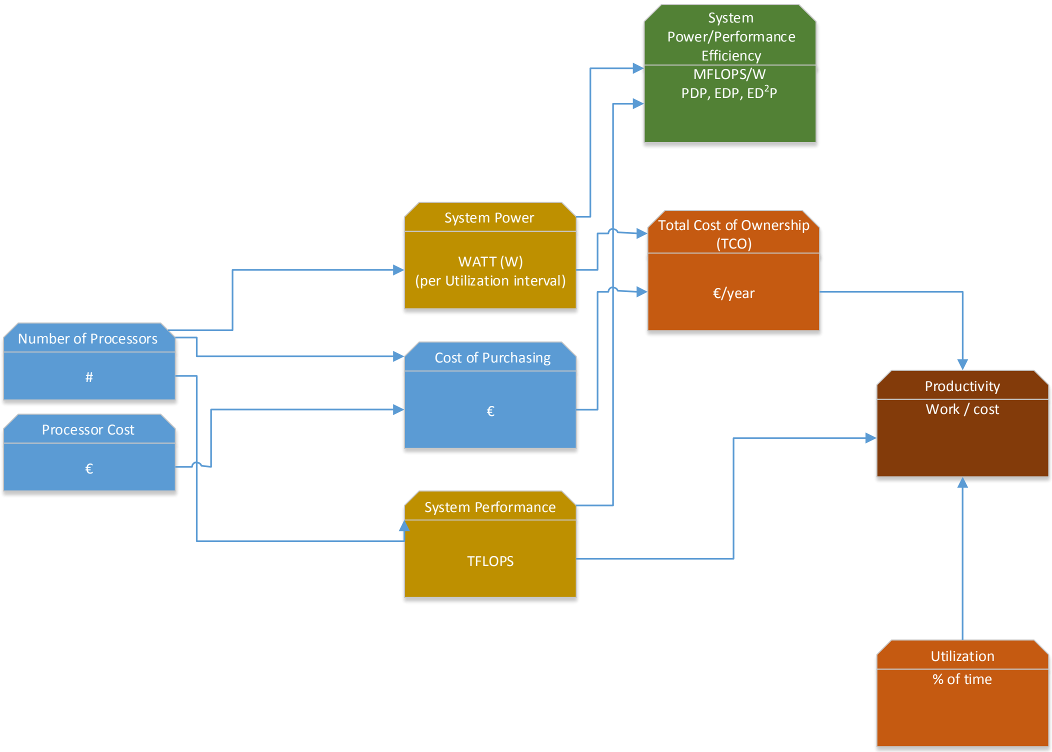


Figure 14: HPC system metrics (cost, power, performance, efficiency and productivity)

Directly directed with the objectives of CLERECO is system reliability evaluation. Reliability will impact the availability of the system and, thus, the utilization. Reliability metrics will estimated through methods that will be defined in the framework of WP5 activities (as the other metrics colored light brown in the figures). On top of that, there will be the inputs to the overall system evaluation (in blue). Finally, the different shades of red indicate the ones that will need the contribution of this task to be defined (or computed) from the preceding ones.

For instance, availability (described as percentage of the time over a year that the system is working –assuming it runs all time) translates to the following downtime per year. Also, we have added some examples of usage domain. Figure 15 summarizes our findings.

9's	Availability	Downtime/year	Examples
1	90.0%	36 days, 12 hours	Personal Computers
2	99.0%	87 hours, 36 min	Entry Level Business
3	99.9%	8 hours, 45.6 min	ISPs, Mainstream Business
4	99.99%	52 min, 33.6 sec	Data Centers
5	99.999%	5 min, 15.4 sec	Banking, Medical
6	99.9999%	31.5 seconds	Military, Defense

Figure 15: Availability needs and computing domain

Availability is actually the addition of the time spent on servicing (i.e., recovering from errors) plus the actual fault-free running time. Servicing, then, it is computed from the system reliability measurements, as well as recovery time.

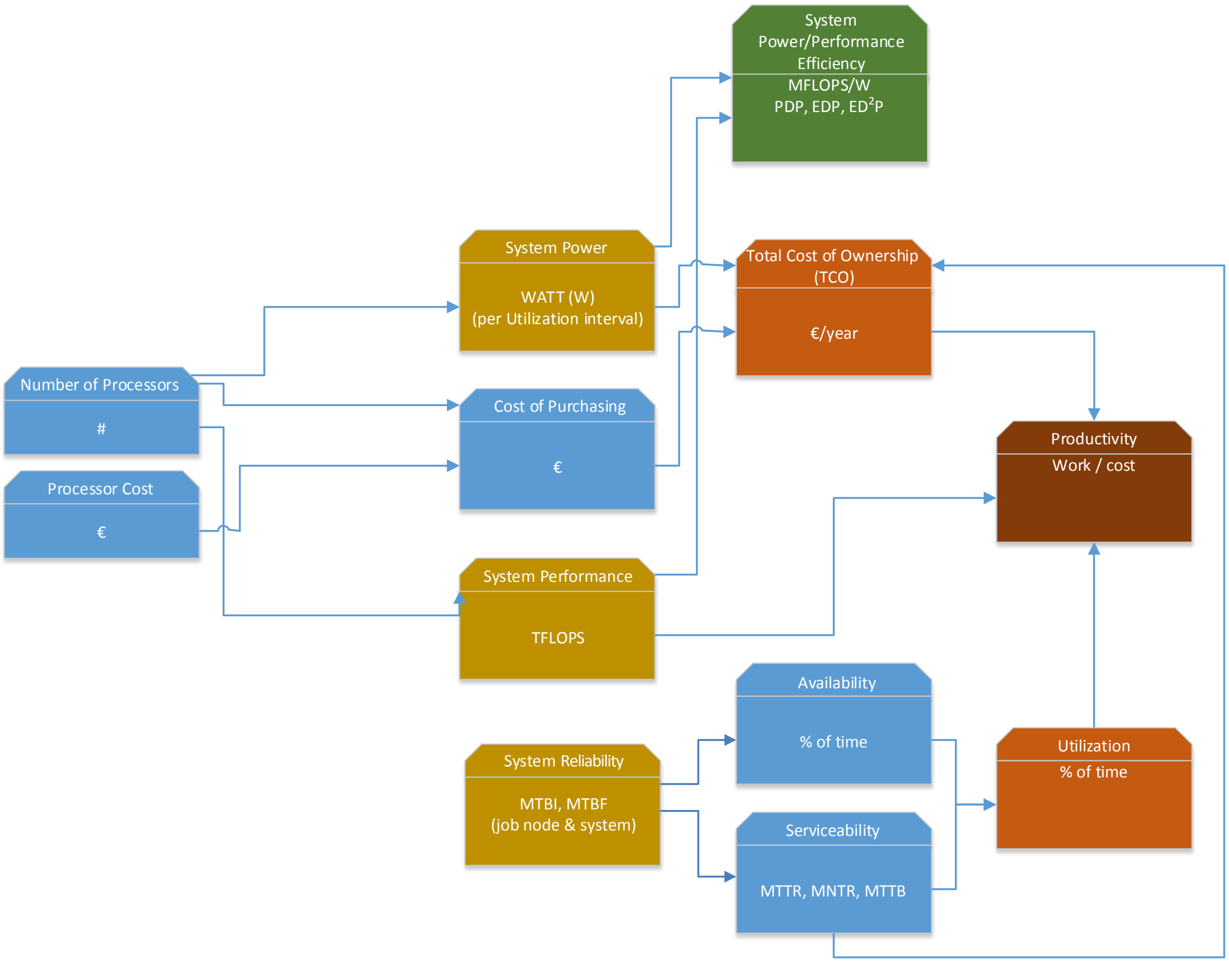


Figure 16: HPC system metrics (plus system reliability, availability and serviceability)

One of the big costs of running an HPC facility is running the cooling system. The cooling system is designed so that computers stay at the correct running temperature (~35-40ºC). The cooling capability depends directly of the number of devices in the HPC center and thus, it can be defined as a function of the number of processors in the system. At the same time, buying such cooling equipment will be part of the total cost of setting up an HPC site. Thus, it will be added into the cost of purchasing. Figure 17 shows the addition of processor metrics and cooling into the overall HPC system metric scenario.

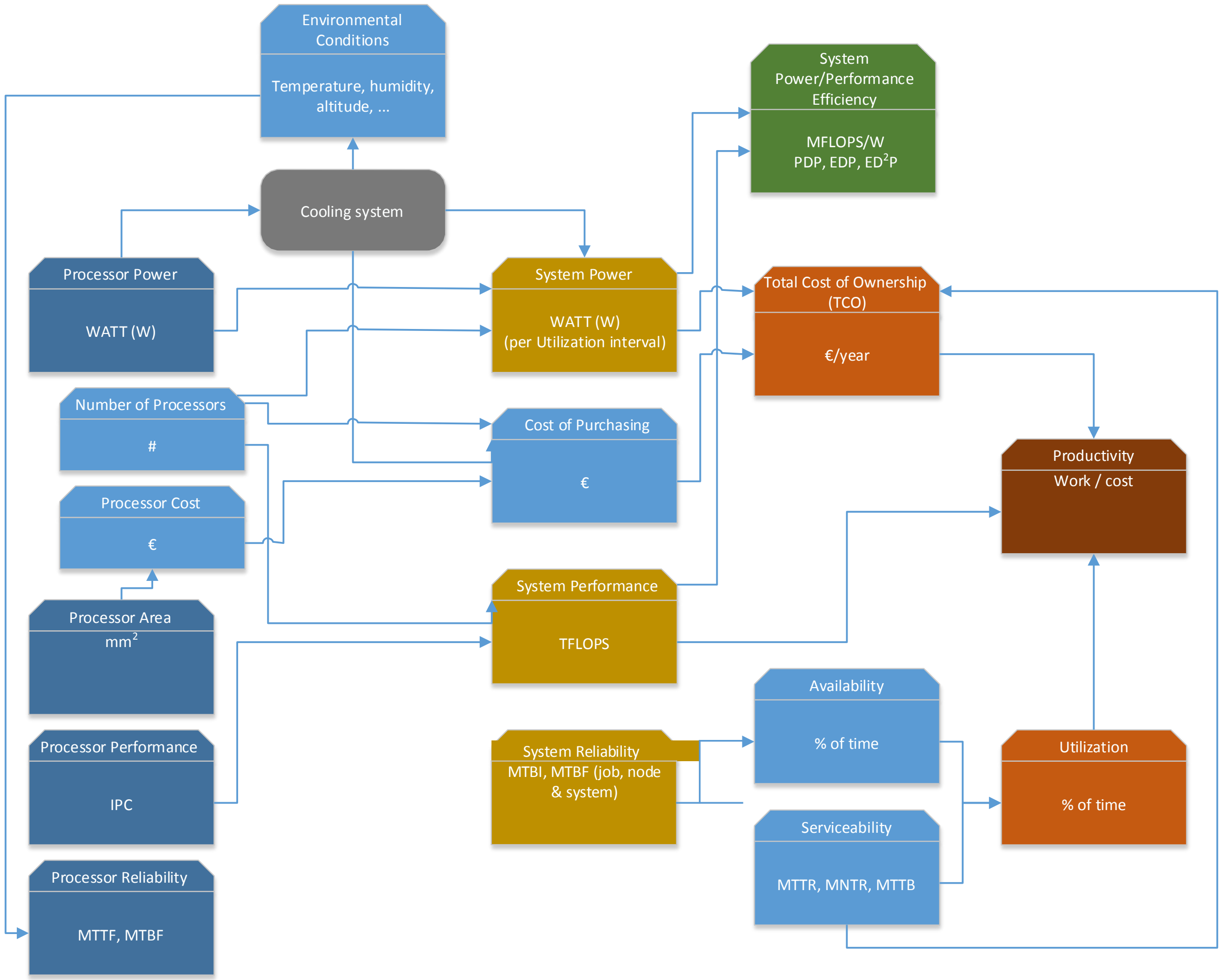


Figure 17: HPC system metrics (plus processor metrics and cooling)

Finally, faults and errors are not only caused by the processor (and memory) system. Thus, network failures/breakages, power outages and other component failures must be brought into the final equation. Figure 18 shows the final picture of all metrics involved in the characterization of an HPC system. This is a complete picture of all the bits and pieces involved in the final evaluation of an HPC system. Figure 18 also shows the relationship between one metric and the rest. The work in this task has identified the sources, the metrics of interest and their relationships.

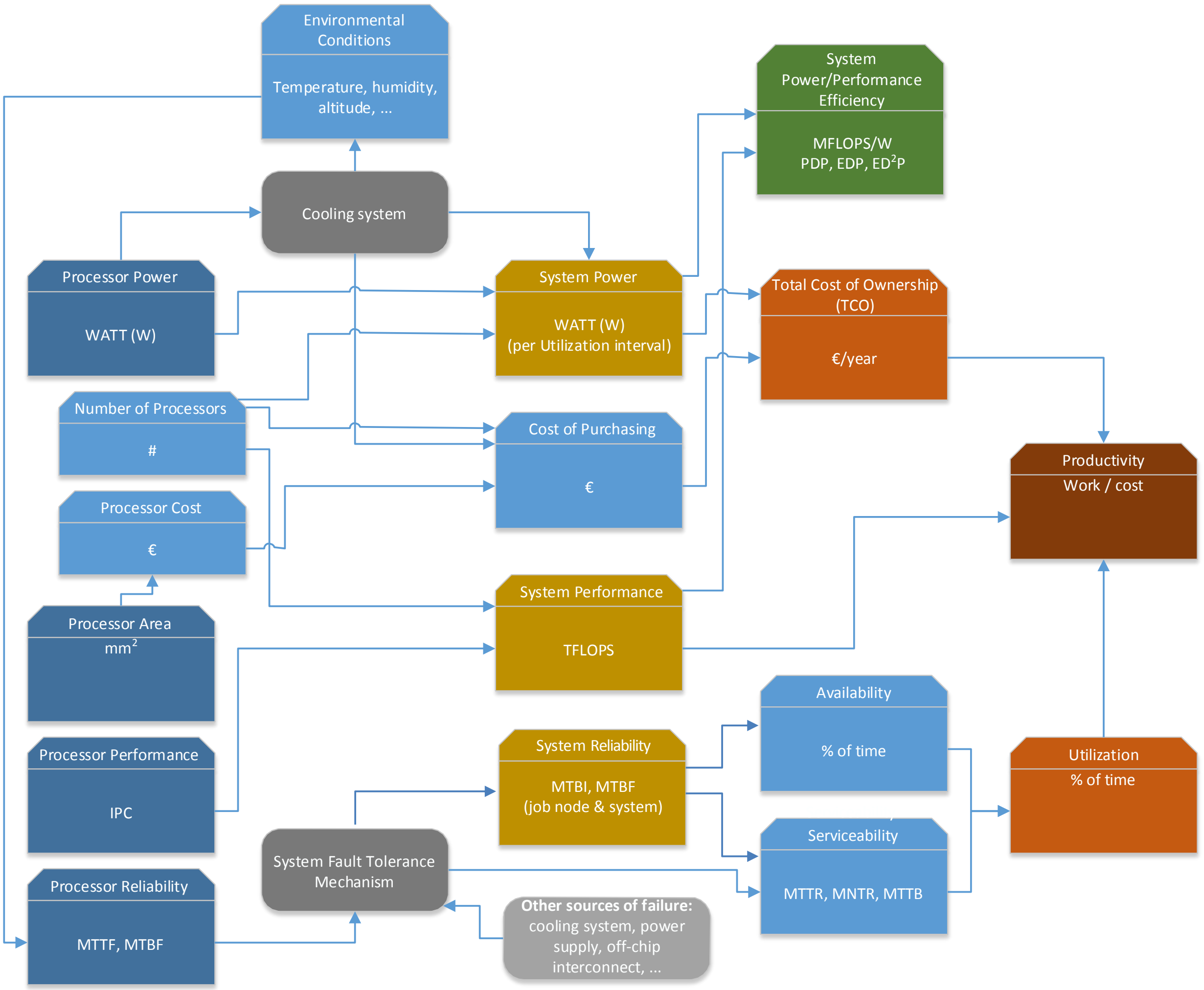


Figure 18: HPC system metrics (plus other sources of failures)

6.2. Metrics for industrial applications

In industrial sector, the safety metrics related to the cross-sector safety standard (i.e., IEC 61508 [11]) are typically adopted and are generally considered to be sufficient. These metrics include (but are not limited to)

1. Top-level safety metrics for subsystems mentioned in Table 10.
2. Parameters in Table 11 (serving as input parameters for the calculation of the top-level safety metrics given in Table 10).

6.3. Metrics for mission-critical systems

In the mission-critical and safety-critical sectors, the CLERECO tool suite will support the system design to improve different attributes. A set of different metrics is thus required for the different use of the methodology that can be done.

Preliminary Safety Assessment: For a Preliminary Safety Assessment, the important metric is of course the global failure rate of the system as it will be the determining criteria of the safety of the system. However, individual failure rates corresponding to the different failures modes as safety analysis methods, based on Fault Tree Analysis and Failure Modes and Effects Analysis (FMEA), are required, too.

SW programming and HW design: An estimation of the vulnerability factors is needed for planning a failure rate budget and assisting the designer.

Table 14 lists the metrics that are required by the mission-critical domain.

Table 14: Main metrics for mission-critical systems

Name/metric	Utilization/Applicable for
Silent Data Corruption (SDC) rate	Used to assess the safety of the system (see section 5.2). Failure rates have to be given for the different failure modes of the system.
Detected Unrecoverable Error (DUE) rate	Used to assess the safety of the system (see section 5.2). Failure rates have to be given for the different failure modes of the system.
Timing failure rate	To evaluate the impact of faults on the real-time behavior of hard real-time systems.
Program Vulnerability Factor (PVF)	To detect weak points of the application.

These metrics should be declined according to the different failure modes of the application. For instance, the Table 15 lists the metrics for the mission-critical case study of the work package 6.

Table 15: Failure modes of the FMS application

Name	Description
Incorrect display message format (DUE)	*A message to the display has an incorrect format:* • *Bad number of arguments* • *Display code not valid*
Corrupted display message (SDC)	*A message with wrong display information has been sent to the display.* • *Wrong localization* • *Bad nearest airports list*
Guidance failure	*Wrong guidance command sent to the autopilot.* [redacted]

Timing failures	*Violation of the timing requirements of the application.*
Crash or unresponsiveness of the application	*The application has crashed or is not responsive.*

6.4. Selected reliability metrics

Table 16 lists and summarizes the metrics that have been selected as interesting metrics for the CLERECO framework. It is important to remark here, that this list has been established in order to focus the project's resources on the most important metrics whose early evaluation is feasible in the project timeframe.

Table 16: Preliminary selection of reliability metrics

Name/metric	**Utilization/Applicable for**
Power	System power (per utilization interval)
Performance	System performance (TFLOPS)
System reliability	MTBI, MTBF per job, node and system
PFH	Average frequency of dangerous failure
PFD_{avg}	Average probability of dangerous failure on demand, mean unavailability.
SFF	Safe Failure Fraction**, SFF = $(\Sigma\lambda_s + \Sigma\lambda_{dd}) / \Sigma\lambda_{tot}$
DC	Diagnostic coverage*, DC = $\Sigma\lambda_{dd} / \Sigma\lambda_d$
HFT	Hardware Fault Tolerance (Input parameter)
T1	Proof test interval
T2	Diagnostic time interval
λ_s	Failure rate corresponding to safe failures
λ_d	Failure rate corresponding to dangerous failures
λ_{dd}	Failure rate corresponding to dangerous detected failures
λ_{du}	Failure rate corresponding to dangerous undetected failures
λ_{sd}	Failure rate corresponding to safe detected failures
λ_{su}	Failure rate corresponding to safe undetected failures
λ_{tot}	$\lambda_d + \lambda_s$
β	Fraction of undetected failures that have a common cause (Common cause factor for subsystems with HFT > 0). See [11].
β_d	Fraction of detected failures that have a common cause. See [11].
MTTR	Mean time to restoration, [11]
MRT	Mean repair time, [11]
Silent Data Corruption (SDC) rate	Used to assess the safety of the system (see section 5.2). Failure rates have to be given for the different failure modes of the system.
Detected Unrecoverable Error (DUE) rate	Used to assess the safety of the system (see section 5.2). Failure rates have to be given for the different failure modes of the system.
Timing failure rate	To evaluate the impact of faults on the real-time behavior of hard real-time systems.

Program Vulnerability Factor (PVF)	To detect weak points of the application.

7. Conclusion

The CLERECO project will provide new means for the early reliability evaluation of the system, and as a result will set-up the foundations for the elaboration of new design approaches. So, the use of the right metrics to assess the design decisions is very important in order to drive the design of reliable systems in an optimal way. The questions of the accuracy of the tools and of the selection of metrics are indeed critical points, as non-accurate reliability estimations or non-relevant metrics could lead the designers to take wrong design decisions and non-optimal results.

From this point of view, the tools and the methodology should allow the designer to evaluate the impact of its decisions on the final reliability of the system, but also to understand in which way they impact the reliability. The metrics have to be derived for each component in order to identify what are the weak points of the system, and to understand why the improvement of protection mechanisms could have lower benefits than anticipated.

Reliability and dependability metrics that are traditionally used in different application domains, at the system level as well as the HW and SW levels have been presented and listed in this document.

New metrics have been explored to evaluate the impact of faults on others characteristics of the system that are traditionally not taken into account. With the evolution of technologies, this traditional approach is becoming less and less appropriate to our requirements. The metrics will be used to drive the design, and not just to validate that the reliability is satisfying at a post-design stage.

A selection of metrics that are required in each application domain has been presented in the last section, and represents the outcomes of the task T2.3 **"Definition of system level reliability metrics"**. This definition of metrics lists the requirements regarding the outcomes of the CLERECO tools. This list of metrics has been refined and continuously "adapted to" the feedback received from others WPs and from the final end users. So, this deliverable updates the deliverable D2.4.1 to take into account the continuous feedback from others work packages.

8. Acronyms

Acronym	Full text
AVF	Architectural Vulnerability Factor
ACE	Architecturally Correct Execution
DPM	Dynamic Power Management
DUE	Detected Unrecoverable Error
EC	Embedded Computing
DC	Diagnostic Coverage
DAL	Design Assurance Level
EMI	Electromagnetic Interference
FD-SOI	Fully Depleted Silicon On Insulator
FinFET	Fin-Shaped Field Effect Transistor
FIT	Failures In Time
FMEA	Failure Modes and Effects Analysis
FMES	Failure Modes and Effects Summary
FTA	Fault Tree Analysis
HFT	Hardware Fault Tolerance
HPC	High Performance Computing
HVF	Hardware Vulnerability Factor
MBU	Multiple Bit Upset
MRT	Mean Repair Time
MTBF	Mean Time Between Failures
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
RRAM	Resistive Random-Access Memory
PFD	Probability of Failure on Demand
PFH	Probability of Failure per Hour
PVF	Program Vulnerability Factor

PVF	Performance Vulnerability Factor
SDC	Silent Data Corruption
SEFI	Single Event Functional Interrupt
SEL	Single Event Latch-Up
SET	Single Event Transient
SEU	Single Event Upset
SFF	Safe Failure Fraction
SoC	System-on-Chip
TCO	Total Cost of Ownership
TTF	Time To Failure

9. References

[1] C. Auth, et al., "A 22nm high performance and low-power CMOS technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density MIM capacitors," in 2012 Symposium on VLSI Technology (VLSIT), 2012.

[2] N. Planes, et al., "28nm FDSOI technology platform for high-speed low-voltage digital applications," in 2012 Symposium on VLSI Technology (VLSIT), pp.133-134, 2012.

[3] S.-S. Sheu, et al., "A 4Mb embedded SLC resistive-RAM macro with 7.2ns read-write random-access time and 160ns MLC-access capability," 2011 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp. 200-202, 2011.

[4] D. Humphrey, W. Schawlee, P. Sandborn, and D. Lorenson, "Utilization life of electronic systems - aging avionics usable life and wear-out issues", presented at World Aviation Congress, 2002.

[5] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," IEEE Micro, vol. 25, pp. 10-16, 2005.

[6] 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, Jan.-March 2004.

[7] S.S. Mukherjee, J. Emer, S.K. Reinhardt, "The soft error problem: an architectural perspective," 11th International Symposium on High-Performance Computer Architecture (HPCA-11), pp. 243-247, 12-16 Feb. 2005.

[8] D. Hardy, I. Sideris, N. Ladas, and Y. Sazeides, "The Performance Vulnerability of Architectural and Non-architectural Arrays to Permanent Faults," International Symposium on Microarchitecture (MICRO-45), pp. 48-59, Dec. 2012.

[9] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor", in Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture (MICRO 36), pp. 29-42, 2003.

[10] RTCA and EUROCAE, "DO-254 / ED-80, Design Assurance Guidance for Airborne Electronic Hardware", 2000.

[11] IEC 61508, "Functional safety of electrical/electronic/programmable electronic safety-related systems", Edition 2.0, 2010.

[12] IEC 61511, "Safety instrumented systems for the process industry sector", Edition 1.0, 2003.

[13] IEC 62061, "Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems", Edition 1.0, 2005.

[14] ISO 13849, "Safety of machinery – Safety-related parts of control systems", Edition2.0, 2008.

[15] S. D. Millman, "Improving quality: yield vs. test coverage (WSI)," Fifth Annual IEEE International Conference on Wafer Scale Integration, pp. 279-288, 1993.

[16] G. S. Sohi, "Cache memory organization to enhance the yield of high performance VLSI processors," IEEE Transactions on Computers, vol. 38, no. 4, pp. 484–492, Apr. 1989.

[17] A. F. Pour and M. D. Hill, "Performance implications of tolerating cache faults," IEEE Transactions on Computers, vol. 42, no. 3, pp. 257–267,Mar. 1993.

[18] N.Foutris, D.Gizopoulos, J.Kalamatianos, V.Sridharan, "Assessing the Impact of Hard Faults in Performance Components of Modern Microprocessors," IEEE International Conference on Computer Design (ICCD), Asheville, NC, USA, Oct. 2013.

[19] G.Yalcin, O.Unsal, A.Cristal, "FaulTM: Error Detection and Recovery Using Hardware Transactional Memory," Design, Automation & Test in Europe (DATE-13), Grenoble, France, Mar. 2013.

[20] T.Li, R.Ragel, S.Parameswaran, "Reli: Hardware/Software Checkpoint and Recovery Scheme for Embedded Processors," Design, Automation & Test in Europe (DATE-12), Dresden, Germany, Mar. 2012.

[21] M. Slijepcevic, L. Kosmidis, J. Abella, E. Quinones, F.J. Cazorla, "DTM: Degraded Test Mode for Fault-Aware Probabilistic Timing Analysis," Euromicro Conference on Real-Time Systems (ECRTS-25), Paris, ECE, France, Jul. 2013.

[22] J. Abella, E. Quinones, F.J. Cazorla, Y.Sazeides, M.Valero, "RVC: A Mechanism for Time-Analyzable Real-Time Processors with Faulty Caches," International Conference on High-Performance and Embedded Architectures and Compilers (HiPEAC), Heraklion, Crete, Greece, Jan. 2011.

[23] N.J.Wang, S.J.Patel, "ReStore: Symptom-Based Soft Error Detection in Microprocessors," IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 3, pp. 188–201, Jul. - Sept. 2006.

[24] J.C.Solens, B.T.Gold, J.Kim, B.Falsafi, J.C.Hoe, A.G. Nowatzyk, "Fingerprinting: Bounding Soft-Error Detection Latency and Bandwidth," International conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-11), Boston, MA, USA, Oct. 2004.

[25] R.Rodrigues, A.Annamalai, I.Koren, S.Kundu, "A study on the Use of Performance Counters to Estimate Power in Microprocessors", IEEE Transactions on Circuits and Systems, Vol.60, Issue 12, 2013.

[26] F.Bower, D.Hower, M.Yilmaz, D.Sorin, S.Ozev, "Applying Architectural Vulnerability Analysis to Hard Faults in the Microprocessor", Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS/Performance, 2006.

[27] S. Mitra, N.R. Saxena, E.J. McCluskey, "A design diversity metric and reliability analysis for redundant systems," International Test Conference, pp. 662-671, 1999.

[28] D. Hardy, M. Kleanthous, I. Sideris, AG. Saidi, E. Ozer, Y. Sazeides, "An analytical framework for estimating TCO and exploring data center design space," 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp.54-63, 2013.

[29] J. Borcsok, S. Schaefer, E. Ugljesa, "Estimation and Evaluation of Common Cause Failures," Second International Conference on Systems (ICONS '07), 2007.

[30] S.R. Nassif, N. Mehta, and Y. Cao, "A Resilience Roadmap," DATE 2010.

[31] M. A. Levin, T. T. Kalal, "Improving Product Reliability: Strategies and Implementation", John Wiley & Sons Ltd, 2003.

[32] R.A. Shafik, J. Mathew, and D.K. Pradhan, "Introduction to energy efficient fault tolerant systems", in *Energy-Efficient Fault Tolerant-Systems*, J. Mathew, R. A. Shafik, and D. K. Pradhan (eds.), Springer, 2014, pp. 1-10.

[33] C. Constantinescu, "Intermittent Faults and Effects on Reliability of Integrated Circuits", in Proceedings of International Symposium on Reliability and Maintainability (RAMS), 2008.

[34] S. Mukherjee, Architecture Design for Soft Errors. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[35] V. Sridharan, D. R. Kaeli, "Using Hardware Vulnerability Factors to Enhance AVF Analysis", ACM/IEEE International Symposium on Computer Architecture (ISCA-37), Saint-Malo, France, June 21-23, 2010.

[36] P. Gupta, R.K. Gupta, "Underdesigned and Opportunistic Computing", Asian Test Symposium (ATS), pp.498-499, 2011.

[37] A. DeHon, H. M. Quinn, N. P. Carter, "Vision for cross-layer optimization to address the dual challenges of energy and reliability", Design, Automation & Test in Europe (DATE 2010), p. 1017-1022, March 2010.

[38] JEDEC Standard JESD-89A, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices", October 2006.

[39] R.A. Reed, M.A. Carts, P.W. Marshall, C.J. Marshall, O. Musseau, P.J. McNulty, D.R. Roth, S. Buchner, J. Melinger, and T. Corbiere, "Heavy ion and proton-induced single event multiple upset," IEEE Transactions on Nuclear Science, vol. 44, no. 6, pp. 2224–2229, 1997.

[40] G.Reis, J.Chang, N.Vachharajani, R.Rangan, D.August, S.S.Mukherjee, "Design and Evaluation of Hybrid Fault-Detection Systems", ACM/IEEE International Symposium on Computer Architecture (ISCA-32), Madison, Winsonsin, USA, June 4-8, 2005.

[41] C.Weaver, J.Emer, S.S.Mukherjee, S.K.Reinhardt, "Techniques to Reduce the Soft Error Rate of High-Performane Microprocessor", ACM/IEEE International Symposium on Computer Architecture (ISCA-31), Munich, Germany, June 19-23, 2004.